AIAA SciTech
5-9 January 2015, Kissimmee, Florida
53rd AIAA Aerospace Sciences Meeting



# A Comparison Between Local $h$-Refinement and a Novel $r$-Refinement Method

James R. Grisham IV,* Nandakumar Vijayakumar,† Guojun Liao,‡
Brian H. Dennis,§ and Frank K. Lu¶
*University of Texas at Arlington, Arlington, Texas 76019, USA*



**Two mesh refinement methods are applied to an inviscid test case and the results are compared with the exact solution from gasdynamics theory. One method is a local $h$-refinement that is native to ANSYS Fluent and the other is an $r$-refinement method called the deformation method. An insightful, flow-based explanation of the deformation method is provided. Results show that both methods are capable of improving the quality of the solution, but the $h$-refinement method increases the number of nodes in the domain and changes connectivity information. The deformation method accomplishes similar results by redistributing points within the mesh. The benefits and potential limitations of both methods are discussed.**

## Nomenclature

| | |
|---|---|
| $f$ | Monitor function |
| $J$ | Jacobian determinant |
| $\hat{\mathbf{n}}$ | Normal vector |
| PEC | Principle of error conservation |
| $t$ | Time |
| $\mathbf{v}$ | Node velocities |
| $V$ | Volume |
| $w$ | Potential function |
| $\varphi$ | Transformation |
| $\omega$ | Relaxation parameter |
| *Superscript* | |
| $n$ | Time level |
| $k$ | Iteration level |
| *Subscript* | |
| $i$ | Step in $x$-direction |
| $j$ | Step in $y$-direction |

## I. Introduction

An adequate discretization or mesh (amongst various factors) is required to ensure that the numerical solution of a partial differential equation is adequate. Because it is difficult to discern *a priori* the

*Graduate Assistant, Aerodynamics Research Center/CFD Lab, Department of Mechanical and Aerospace Engineering. Student Member AIAA.
†Graduate Assistant, Aerodynamics Research Center, Department of Mechanical and Aerospace Engineering. Member AIAA.
‡Professor, Department of Mathematics.
§Associate Professor and Director, CFD Lab, Department of Mechanical and Aerospace Engineering.
¶Professor and Director, Aerodynamics Research Center, Department of Mechanical and Aerospace Engineering. Associate Fellow AIAA.

regions where the solution will not be adequately resolved, a myriad of solution-based mesh adaptation schemes that allow for increased accuracy have been developed and are in wide use. Amongst these, only the subdivision and moving-mesh schemes are considered here. The former inserts points in regions of interest, changes connectivity information and increases the total number of points. The latter simply redistributes the points in the domain so that points are clustered towards regions of interest without affecting connectivity information.

Current methods for mesh adaptation include local refinement via mesh motion ($r$-refinement) and subdivision/coarsening schemes such as AMR ($h$-refinement). In both methods, a physical variable that is related to solution gradients or an error estimate is used to identify regions where refinement should occur. Implementation of subdivision schemes is complicated because the data structure must change because of changes in connectivity and total number of nodes. One way to bypass this data structure issue is to use mesh motion schemes which typically employ physics-based analogies such as the spring analogy[1,2] or linear elasticity PDEs[3] to move the mesh points toward areas of interest. Physics-based analogies for mesh motion work reasonably well, but have no guarantee of positive cell volumes. Some codes attempt to untangle folded meshes, but for large deformations, the mesh motion and untangling can fail thereby resulting in failure of the flow solver.

Liao and Anderson[4] introduced a new approach to mesh generation that makes use of concepts in differential geometry. The new method was based on previous work on volume elements on a Riemannian manifold by Moser.[5] The method has been successfully applied to mesh adaptation and moving meshes.[6,7,8,9,10,11] The deformation method is particularly useful for unsteady flows as compared to $h$-refinement schemes because it does not require any point-insertion or coarsening. The clustering travels along with the moving features in the flow.

The main benefits of the deformation method over other mesh movement schemes and the well-known elliptic grid generation method are: (1) It has solid mathematical foundation, namely, the ability to prescribe the Jacobian determinant in both two- and three-dimensions has been mathematically proven in the continuous sense, which implies that the method will create non-folding meshes if properly implemented; and (2) Its implementation is based on linear differential operators such as the scalar Laplacian operator. Being able to prescribe the Jacobian allows indirect, but practical control over the point distribution. The implementation of the deformation method is straightforward and involves solution of a scalar Poisson equation with Neumann boundary conditions and dynamic ordinary differential equations.

The goal of the current work is to compare the deformation method refinement with a local subdivision scheme native to ANSYS® Fluent®. An insightful explanation of the deformation method is provided, and a simple, inviscid test case involving supersonic flow in a duct is used to compare the results from the adaptation methods with inviscid gasdynamics theory. The pros and cons of both methods are discussed.

