Spring 5-1-2008

Exploring the Accuracy of Several Grid Clustering Schemes for Computing Forces

Rebecca Rought

Exploring the Accuracy of Several Grid Clustering Schemes for Computing Forces

Rebecca Rought

Candidate for B.S. Degree
in Aerospace Engineering with Honors

05/2008

APPROVED

Thesis Project Advisor: _____
John F. Dannenhoffer III

Honors Reader: _____
Mark N. Glauser

Honors Director:_____
Samuel Gorovitz

Date:_____

Abstract

Forces acting on an airfoil are often found using computational fluid dynamics, and the accuracy of the solution is highly dependent on the type and resolution of the grid used to capture the flow. The simplest way to increase the resolution of a grid is to globally increase the number of grid points. This method results in significantly higher accuracy at the expense of computing time, and therefore is not generally the best solution. Alternatively, the error can be reduced without significantly increasing the computation time if the grid points are moved such that the points cluster around areas of high errors. Previous efforts have used gradients of properties such as pressure or density as a surrogate for error. These techniques have limited success since the errors in the force calculation are not always linked to high solution gradients; the grid points that are closest to the airfoil have the largest impact on the accuracy of the force calculations, while the flow behind an airfoil does not affect the lift and drag on the airfoil. A previously proposed solution to this problem came in the form of using adjoint flow equations, which provide a direct link between the computed force coefficients and truncation errors in the flow field. This method has been developed theoretically, but has only been verified computationally using an unstructured grid. The focus of this paper is to implement the adjoint adaptation for structured grids and to examine the error in force calculations for several different types of grids. A newly-written program which applies an elliptic grid smoother to an algebraic grid was used to generate the grids. Once this elliptic grid was created, it was read into a separate flow solver to find the coefficients of lift and drag of the airfoil. A double wedge airfoil in a supersonic stream was selected to test the program because an analytical solution for lift and drag could be used for comparison.

Table of Contents

List of Figures

List of Tables

Acknowledgments

I would like to thank my thesis advisor Dr. John F. Dannenhoffer III for all of his support and encouragement. Thank you for helping me learn not only about computational grid manipulation, but how to write a thesis and conduct a research project. I feel much more confident about going to graduate school because of the help you gave me this year.

I would also like to thank Dr. Mark Glauser for agreeing to be my reader.

Thanks to the Renée Crown Honors Department for giving me the opportunity to participate in an undergraduate research project, as well as for the financial support.

Finally I would like to thank my parents for their financial and emotional support throughout the last four years. If it weren't for this help, I would never have made it to college, never less graduation.

1 Background

The type of grid that is used in a flow solver has a significant effect on the accuracy of solutions found using computational fluid dynamics (CFD). Different grids have varying levels of success in capturing large changes in a flow characteristic, such as pressure or density. These high-gradient features tend to cause larger errors in the flow solution than do relatively uniform flow areas. In order to reduce the error in these parts of the flow field, the grid needs to be refined to better capture the gradient. One way to refine the grid is to perform a global grid refinement by increasing the number of grid points. The major drawback to the method is that more grid points cause the flow solver to require more CPU and memory usage to find a solution. Since increasing the number of grid points can be time prohibitive, it is common to refine the grid locally instead of globally. One method is to arbitrarily select an area to refine herein termed "local clustering". This method is useful if an area of particularly high gradient in the flow is known a priori. Another method of grid refinement is to look at the features of the flow as they emerge and use them to cluster the grid accordingly. These methods are not as effective as they could be because they look at the entire flow, and not the part of the flow which has the greatest impact on the purpose of the calculation, namely to predict the forces. The coefficients of lift and drag are dependent on the pressures acting on the surface of the airfoil. Therefore, the flow which is behind the airfoil has little or no affect on the force calculations. Yet existing grid refinement techniques that are driven by density or pressure gradients refine the flow behind the airfoil in addition to the flow over the airfoil.

One proposed solution to this problem was to use a set of adjoint Euler equations. Giles provided a theoretical framework, and it has been verified computationally using an unstructured grid[1]. The purpose of this paper is to implement the adjoint adaptation for structured grids and to compare the error in force calculations for several different types of grids.

2 Technical Approach

The first step towards finding a computational solution is to create a basic algebraic grid, shown in Fig. 1 for a double-wedge airfoil in a supersonic free-stream:

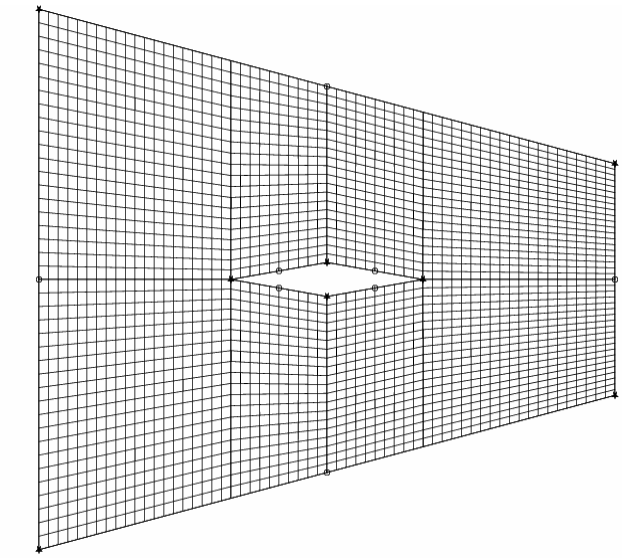


Figure 1: The Algebraic Grid

The algebraic grid is defined by 8 separate blocks which form the grid. The corners of these blocks are specified and are used to create boundary lines for each of the blocks. The boundaries are then evenly divided, giving a set of grid

points on the boundary. The internal points are created from these points using transfinite interpolation:

$$\begin{aligned} x_{i,j} = {} & (1-\xi)x_{i,j} + \xi x_{i,j} + (1-\eta)x_{i,j} + \eta x_{i,j} \\ & -(1-\xi)(1-\eta)x_{i,j} - \xi(1-\eta)x_{I,1} - \eta(1-\xi)x_{1,J} - \xi\eta x_{I,J} \end{aligned} \tag{1}$$

where ξ and η are given as:

$$\xi = \frac{i-1}{I-1} \qquad \text{i = 1,…, I} \tag{2}$$

$$\eta = \frac{j-1}{J-1} \qquad \text{j = 1, …, J} \tag{3}$$

i and *j* represent the index of a grid point within each block, and I and J represent the maximum values of i and j respectively. On the ξ = constant boundaries, *i* is equal to 1 or I, and on the η = constant boundaries, *j* is equal to 1 or J.

The initial algebraic grid is generally not smooth, so two Poisson equations are used to smooth the grid:

$$\xi_{xx} + \xi_{yy} = P \tag{4}$$

$$\eta_{xx} + \eta_{yy} = Q \tag{5}$$

In theory, these equations can be solved using Dirichlet boundary conditions on ξ and η and with the prescribed forcing functions P and Q. However, this requires a computational grid, which we are trying to generate.

Thompson showed that when these equations are inverted[2], they can be used to solve for new grid point locations directly. Thompson's inverted equations are:

$$\alpha\left(x_{\xi\xi} + Px_{\xi}\right) - 2\beta x_{\xi\eta} + \gamma\left(x_{\eta\eta} + Qx_{\eta}\right) = 0 \tag{6}$$

$$\alpha\left(y_{\xi\xi} + Py_{\xi}\right) - 2\beta y_{\xi\eta} + \gamma\left(y_{\eta\eta} + Qy_{\eta}\right) = 0 \tag{7}$$

where α, β, and γ are defined as

$$\alpha = x_\eta^2 + y_\eta^2 \tag{8}$$

$$\gamma = x_\xi^2 + y_\xi^2 \tag{9}$$

$$\beta = x_\xi x_\eta + y_\xi y_\eta \tag{10}$$

In order to create a grid, the discrete representations of the partial derivatives x_η, x_ξ, y_η, and y_ξ are defined using central differencing:

$$x_\xi = \frac{x_{i+1,j} - x_{i-1,j}}{2} \qquad y_\xi = \frac{y_{i+1,j} - y_{i-1,j}}{2} \tag{11}$$

$$x_\eta = \frac{x_{i,j+1} - x_{i,j-1}}{2} \qquad y_\eta = \frac{y_{i,j+1} - y_{i,j-1}}{2} \tag{12}$$