### A. Background

Suppose there is an initial uniform mesh on a domain $\Omega$. Let $f(\xi,\eta,\zeta,t) > 0$ be a size function (or monitor function) which is small in regions where the error is large. In order to enhance the mesh resolution where the error is large, the initial mesh is deformed by a family of transformations $\varphi(\xi,\eta,\zeta,t)$ for $t > 0$ such that

$$J(\varphi) = f(\varphi,t) \quad t > 0 \tag{1}$$

where $J(\varphi) > 0$. The family of transformations given by $\varphi(\xi,\eta,\zeta,t)$ can be viewed as a "mesh flow" that is independent of the fluid flow with a velocity field given by

$$\mathbf{v}(\xi,\eta,\zeta,t) = \frac{\partial\varphi}{\partial t} \tag{2}$$

The method for determining the proper velocity field so that (1) is satisfied is discussed next. Let $W$ be any cell of the initial mesh, and let $\varphi_t(W)$ be the deformed cell of $W$ under the transformation $\varphi(\xi,\eta,\zeta,t)$. A proper velocity field is determined by the following *principle of error conservation* (PEC) which is equivalent to the equidistribution principle:[12,13]

$$\frac{d}{dt}\int_{\varphi_t(W)} \frac{1}{f}\, dV = 0 \quad t > 0 \tag{3}$$

The "mesh flow" represents changes in the point distribution of the mesh which are accomplished so that the total error is conserved. Figure 1 depicts the deformation of a quadrilateral cell $W$ and its image $\varphi_t(W)$

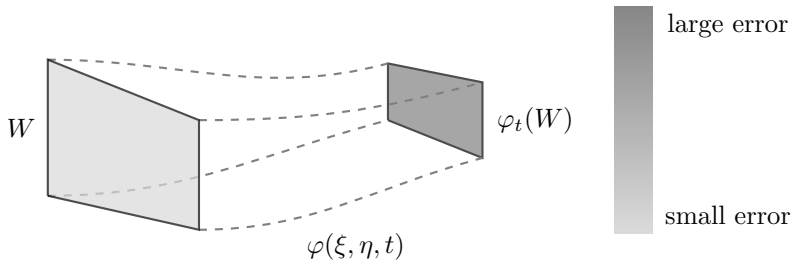



**Figure 1. Diagram of "mesh flow."**

under the "mesh flow" $\varphi(\xi,\eta,t)$ which both contain the same amount of error represented by $1/f$. The shades of gray correspond to different levels of error.

For purposes of the PEC analogy, $1/f$ is replaced by density $\rho$, and (3) becomes the conservation of mass under the "flow" $\varphi(\xi,\eta,\zeta,t)$; hence,

$$\frac{d}{dt}\int_{\varphi_t(W)} \rho\, dV = 0 \quad t > 0 \tag{4}$$

The differential form of (4) is hence the familiar continuity equation

$$\frac{\partial \rho}{\partial t} + \nabla\cdot(\rho\mathbf{v}) = 0 \tag{5}$$

The differential form of (3) is then

$$\frac{\partial}{\partial t}\left(\frac{1}{f}\right) + \nabla\cdot\left(\frac{\mathbf{v}}{f}\right) = 0 \tag{6}$$

Letting $\mathbf{u} = \mathbf{v}/f$ and rearranging, (6) becomes

$$\nabla\cdot\mathbf{u} = -\frac{\partial}{\partial t}\left(\frac{1}{f}\right) \tag{7}$$

Assuming $\nabla\times\mathbf{u} = 0$, a potential function $w$ can be defined whereby

$$\nabla w = \mathbf{u} \tag{8}$$

Inserting (8) into (7) yields

$$\nabla^2 w = -\frac{\partial}{\partial t}\left(\frac{1}{f}\right) \tag{9}$$

which is a scalar Poisson equation. The Neumann boundary condition on $\partial\Omega$

$$\nabla w\cdot\hat{\mathbf{n}} = 0 \tag{10}$$

is applied so that the nodes can move tangential to the boundaries, but not normal. The scalar Poisson equation with Neumann boundary conditions is then solved to determine nodal velocities used to deform the mesh. A proof is provided in Appendix A which shows that (1) holds under (2), (8) and (9).

## B. Mesh adaptation process

In general, mesh adaptation for steady problems can be described using the diagram in Fig. 2. The geometry and mesh are created, the flow solve is accomplished and the mesh is adapted according to the error estimates extracted from the initial solution. After adapting the mesh, the flow solver is run again to evaluate the effect of the adaptation on the solution. A physical variable of interest is chosen (in this case, total pressure ratio) and a loop between adaptation and the flow solver is iterated until the variation in the chosen variable is sufficiently small. In other words, if the chosen variable is not changing with further adaptation, the iterative process has converged.

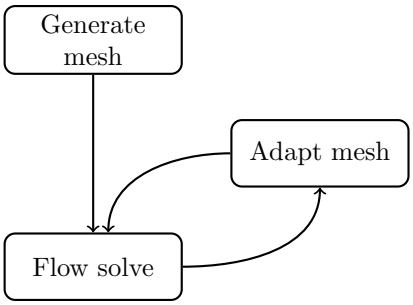



**Figure 2. Diagram of the mesh adaptation process.**



## II. Implementation for steady simulations

The previously described deformation method can be used for dynamic simulations where $t$ is the same $t$ used in the simulation or it can be used in static mode where $t$ is artificial time. In the work reported here, the method was applied to a steady case so $t$ is artificial time. The finite difference method with the appropriate transformations was used in this implementation. The inverse transformations between the physical and computational planes are given by

$$x = x(\xi, \eta) \tag{11a}$$
$$y = y(\xi, \eta) \tag{11b}$$

The monitor function used to adapt the mesh is formed as follows:

$$f_0 = \frac{C_0}{1 + C|\nabla p|} \tag{12}$$

The goal of the adaptation is to generate a mesh with $J = f_0$. In (12), $C$ is a constant controlling the strength of adaptation, and $C_0$ is a normalization constant which is determined so that $f_0$ satisfies the compatability condition given in (16) with $t = 1$. To compute the gradient of the chosen flowfield variable, a second-order accurate finite difference method with the transformations given in (11) was used. The monitor function $f_0$ represents the final point distribution desired. Next, artificial time $t$ is introduced and an intermediate function, $\widetilde{f}$, is constructed as follows:

$$\widetilde{f} = 1 - t + t f_0 \quad \text{for} \quad 0 \le t \le 1 \tag{13}$$

At $t = 0$, (13) yields $\widetilde{f} = 1$. At $t = 1$, (13) yields $f_0$ which is the desired point distribution. In discrete form,

$$\widetilde{f}^n = 1 + t^n - t^n f_0 \tag{14a}$$
$$\widetilde{f}^{n+1} = 1 + t^{n+1} - t^{n+1} f_0 \tag{14b}$$

where

$$t^{n+1} = t^n + \Delta t \tag{15}$$

The number of time steps is chosen and the function $\widetilde{f}$ is normalized at each time step so that the following condition is satisfied (in two dimensions for this work):

$$\iint \frac{1}{\widetilde{f}}\, dA = |\Omega| \tag{16}$$

In (16), $|\Omega|$ is the total area of the domain. Application of (16) yields

$$f = \frac{\widetilde{f}}{|\Omega|} \iint \frac{1}{\widetilde{f}}\, dA \tag{17}$$

where $f$ is the monitor function. Equation (17) is required for $f$ to be a Jacobian; also it enforces compatibility[a] so that the solution of the Poisson equation is unique. This simple normalization is a critical step in the application of the deformation method. Next, the right hand side of (9) is determined using first-order accurate finite differencing:

$$-\frac{\partial}{\partial t}\left(\frac{1}{f}\right) \approx \frac{1}{\Delta t}\left(\frac{1}{f^{n+1}} - \frac{1}{f^{n}}\right) \tag{18}$$

Poisson's equation is solved using second-order accurate finite differencing and the successive over-relaxation (SOR) method. The difference equation for the interior points is given by

$$w_{i,j}^{k+1} = \frac{1}{4}\left(w_{i+1,j}^{k} + w_{i-1,j}^{k+1} + w_{i,j+1}^{k} + w_{i,j-1}^{k+1} - h^2 g_{i,j}\right) \tag{19}$$

where $g_{i,j}$ is the source term determined in (18). For the present work, the relaxation parameter was set to 1.2.

After solving Poisson's equation for $w$, the node velocities must be computed. The velocity field is given by

$$\mathbf{v} = f\mathbf{u} = f\,\nabla w \tag{20}$$

The new node coordinates are determined by solving first-order ODEs in the spatial coordinates.

$$\frac{d\mathbf{x}}{dt} = \mathbf{v} \tag{21}$$

This is accomplished using Euler's method. In two dimensions, the numerical solution is given by

$$x^{n+1} = x^{n} + v_x\,\Delta t \tag{22a}$$

$$y^{n+1} = y^{n} + v_y\,\Delta t \tag{22b}$$

Time is then incremented by $\Delta t$ and the process beginning with normalizing the monitor function is repeated until $t = 1$. A diagram of the algorithm is provided in the Appendix.

### A. Deformation method code

A general C++ code was developed to implement the deformation method. Employing object-oriented programming enabled development of a code that is easily extensible and stand-alone testing of each method before it was implemented in the broader code.

### B. Test case

The test case used to compare the adaptation is a duct with a ramp and oblique shock reflection. The solution from inviscid gasdynamics will be compared with the results from Fluent's adaptation and the deformation method adaptation. A diagram of the geometry can be seen in Fig. 3. The test case was designed so that the reflecting oblique shock cancels at the bottom of the ramp. A Mach number of 3 was used for the simulations.

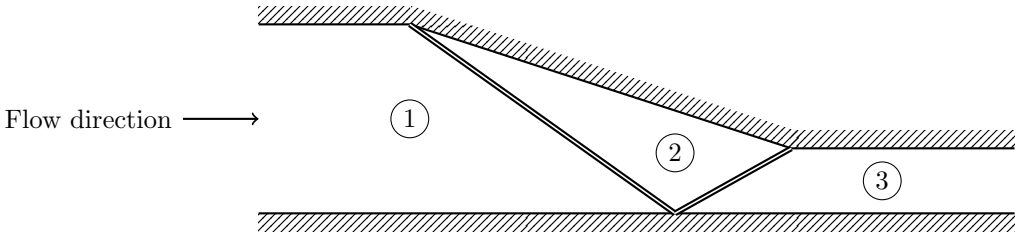



**Figure 3. Test case geometry.**

[a]Also known as a gage condition.

### C. Computational Tools

#### 1. *CFD General Notation System (CGNS)*

The developed code is linked with CGNS libraries so that binary CGNS files can be read and written. Use of CGNS formatted files allows for a unified file format that is compatible with mesh generation software, CFD codes, and post-processing software.