The elliptic grid is generated using a point-Jacobi iterative scheme, using the algebraic grid as the initial solution. To do this, (6) and (7) can be rearranged to find the change in the initial grid point's location:

$$\Delta x_{i,j} = \frac{1}{2\alpha + 2\gamma}\left(\begin{array}{l} \alpha\left(x_{i+1,j} - 2x_{i,j} + x_{i-1,j}\right) \\ \quad - 2\beta\left(\frac{x_{i+1,j+1} - x_{i-1,j+1}}{4} - \frac{x_{i+1,j-1} - x_{i-1,j-1}}{4}\right) \\ \quad + P\alpha\left(\frac{x_{i+1,j} - x_{i-1,j}}{2}\right) + Q\gamma\left(\frac{x_{i,j+1} - x_{i,j-1}}{2}\right) \\ \quad + \gamma\left(x_{i,j+1} - 2x_{i,j} + x_{i,j-1}\right) \end{array}\right) \tag{13}$$

$$\Delta y_{i,j} = \frac{1}{2\alpha + 2\gamma}\left(\begin{array}{l} \alpha\left(y_{i+1,j} - 2y_{i,j} + y_{i-1,j}\right) \\ \quad - 2\beta\left(\frac{y_{i+1,j+1} - y_{i-1,j+1}}{4} - \frac{y_{i+1,j-1} - y_{i-1,j-1}}{4}\right) \\ \quad + P\alpha\left(\frac{y_{i+1,j} - y_{i-1,j}}{2}\right) + Q\gamma\left(\frac{y_{i,j+1} - y_{i,j-1}}{2}\right) \\ \quad + \gamma\left(y_{i,j+1} - 2y_{i,j} + y_{i,j-1}\right) \end{array}\right) \tag{14}$$

Once all of the changes have been computed, the updated grid points are found by:

$$\left(x_{i,j}\right)_{new} = \left(x_{i,j}\right)_{old} + \omega \Delta x_{i,j} \tag{15}$$

$$\left(y_{i,j}\right)_{new} = \left(y_{i,j}\right)_{old} + \omega \Delta y_{i,j} \tag{16}$$

where ω is the under-relaxation factor that has been set to an arbitrary value of 0.75. This value is small enough to reduce the possibility of the grid diverging, but is high enough that it does not drastically increase the number of iterations required for convergence of the grid solver.

In Eq (13) and (14), the forcing functions P and Q can be calculated in a variety of ways. For example, several authors have used P and Q to generate grids with prescribed off-boundary spacing and angles. Here, P and Q will be used to perform the grid adaptation. Following the method developed by Eisemann[3], P and Q can be used to equi-distribute a field variable w by:

$$P = \frac{w_\xi - \left(\beta / \alpha\right) w_\eta}{w} \tag{17}$$

$$Q = \frac{-\left(\beta / \gamma\right) w_\xi + w_\eta}{w} \tag{18}$$

The definition of the equi-distribution function w is described below. Once the forcing functions are found, the elliptic grid generator calculates the new values of x and y for each grid point. After finding a new x and y for each grid point, the local values of w have to be updated to account for changes in the x and y values of a grid points. This is accomplished using:

$$\Delta w = \frac{\partial w}{\partial x}\Delta x + \frac{\partial w}{\partial y}\Delta y \tag{19}$$

where $\frac{\partial w}{\partial x}$ and $\frac{\partial w}{\partial y}$ are calculated by:

$$\frac{\partial w}{\partial x} = \frac{w_\xi y_\eta - w_\eta y_\xi}{x_\xi y_\eta - x_\eta y_\xi} \tag{20}$$

$$\frac{\partial w}{\partial y} = \frac{x_\xi w_\eta - x_\eta w_\xi}{x_\xi y_\eta - x_\eta y_\xi} \tag{21}$$

Here w_ξ and w_η are computed in a similar manner to Eqs (11) and (12). Equations (11)-(21) only pertain to the internal grid points of the grid. For points on a boundary, the grid points are only moved along the ξ or η directions, as opposed to the internal points which are moved in both directions. This modification allows the points to slide along the boundary, without changing the shape and location of the boundary line. The points are moved by treating the boundary as a one-dimensional arc. Modified forms of Thompson's inverted equations, Eq (6) and (7) are used to find the change:

$$s_{\xi\xi} + Ps_\xi = 0 \tag{21}$$

$$s_{\eta\eta} + Qs_\eta = 0 \tag{22}$$

where P and Q are defined as:

$$P = \frac{w_\xi}{w} \tag{23}$$

$$Q = \frac{w_\eta}{w} \tag{24}$$

In Eq (23) and (24), w_ξ and w_η are defined as before. Eq. (21) and (22) can be rearranged to solve for the grid point's movement:

$$\Delta s_{i,j} = \frac{\sqrt{\left(x_{i+1,j} - x_{i,j}\right)^2 + \left(y_{i+1,j} - y_{i,j}\right)^2} - \sqrt{\left(x_{i-1,j} - x_{i,j}\right)^2 + \left(y_{i-1,j} - y_{i,j}\right)^2} + Ps_{\xi}}{2} \tag{25}$$

$$\Delta s_{i,j} = \frac{\sqrt{\left(x_{i,j+1} - x_{i,j}\right)^2 + \left(y_{i,j} - y_{i,j+1}\right)^2} - \sqrt{\left(x_{i,j} - x_{i,j-1}\right)^2 + \left(y_{i,j} - y_{i,j-1}\right)^2} + Qs_{\eta}}{2} \tag{26}$$

where s_ξ, s_η are defined as:

$$s_{\xi} = \frac{\sqrt{\left(x_{i+1,j} - x_{i,j}\right)^2 + \left(y_{i+1,j} - y_{i,j}\right)^2} + \sqrt{\left(x_{i-1,j} - x_{i,j}\right)^2 + \left(y_{i-1,j} - y_{i,j}\right)^2}}{2} \tag{27}$$

$$s_{\eta} = \frac{\sqrt{\left(x_{i,j+1} - x_{i,j}\right)^2 + \left(y_{i,j+1} - y_{i,j}\right)^2} - \sqrt{\left(x_{i,j-1} - x_{i,j}\right)^2 + \left(y_{i,j-1} - y_{i,j}\right)^2}}{2} \tag{28}$$

For the ξ = constant boundaries only (22), (24), (26), and (28) are used, and for the η = constant boundaries, only (21), (23), (25) and (27) are used. The Δs value can be used to update the Cartesian coordinates of the grid point:

$$\Delta x_{i,j} = \frac{\Delta s_{i,j} x_{\xi}}{s_{\xi}} \text{ or } \Delta x_{i,j} = \frac{\Delta s_{i,j} x_{\eta}}{s_{\eta}} \tag{29}$$

$$\Delta y_{i,j} = \frac{\Delta s_{i,j} y_{\xi}}{s_{\xi}} \text{ or } \Delta y_{i,j} = \frac{\Delta s_{i,j} y_{\eta}}{s_{\eta}} \tag{30}$$

The w value can be updated at the grid point by:

$$\Delta w_{i,j} = \frac{\Delta s_{i,j} w_{\xi}}{w_{\xi}} \text{ or } \Delta w_{i,j} = \frac{\Delta s_{i,j} w_{\eta}}{w_{\eta}} \tag{31}$$