#### 2. *Pointwise®*

A series of scripts in the Pointwise® Glyph language were developed to facilitate rapid generation of high-quality structured meshes. The scripts used the elliptic mesh generation capabilities of Pointwise.

#### 3. *ANSYS® Fluent®*

ANSYS® Fluent®is a popular, commercial flow solver that is capable of solution-based mesh adaptation. Three options are available for mesh adaptation within Fluent®, namely, gradient-based, isovalue-based and curvature-based. The static, gradient-based approach is used for this project since the test case involves shock discontinuities in the flow field. The mesh adaptation is accomplished by specifying a flow variable, the limits of the gradient, and choosing either hanging node or conformal mapping adaptation. Hanging node adaptation was chosen for the case presented in paper.

To locate regions where refinement should occur, Fluent®uses an error estimate given by

$$|e| = A^{r/2}|\nabla f| \tag{23}$$

where $A$ is the cell area, $r$ is a weight factor, and $f$ is the desired field variable (in this case, static pressure).[14] This error estimate is normalized by the maximum value of $\nabla f$ in the domain. The density-based, explicit solver with Roe's flux difference splitting and second-order spatial discretization was used to solve Euler's equations.

#### 4. *CFL3D*

CFL3D is a structured, three-dimensional, finite volume flow solver with a wide range of features that was developed at NASA Langley Research Center. The Euler solver within CFL3D was used in this work and is implemented using a semi-discrete finite-volume formulation.[15] Second-order accurate spatial differencing is used for this work. Flux-difference splitting was accomplished using Roe's scheme and the smooth inviscid flux limiter was used to avoid oscillations in the numerical solution due to the shock waves. Because CFL3D only writes out a vector of conserved variables, density was used to form the monitor function for the deformation method rather than pressure. This is acceptable because both, density and pressure, increase rapidly because of the shock compression.

## III. Results

Simulations were performed on the same initial structured grid using ANSYS® Fluent®with local $h$-refinement and CFL3D with iterative deformation method $r$-refinement. The dimensions of the initial mesh

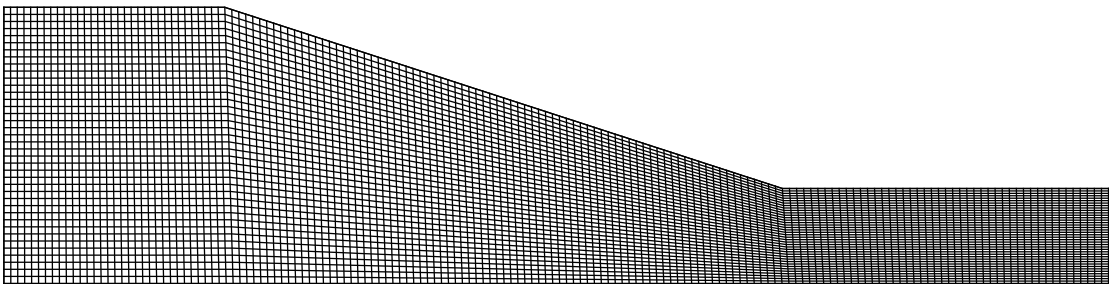


**Figure 4. Initial mesh.**

are $160 \times 40$. The mesh can be seen in Fig. 4. Contours of pressure for the initial mesh using Fluent®and CFL3D can be seen in Figures 5 and 6, respectively.

In an inviscid flow, shock waves are assumed to be infinitely thin discontinuities.[16] Initial results from both flow solvers show poorly resolved shocks (i.e., they are smeared).

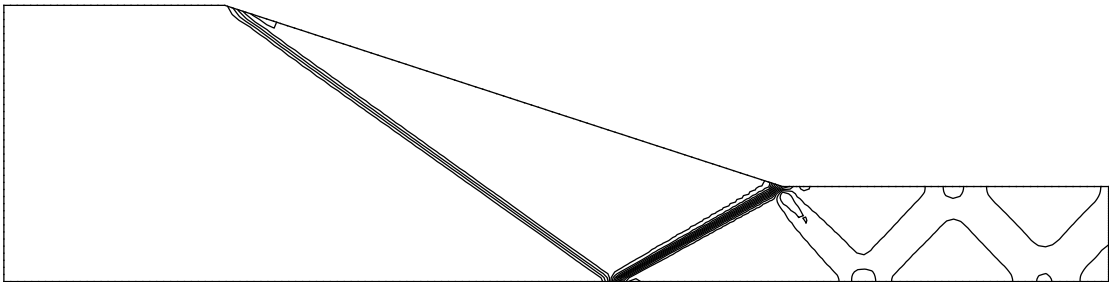


**Figure 5. Contours of pressure on the initial mesh from Fluent.**

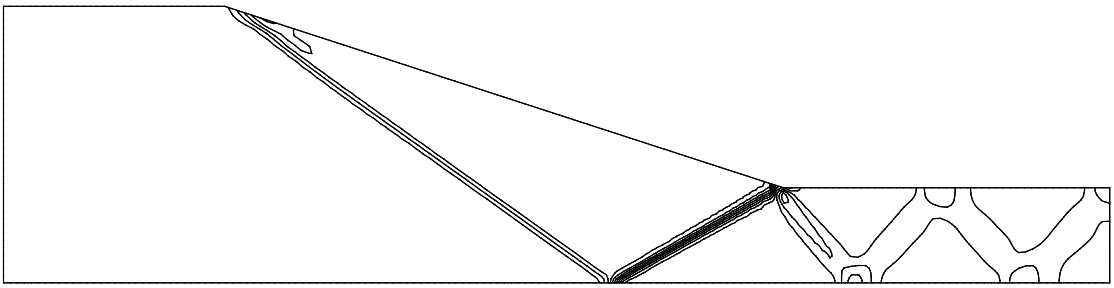


**Figure 6. Contours of pressure on the initial mesh from CFL3D.**

## A. Deformation method adaptation

The deformation method refinement was applied iteratively (five iterations) with one iteration consisting of refining the grid from the previous solution and running a flow solve using CFL3D. The final, deformed mesh is shown in Fig. 7. The final mesh shows clustering toward the location of the shock.

Examining the contours of density in Fig. 8, it can be seen that the final deformed grid resolves the shock much better than the initial grid.

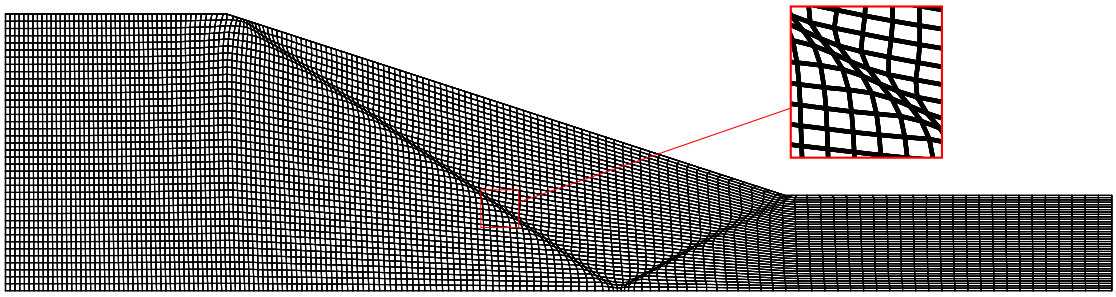


**Figure 7. Final deformed mesh.**

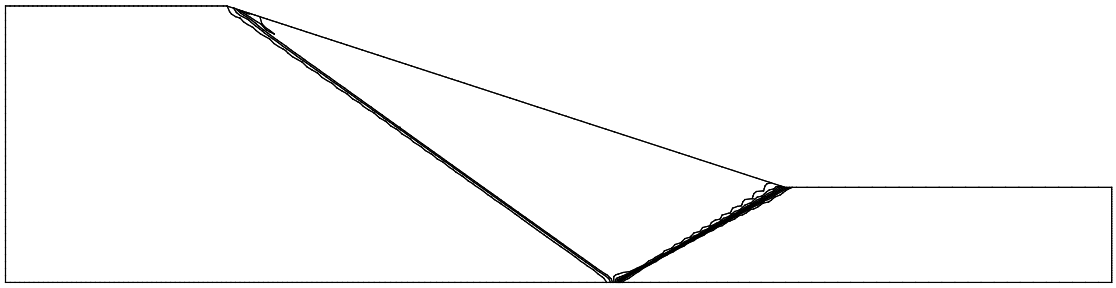


**Figure 8. Contours of pressure on the deformed mesh.**



## B. Fluent adaptation

Four flow solves were accomplished using ANSYS® Fluent®, one for the initial mesh and three refining steps. The hanging node adaptation was used and the mesh was adapted to the gradient of pressure. The final mesh can be seen in Fig. 9. The contours of pressure on the refined mesh can be seen in Fig. 10. A similar trend of better resolved, thinner shocks on the adapted mesh is evident here as well.