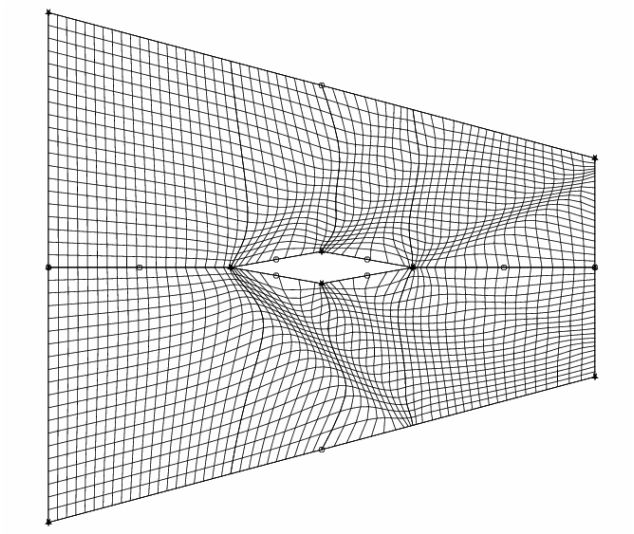
After each iteration, the program checks to see if the grid had converged. The convergence is found by looking at the movement of each grid point:

$$\Delta s = \sqrt{\Delta x^2 + \Delta y^2} \tag{32}$$

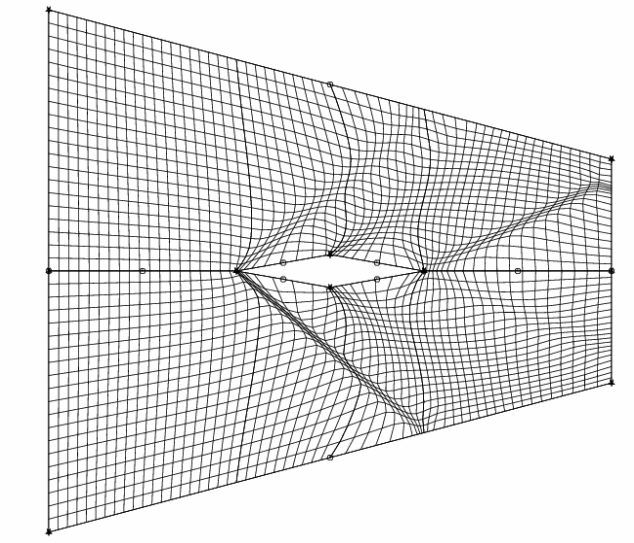
where Δx and Δy are the change in position of the grid point. If the maximum value of Δs over the whole domain is less than the selected convergence value of 10^{-4}, then the grid is determined to be converged. Once the elliptic grid creator has converged, it is read into the flow solver to calculate the lift and drag on the airfoil.

The flow solver that is used to calculate the lift and the drag on the airfoil was developed by Dr. John F. Dannenhoffer III[4]. It approximates the Euler equations via a Lax-Wendroff scheme with added first-order dissipation at the shocks. It also computes the adjoint solution via backward differentiation of the basic scheme.

Once the flow solver finds a solution, it outputs the density and adjoint values at each grid point. These values are used to find new values of *w*, which can be used to modify the grid. The new grid is then used in the flow solver. These steps are repeated several times to ensure a highly accurate flow solution. Fig. 2 shows the creation of an elliptic grid using the values of the density gradient for the refinement parameter. The initial algebraic grid is shown in Fig. 1.



(a)



(b)

Figure 2: Creation of an Adaptive Grid Using Density Gradient

Fig. 2a shows the grid after the first adaptation, while Fig. 2b shows the grid after the final adaptation. The grid becomes increasingly clustered around the oblique shock waves in the flow because these features experience a high change in density.

Sometimes the grid can be modified in a simpler way by clustering the grid points in regions specified by the user. An example of this type of grid clustering uses hyperbolic tangent stretching. This type of clustering moves the grid points on the boundary closer to a specified point, as shown in Fig. 3:

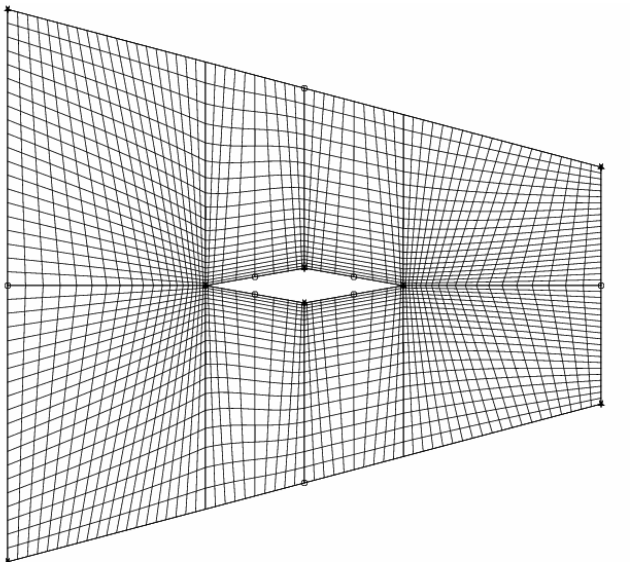


Figure 3: Clustering by Hyperbolic Stretching

The stretching in Fig. 3 was applied at the leading edge, the trailing edge, and the mid point of the airfoil since these are known locations of shock waves and expansion waves. The off-body spacing was controlled in a similar manner.

3 Computed Results

The flow solution was calculated on three different types of grids. The first type of grid was a simple algebraic grid with no adaptation. The second type of grid examined used local clustering at the leading edge, trailing edge, and midpoint of the airfoil. The final type of grid used an adaptive grid smoother with the forcing functions P and Q. The adaptive grid was modified using the density difference, adjoint, and a combination of density and adjoint. The accuracy of the

computational solution for each grid was determined by first calculating the analytical solution.

3.1 Analytical Solution

In order to determine the accuracy of different grid modification techniques, the true forces acting on the airfoil are first found analytically. An airfoil is selected where the forces can be easily calculated using knowledge of compressible flow. The airfoil in the test case is a double wedged airfoil with leading and trailing edge angles of 20°. The airfoil is subjected to a flow with a free-stream Mach number of 2.0 at a 5° angle of attack. To find the coefficients of lift and drag, the pressure acting on the upper and lower surfaces of the airfoil are calculated. The airfoil experiences a shock wave both above and below the leading edge of the airfoil. It also experiences an expansion fan at the corner in the middle of the airfoil. Fig 4 illustrates the shock and expansion waves acting on the airfoil.

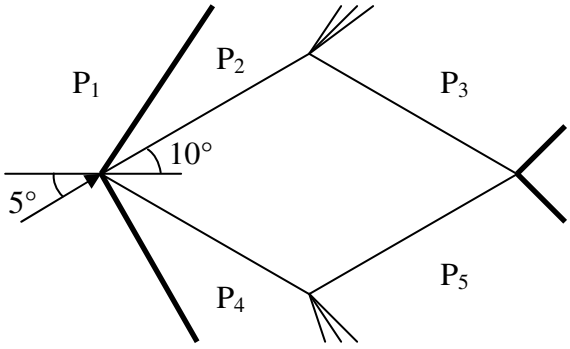


Figure 4: Airfoil Used to Verify Accuracy

The pressures P_2 and P_4 are calculated by finding the Mach numbers behind the leading edge shock using a standard oblique shock reference table. The pressures

P_3 and P_5 can be found using a Prandtl – Meyer expansion wave function table. The values of the pressures are given in Table 1.

Table 1: Pressure Over the Airfoil

Location	Pressure
P2/P1	.179
P3/P1	.235
P4/P1	.0702
P5/P1	.392

The coefficients of the forces in the x and y directions acting on the airfoil are computed by:

$$C_x = \frac{2}{\gamma M_1^2}\left[\left(\frac{P_4 - P_2}{P_1}\right)\cos(\theta) + \left(\frac{P_5 - P_3}{P_1}\right)\cos(\theta)\right] \tag{33}$$

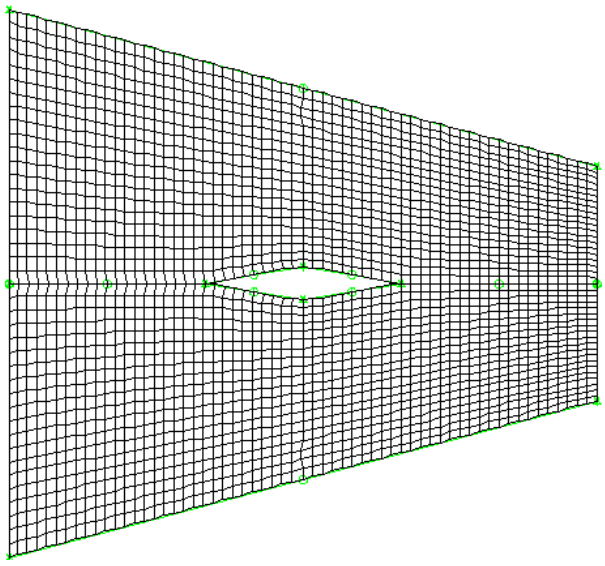
$$C_y = \frac{2}{\gamma M_1^2}\left[\left(\frac{P_4 - P_2}{P_1}\right)\sin(\theta) + \left(\frac{P_5 - P_3}{P_1}\right)\sin(\theta)\right] \tag{34}$$

The angle θ is the angle between the surface of the airfoil and the horizontal, M_1 is the free-stream Mach number, and γ is the specific heat ratio, taken to be 1.4 for this case. In the test case, the value of θ is 10°. Once the coefficients of forces in the x and y directions are found, they are converted into lift and drag coefficients. The coefficient of drag was found to be 0.0938 and the coefficient of lift was found to be 0.215.