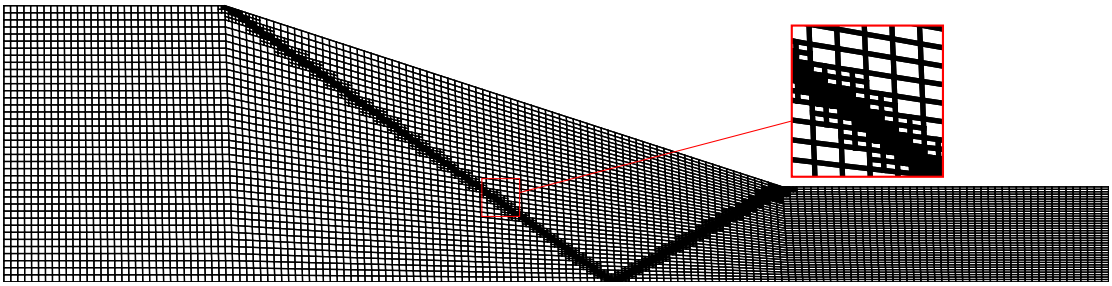


**Figure 9. Final mesh refined by Fluent.**

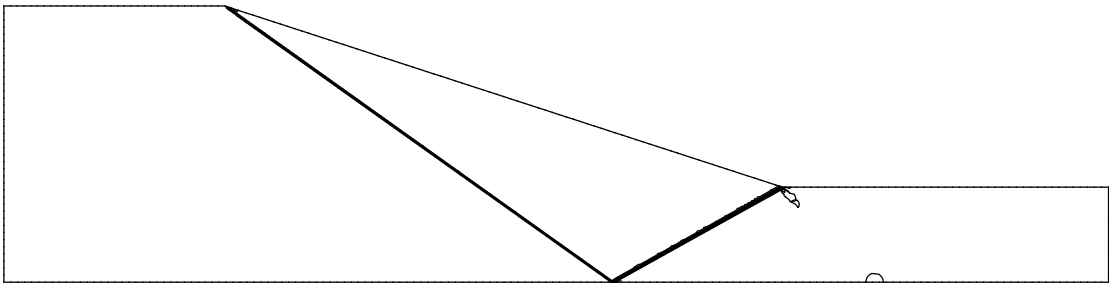
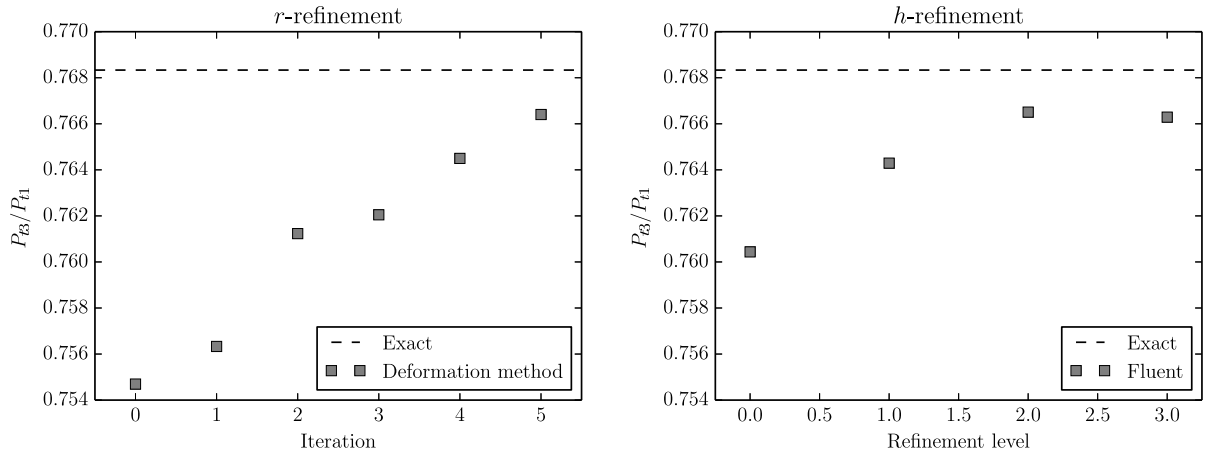


**Figure 10. Final pressure contours from Fluent.**

## C. Discussion

Fig. 11 shows a comparison between total pressure ratios on refined grids from Fluent®and from the deformation method. Both methods approach the exact total pressure ratio from gasdynamics theory. The results from the Fluent®refinement approach the exact solution in fewer iterations, but this is accomplished by inserting more points. The deformation method merely redistributes the existing points. The number of points




(a) Deformation method total pressure ratio.

(b) Fluent total pressure ratio.

**Figure 11. Comparison between deformation method and Fluent total pressure ratio convergence.**

added is shown in Table 1. The Fluent®refinement increases the total number of nodes by approximately 48 percent.

**Table 1. Number of nodes added by Fluent.**

| Refinement level | Number of nodes |
|---|---|
| Initial | 6,400 |
| Iteration 1 | 7,398 |
| Iteration 2 | 9,504 |
| Iteration 3 | 9,511 |

Another important benefit of the deformation method is the data structure. Inserting more points into a mesh requires a complex data structure. The deformation method maintains the structured format which allows for faster processing. As a preprocessing step, Fluent®converts all structured grids to unstructured before operating on them. To highlight this difference, the wall-clock times of CFL3D, and Fluent®were recorded for comparison. Both codes, in serial, were run on the initial grid, for 1500 iterations (on the same machine with an Intel i5 CPU). Fluent took 58.593 seconds to solve the problem. The structured grid solver, CFL3D, took 10.039 seconds to solve the problem, approximately six times faster than Fluent®. This increase in speed can be partially attributed to the grid format. Structured grids are much more efficient than unstructured grids with regard to processing time, memory and solution accuracy.[17] The rest of the speed up is most likely due to differences between the algorithms and solution methods in the different codes.

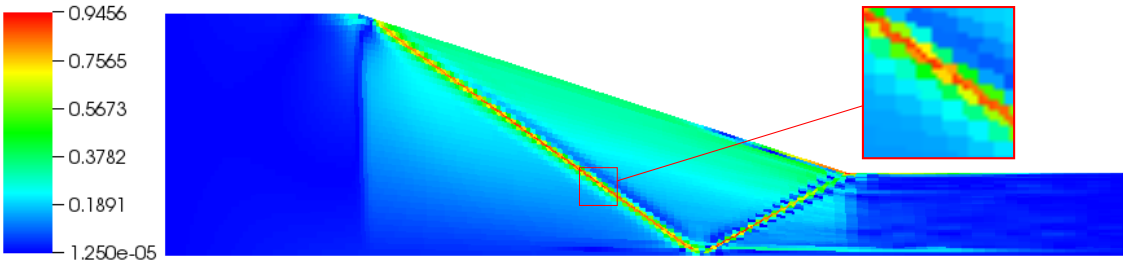



**Figure 12. Contours of skewness on deformed mesh.**

The deformation method avoids complications with the data structure, but it can create highly skewed cells. Figure 12 shows contours of skewness on the deformed grid. A definition of skewness can be found in Ref. 18. Generally, skewed cells should be avoided as they can cause accuracy issues for finite volume schemes.[17] Although the cells in the region of the shock are skewed, the solution on the adapted mesh is still much better than the original mesh as shown by a more accurate total pressure ratio through the shocks and sharper, better resolved shocks. If the skewness was an issue, the methods of Salako are one approach by which the adverse effects of mesh skewness on the flow solver could be mitigated.[19]

## IV. Conclusion

A different, flow-based explanation of the deformation method was provided. The deformation method was compared with a local point-insertion scheme within ANSYS® Fluent®and the results were discussed. The results presented above show the effectiveness of deformation method refinement and of Fluent's point insertion refinement. Fluent's $h$-refinement increases the number of cells in the domain, thereby increasing computation time and complicating the data structure by requiring an unstructured format. This is especially true for unsteady flows because moving features in the flow field require refinement and coarsening. The structured format is desirable because it is more efficient in terms of the data structure. Fluent® (an unstructured solver) and CFL3D (a structured solver) were used to compare computation time. CFL3D was much faster than Fluent®, which can at least be partially attributed to the data structure. The deformation method is a robust approach to $r$-refinement. Unlike the popular physics-based analogies for moving grids, the deformation method guarantees no negative cell volumes. However, the deformation method can create highly skewed cells. Both methods have strengths and weaknesses. In the end, the analyst must select whichever method is best for their particular application.

## Appendices

### A. A direct derivation of equation (6)

We begin with a change of variables from the physical plane, $\mathbf{x} = (x_1, x_2, x_3)$, to the computational plane, $\boldsymbol{\xi} = (\xi_1, \xi_2, \xi_3)$. The left hand side of (3) satisfies the following:

$$\int_{\varphi_t(W)} \frac{1}{f}\, dV(\mathbf{x}) = \frac{d}{dt}\int_W \frac{1}{f} J(\varphi) dV(\boldsymbol{\xi}) = \int_W \frac{d}{dt}(f^{-1}J(\varphi))dV \tag{A.1}$$

$$= \int_W \left[\frac{dJ}{dt} f^{-1} + J\left(\frac{d}{dt} f^{-1}(\varphi, t)\right)\right] dV \tag{A.2}$$

$$= \int_W \left[J\left(\operatorname{div}\left(\frac{\partial \varphi}{\partial t}\right)\right) f^{-1} + J\left((\nabla_\varphi f^{-1}) \cdot \frac{\partial \varphi}{\partial t} + \frac{\partial f^{-1}}{\partial t}\right)\right] dV \tag{A.3}$$

$$= \int_W \left[(\operatorname{div}_\varphi \mathbf{v}) f^{-1} + (\nabla_\varphi f^{-1}) \cdot \mathbf{v} + \frac{\partial f^{-1}}{\partial t}\right] J\, dV \tag{A.4}$$

$$= \int_W \left[\operatorname{div}_\varphi(\mathbf{v} f^{-1}) + \frac{\partial f^{-1}}{\partial t}\right] J\, dV \tag{A.5}$$

$$= \int_{\varphi_t(W)} \left[\operatorname{div}(\mathbf{v} f^{-1}) + \frac{\partial f^{-1}}{\partial t}\right] dV = 0 \tag{A.6}$$

Equation (A.6) is true if the integrand is equal to zero, or

$$\boxed{\operatorname{div}(\mathbf{v} f^{-1}) + \frac{\partial}{\partial t}\left(\frac{1}{f}\right) = 0} \tag{A.7}$$

which is identical to (6).

*Note:* Going from (A.2) to (A.3), we used Abel's lemma which is available in a standard textbook on differential equations and in Hughes and Marsden.[20]