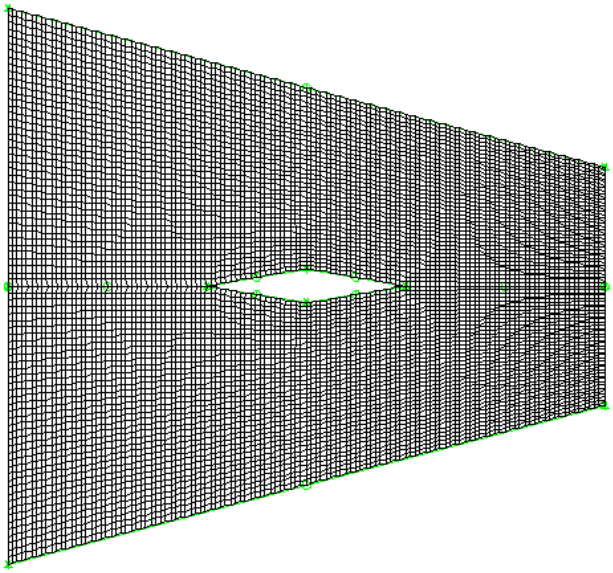
3.2 Global Refinement

The global refinement took the algebraic grid and added more grid points. Two refinements were performed and the error in the lift and drag calculations was examined. After the first refinement, the grid increased from a 61 x 41 grid to

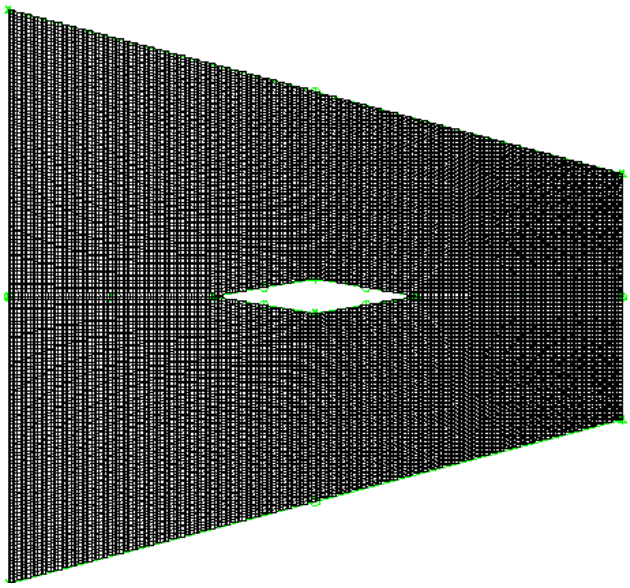
a 121 x 81 grid. The second refinement increased the number of points to 241 x 161 points. Fig. 5 shows the grid after each global refinement:



(a)



(b)



(c)

Figure 5: Global Grid Refinement. (a) 61x41 Grid (b) 121x81 (c) 241x161

Fig. 6 illustrates the error in the coefficient of lift and drag after each refinement:

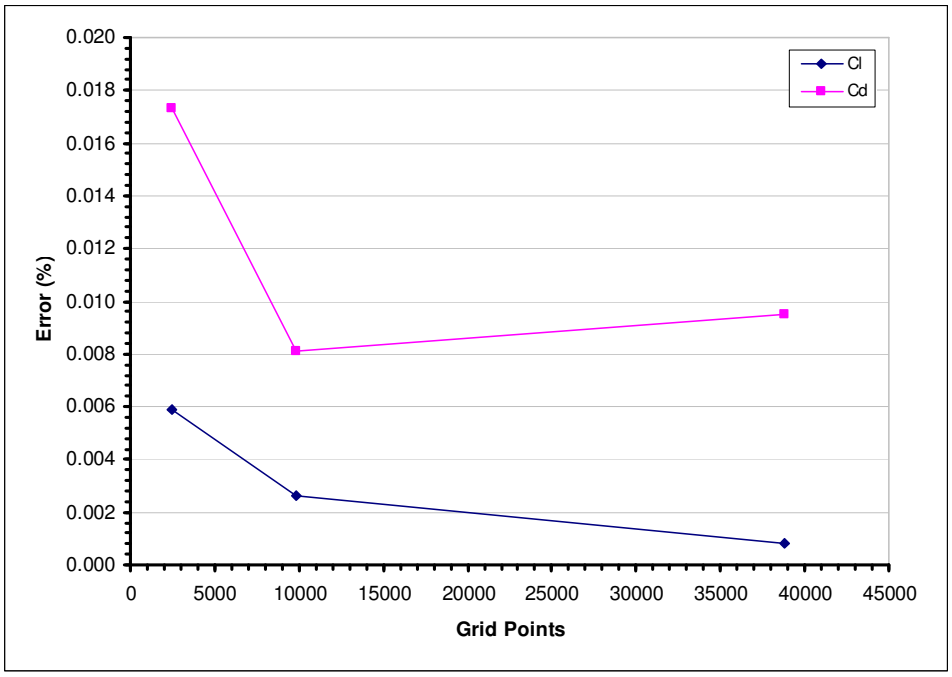


Figure 6: Affect of Global Refinement on the Accuracy

Fig. 6 shows that increasing the number of grid points initially caused a large decrease in the error associated with the calculations. Refining the grid further, however, did not cause significant change in results. This was caused by a combination of the added dissipation and round-off error. Table 2 shows the number of iterations it took the flow solver to converge for the different number of grid points.

Table 2: Iterations until Convergence

Number of points	Iterations
2501	1410
9801	1807
38801	2808

The number of iterations it took for the flow solver to converge increased when the number of grid points increased. Also, each iteration took longer to complete because there were more points on the grid. As a result, increasing the number of points greatly increased the computation time. Fig. 7 shows the relationship between the error in the calculation and the computational time costs, which was calculated by multiplying the number of iterations by the number of points in the grid.

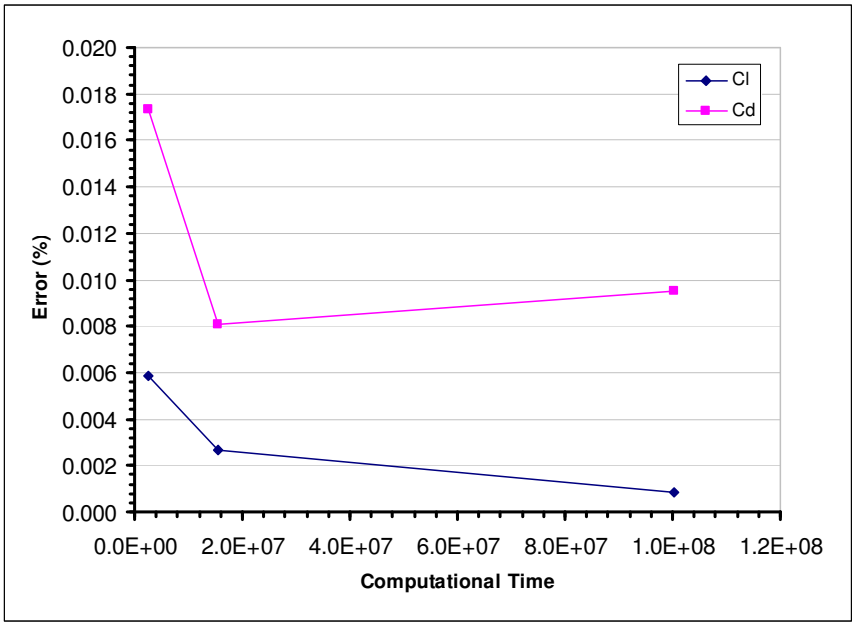
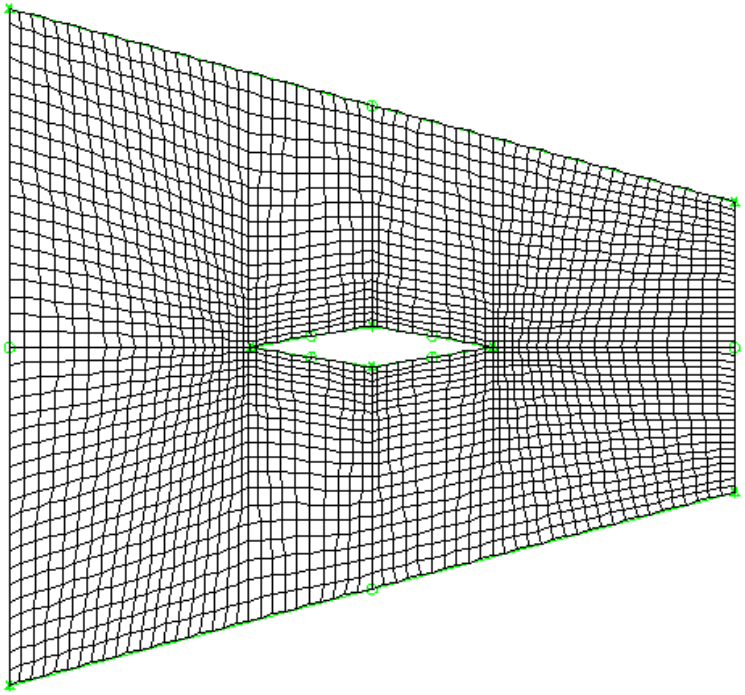