### B. Algorithm diagram

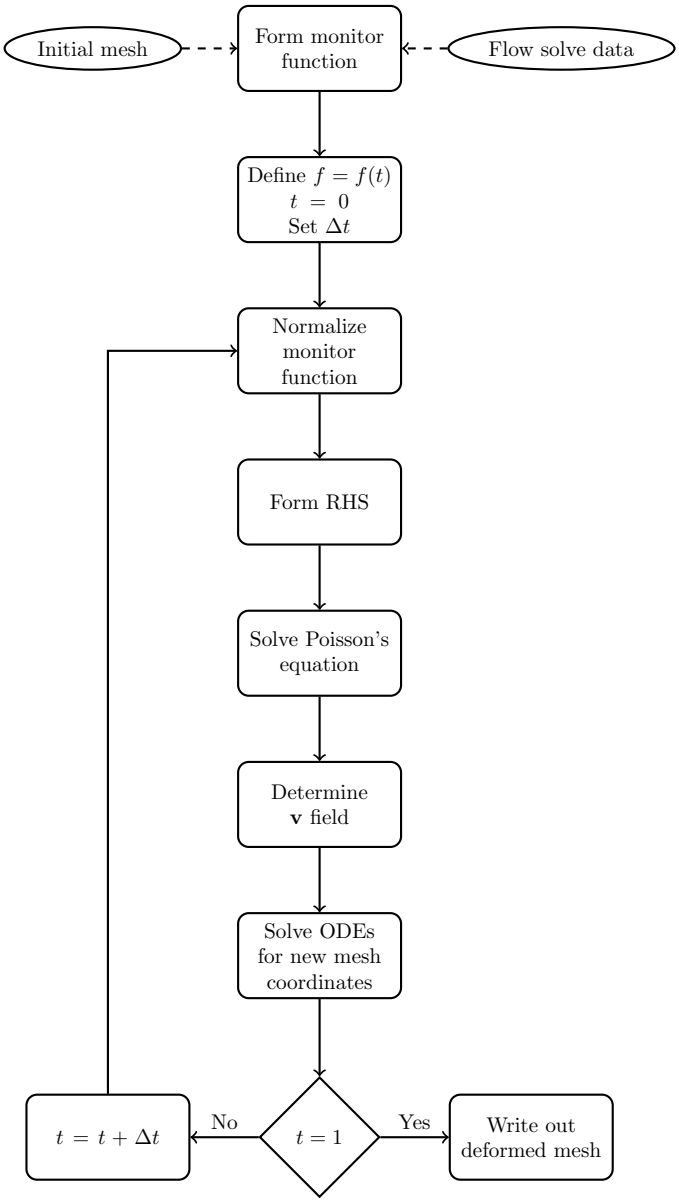



**Figure 13. Algorithm diagram.**

## Acknowledgments

The first author would like to thank the CFL3D team for answering multiple questions about the code and its operation.

# References


[1]Batina, J. T., "Unsteady Euler Airfoil Solutions using Unstructured Dynamic Meshes," AIAA Paper 89–0115, 1989.

[2]Degand, C. and Farhat, C., "A Three-Dimensional Torsional Spring Analogy Method for Unstructured Dynamic Meshes," *Computers & Structures*, Vol. 80, No. 34, 2002, pp. 305–316.

[3]Nielsen, E. J. and Anderson, W. K., "Recent Improvements in Aerodynamic Design Optimization on Unstructured Meshes," *AIAA Journal*, Vol. 40, No. 6, 2002, pp. 1155–1163.

[4]Liao, G. and Anderson, D., "A New Approach to Grid Generation," *Applicable Analysis*, Vol. 44, No. 3–4, 1992, pp. 285–298.

[5]Moser, J., "On the Volume Elements on a Manifold," *Transactions of the American Mathematical Society*, Vol. 120, No. 2, 1965, pp. 286–294.

[6]Semper, B. and Liao, G., "A Moving Grid Finite Element Method using Grid Deformation," *Numerical Methods for Partial Differential Equations*, Vol. 11, No. 6, 1995, pp. 603–615.

[7]Bochev, P., Liao, G., and de la Pena, G., "Analysis and Computation of Adaptive Moving Grids by Deformation," *Numerical Methods for Partial Differential Equations*, Vol. 12, No. 4, 1996, pp. 489–506.

[8]Liu, F., Ji, S., and Liao, G., "An Adaptive Grid Method and its Applications to Steady Euler Flow Calculations," *SIAM Journal of Scientific Computing*, Vol. 20, No. 3, 1998, pp. 811–825.

[9]Liao, G., Lei, Z., and de la Pena, G., "Adaptive Grids for Resolution Enhancement," *Shock Waves*, Vol. 12, No. 2, 2002, pp. 153–156.

[10]Xue, J., *Moving Grids by the Deformation Method*, Ph.D. thesis, University of Texas at Arlington, 2004.

[11]Fleitas, D. L., *The Least-Squares Finite Element Method for Grid Deformation and Meshfree Applications*, Ph.D. thesis, University of Texas at Arlington, 2005.

[12]Boor, C., "Good Approximation by Splines with Variable Knots. II," *Conference on the Numerical Solution of Differential Equations*, edited by G. Watson, Vol. 363 of *Lecture Notes in Mathematics*, Springer, 1974, pp. 12–20.

[13]Zegeling, P. A., "R-refinement for Evolutionary PDEs with Finite Elements or Finite Differences," *Applied Numerical Mathematics*, Vol. 26, No. 1-2, 1998, pp. 97–104.

[14]ANSYS, Inc. Canonsburg, PA 15317, *ANSYS Fluent Theory Guide 14.5*, October 2012.

[15]Biedron, R. T. and Rumsey, C. L., *CFL3D User's Manual (Version 5.0)*, September 1997.

[16]Pletcher, R. H., Tannehill, J. C., and Anderson, D. A., *Computational Fluid Mechanics and Heat Transfer*, CRC Press, 3rd ed., 2013.

[17]Hirsch, C., *Numerical Computation of Internal and External Flows*, Elsevier, second edition ed., 2007.

[18]Jasak, H., *Error Analysis and Estimation for the Finite Volume Method with Applications to Fluid Flows*, Ph.D. thesis, University of London, June 1996.

[19]Salako, S. T., *Optimal Control Approach to Image Registration*, Ph.D. thesis, University of Texas at Arlington, 2009.

[20]Hughes, T. J. R. and Marsden, J. E., *A Short Course in Fluid Mechanics*, Boston: Publish or Perish, 1976.