Figure 7: Computational Time for Global Refinement

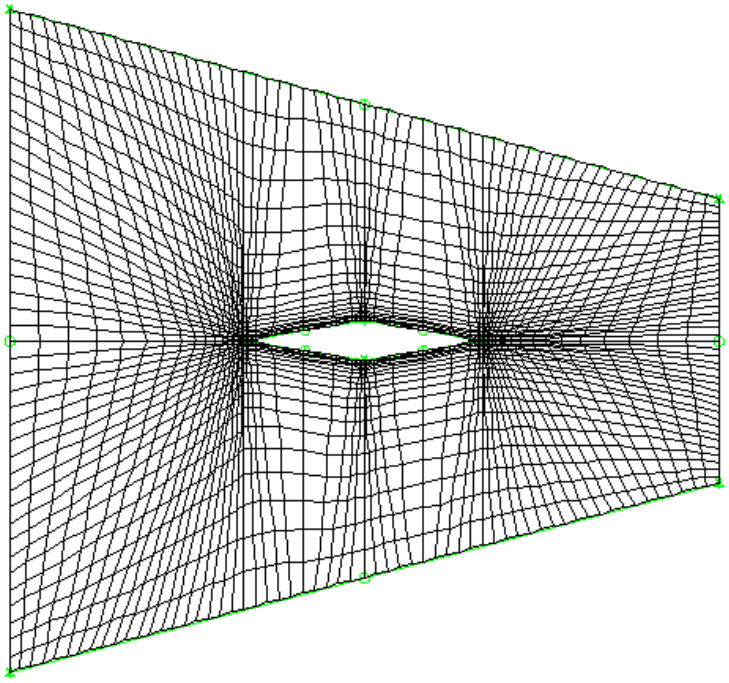
Fig. 7 shows that simply increasing the number of points to increase the accuracy of the solution is not an effective method.

3.3 Local Clustering

Local clustering was examined next by a priori stretching the 61 x 41 grid. Hyperbolic tangent stretching was used at the middle, leading and trailing edges of the airfoil. Fig. 8 shows two grids with different initial spacing.



(a)



(b)

Figure 8: Local Clustering with Initial Spacing Specified: (a) .025 (b) 0.00312

The effects on the accuracy by clustering the grid points near the edges is shown in Fig. 9

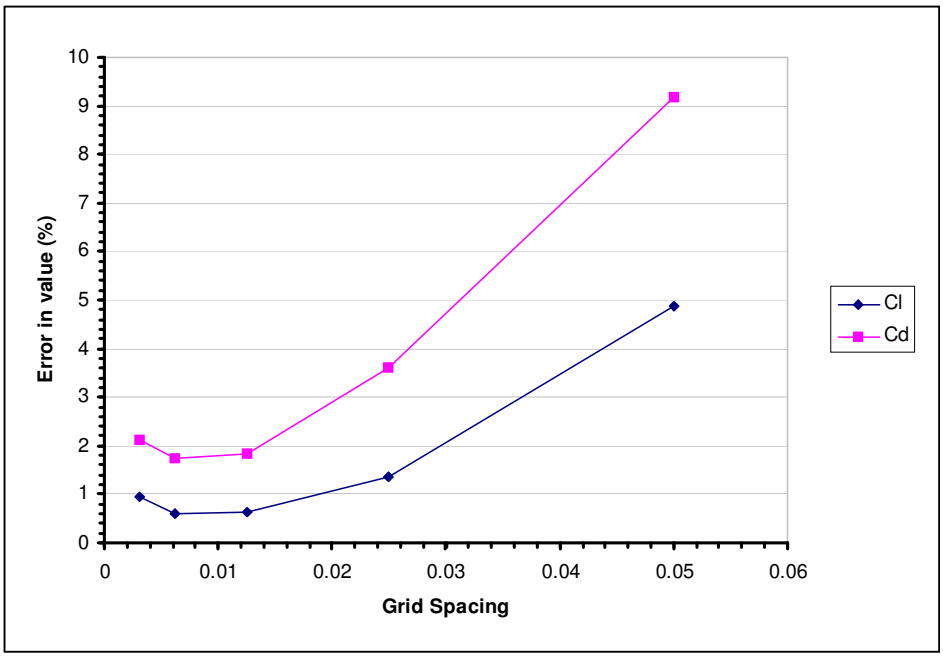


Figure 9: The Accuracy of a Grid Compared to the Spacing

Fig. 9 shows that the accuracy of the grid increases for coefficients of lift and drag as the clustering increases. The results also show that when the grid becomes too highly clustered, it loses accuracy. As the grid is clustered, it decreases the error in the part of the grid where the clustering occurs, but the clustering causes the spacing between other cells to increase, and the error in those areas increases. When clustering becomes too great, the reduction in error caused by the clustering is not enough to overcome the increase in error of the other grid points. Fig. 8b clearly shows the areas with poor refinement when clustering becomes too high. Unlike the global refinement method, the local clustering does not drastically increase the computational time.

A major drawback to the local clustering method is that it assumes the locations of major flow features are known. Unfortunately for most airfoils, the location of a shock wave is not known a priori, as it can move depending on the Mach number and angle of attack.

3.4 Adaptive Scheme

The adaptive scheme can be performed based on several different flow properties. This study examined a refinement parameter based on density difference, the adjoint flow solution, and the product of the density difference and the adjoint.

3.4.1 Density Difference Redistribution

The adapted grid scheme clustered the grid based on differences in density instead of arbitrarily clustering the grid. The contours of density around the airfoil are shown in Fig. 10:

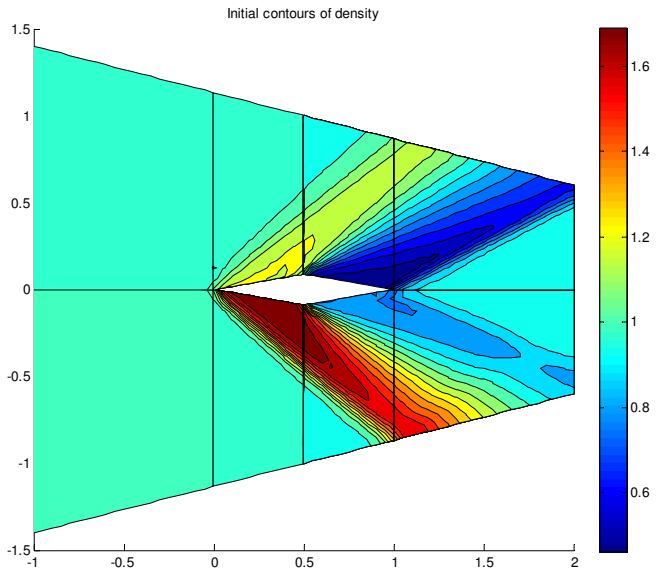


Figure 10: Contours of Density

The contours of density clearly show the shock waves and expansion waves at the locations expected. The strongest shock wave is at the lower leading edge, as evident by the closeness of the contours. From the contours of density, the equi-distribution function w was calculated. The equi-distribution function is based on the refinement parameter, R:

$$w = 1 + \frac{w_{\max} - 1}{R_{\max}} R \tag{35}$$

For the density difference redistribution, R is defined as the difference between the maximum and minimum density at the grid points immediately surrounding the specified location. The contours of w are shown in Fig. 11:

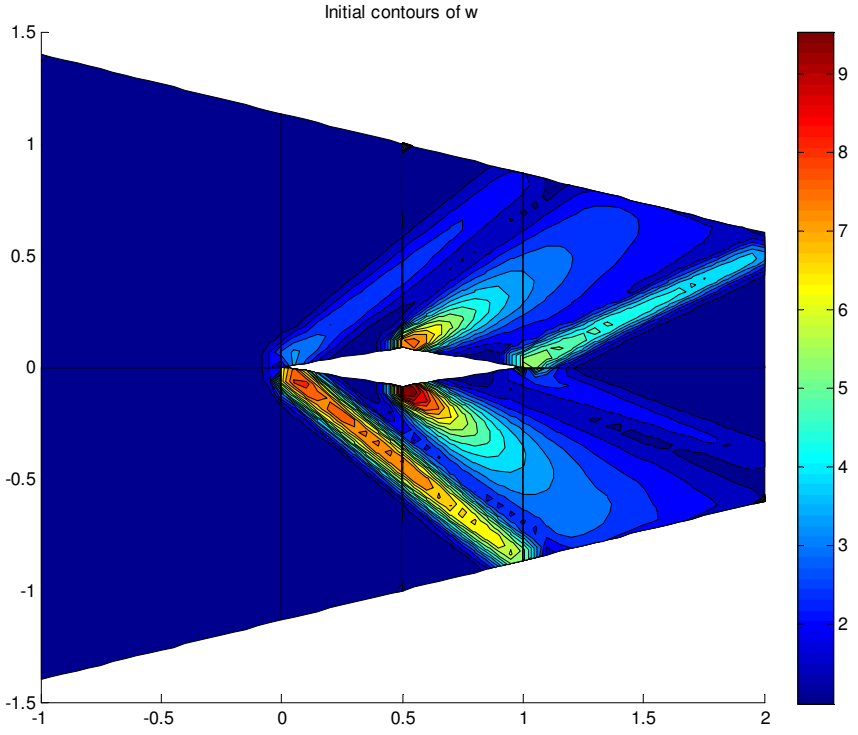
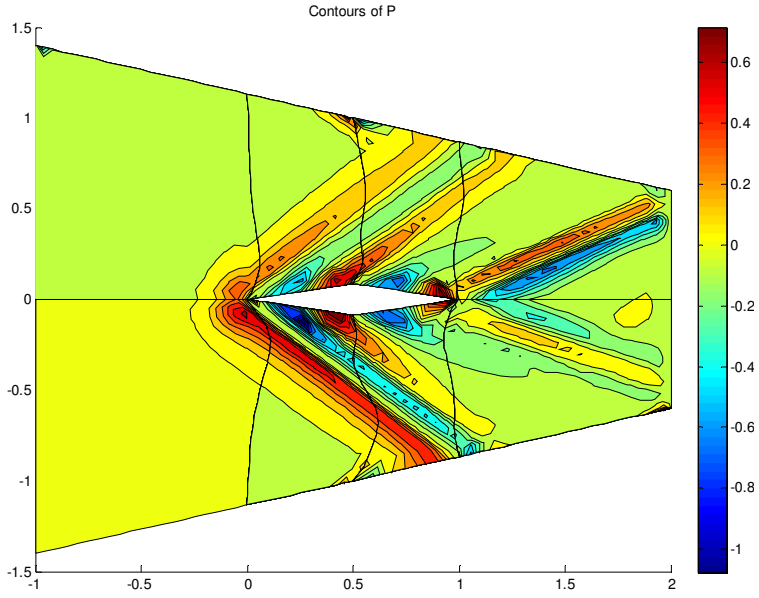


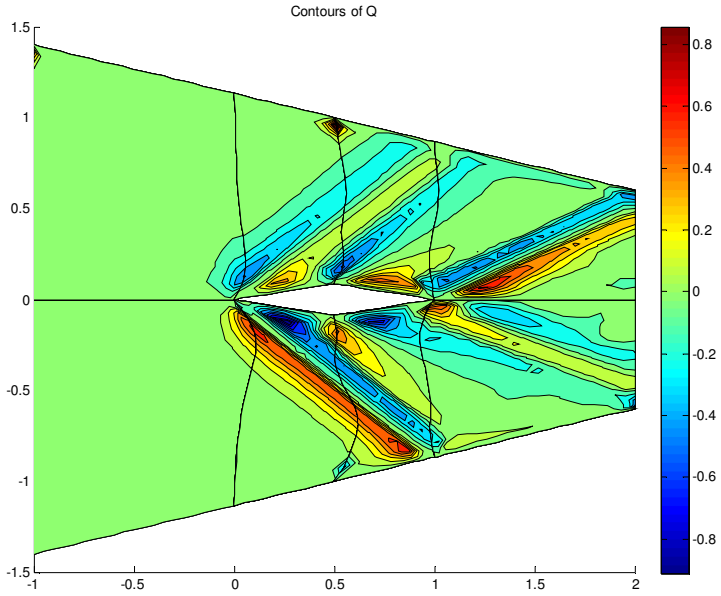
Figure 11: Initial Contours of *w*

The contours of w are highest where the density gradient is the highest, which is along the shock waves and the expansion waves. These high values of w

cause the forcing functions P and Q to be highest at these locations, as shown in Fig. 12:



(a)



(b)

Figure 12: (a) Contours of P (b) Contours of Q

The contours of P and Q are highest where the shock and expansion waves are. It is at these locations where the grid will be clustered. The final grid based on the initial flow solution is shown in Fig. 13:

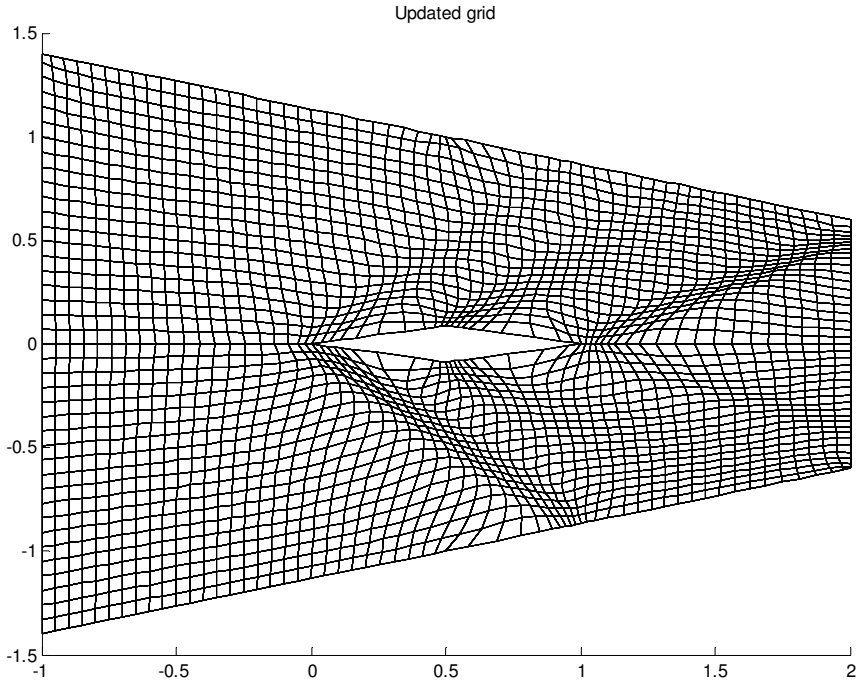


Figure 13: Adapted Grid for First Flow Solution

This grid was used in the flow solver to find a new flow solution. After several adaptations, the grid was modified to:

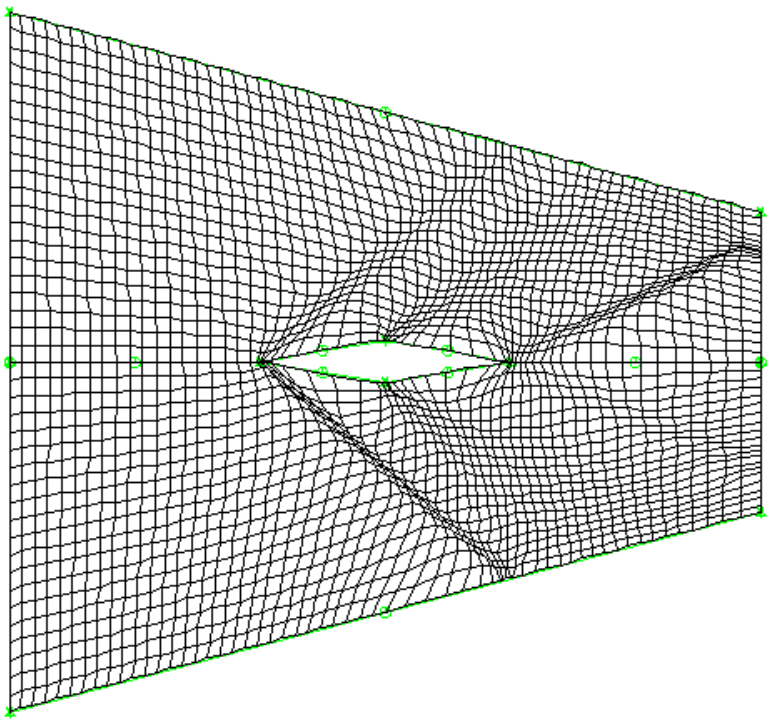


Figure 14: Final Grid after 4 Adaptations

The grid becomes even more clustered around shock and expansion waves after successive adaptations. Fig. 15 shows the error in the adapted grid solution after each adaptation cycle.

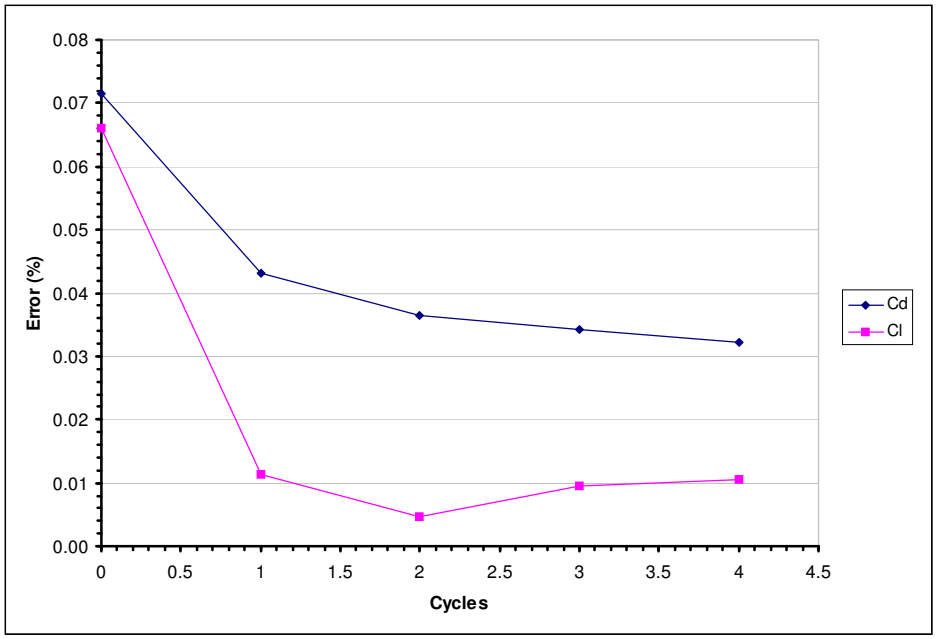


Figure 15: Error after Successive Adaptations Based on Density Distribution

From Fig. 15, it is clear that the adaptive scheme eventually converges to a non-zero error. This occurs because the number of grid points, which is fixed, is insufficient to achieve an error of zero.

The accuracy of the density adaptive scheme is compared to the algebraic and clustering schemes for a 61x 41and is shown in Table 3.

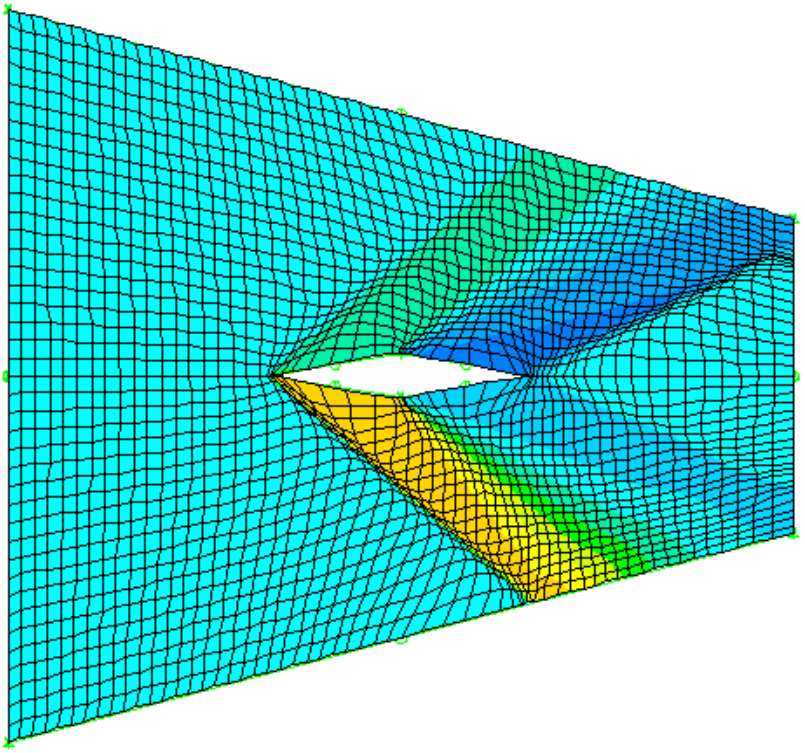
Table 3: Percent Error Associated with Different 61x41 Grids

	C_d	C_l
Algebraic	0.0916	0.0488
Adapted - Density	0.0322	0.0105
Clustering (.00312)	0.0213	0.0095

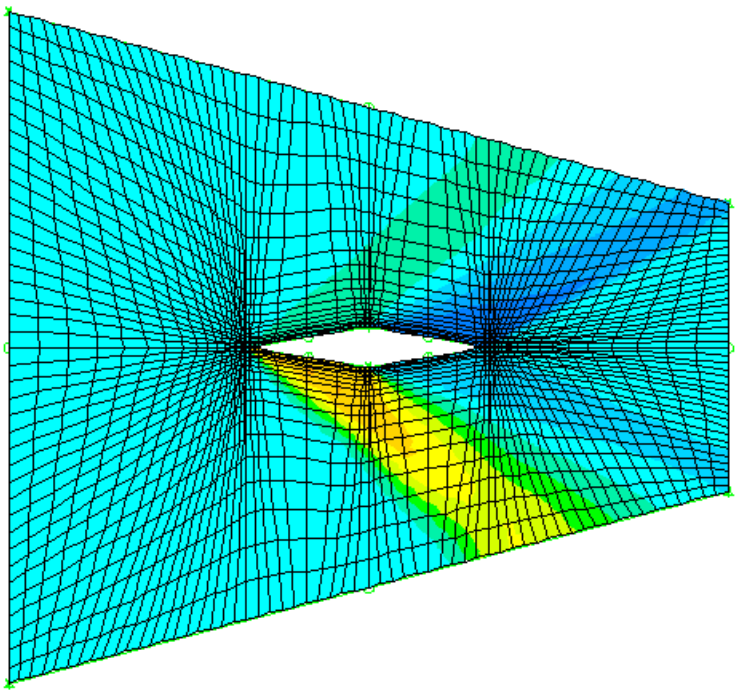
From Table 3, it is clear that both the adaptive and the local clustering schemes reduce the amount of error in the calculation. It appears, however, that local clustering improves the accuracy more than the adaptive scheme. For the wedge airfoil, it is difficult to tell when the optimum configuration has been

reached, and clustering further can increase error as seen in Fig. 9. The density difference adaptive grid method does not have this problem.

The problem of the local clustering technique can be seen by examining the grid with respect to the flow solution. Fig. 16 shows the contours of pressure from the flow solution superimposed on top of the grid for both the adaptive grid and the clustered grid



(a)



(b)

Figure 16: Adapted Grid (a) and Clustered Grid (b) Over Flow Solution

The adaptive grid has clustering which is aligned with the shock waves, refining the grid near the locations with the highest change in density. The grid in Fig. 16b is poorly refined further from the focal point of the clustering. Looking at the pressure gradient of the flow superimposed over the grid, it is evident that some of this poor refinement occurs in high gradient areas. This poor refinement is not reflected in a comparison of the accuracies of the two solutions, however. From Table 3, the clustered grid had less error than the adaptive grid. The reason for this is that the clustered grid moves the grid lines closer to the airfoil. Since the flow closest to the airfoil has the most affect on the calculations of lift and drag, increasing the resolution close to the grid also increase accuracy. The adaptive scheme clusters the grid far from the airfoil as well as behind it. This

clustering does not improve accuracy because the flow in those locations does not affect the lift and drag on the airfoil.

3.4.2 Adjoint Redistribution

Using the adjoint solution as the refinement parameter was examined and compared to the accuracy of an adaptive grid using the density gradient. The adjoint solution was proposed because it ignores the flow behind the airfoil and instead looks at the flow which has the largest effect on the forces acting on the airfoil.

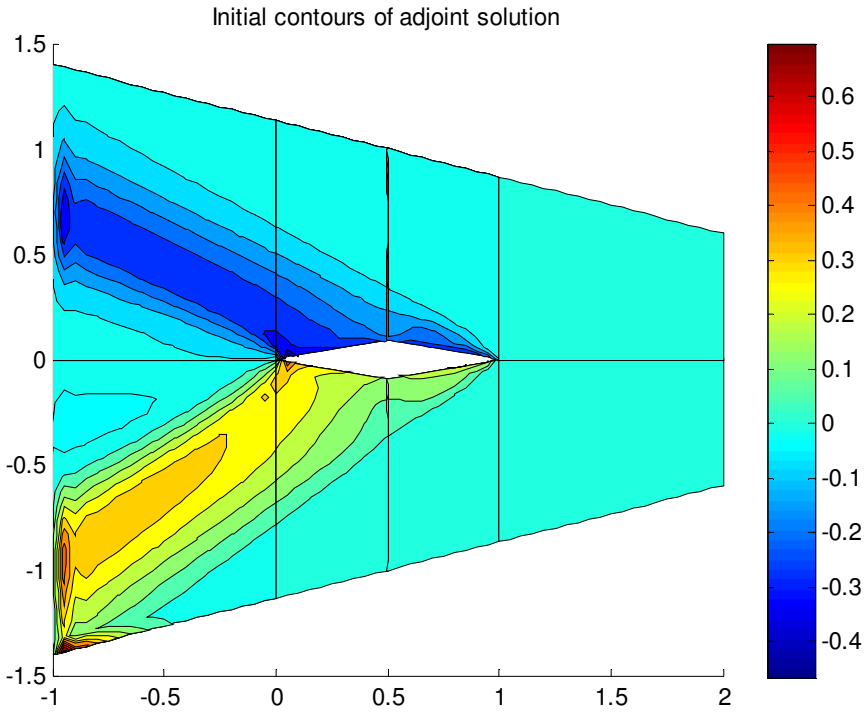


Figure 17: Contours of Adjoint Solution

From Fig. 17 it is apparent that the adjoint solution is highest at the inflow boundary, as well as the leading edge of the airfoil. The inflow boundary is important because it controls the Mach waves that impinge on the airfoil. This can

result in a significant change in the lift and drag on the airfoil. The flow around the leading edge is extremely important as well because the shocks are located around this point. The lower edge shock is stronger, so the adjoint is larger on the bottom of the airfoil than it is on the top. Behind the airfoil, the adjoint is nearly zero, which was expected since the flow behind the airfoil does not affect the forces on the airfoil. The adjoint is also zero far above and below the airfoil. Using the adjoint as the refinement parameter results in the contours of *w* shown in Fig. 18:

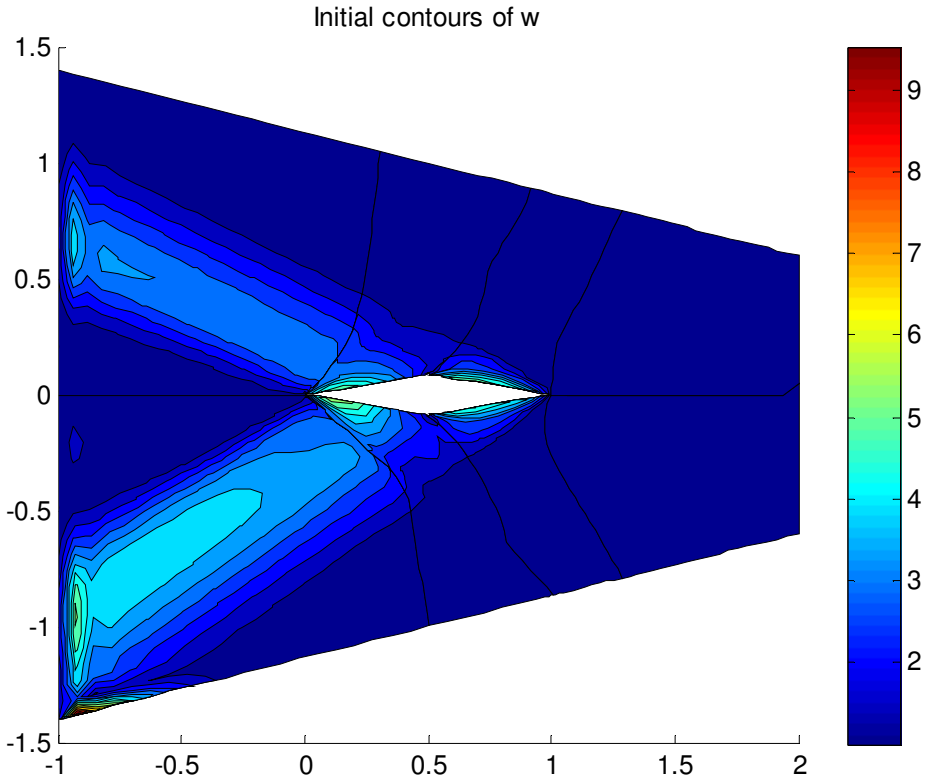
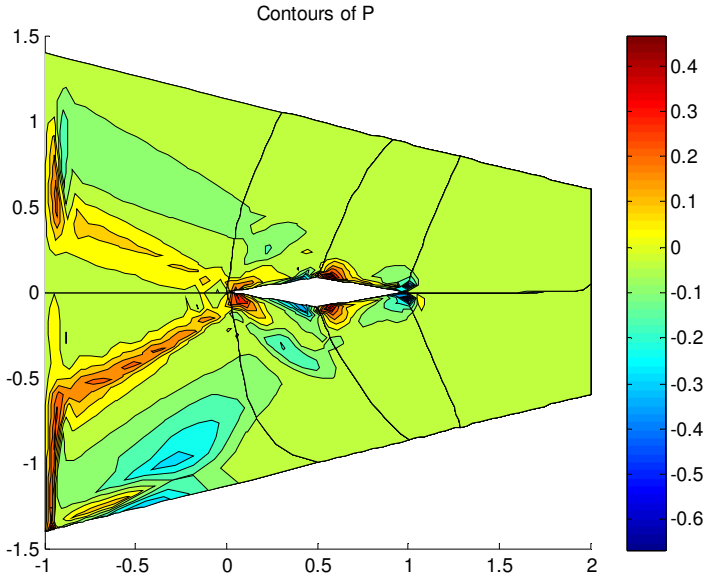
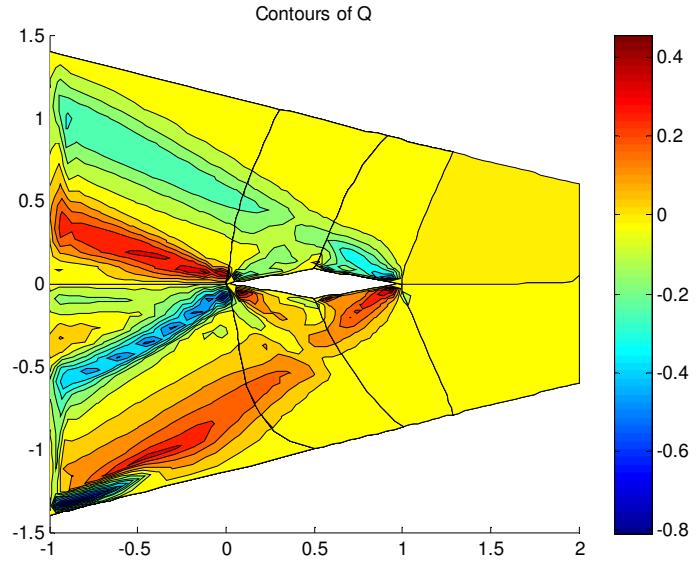


Figure 18: Initial Contours of *w* based on Adjoint

The contours of w show that the P and Q values should be highest close to the airfoil, as well as in the area swept forward from the airfoil. Fig. 19 shows the values of P and Q:



(a)



(b)

Figure 19: (a) Contours of P (b) Contours of Q

The values of P and Q cause the grid to be swept forward and also brought in close to the airfoil, as seen in Fig. 20:

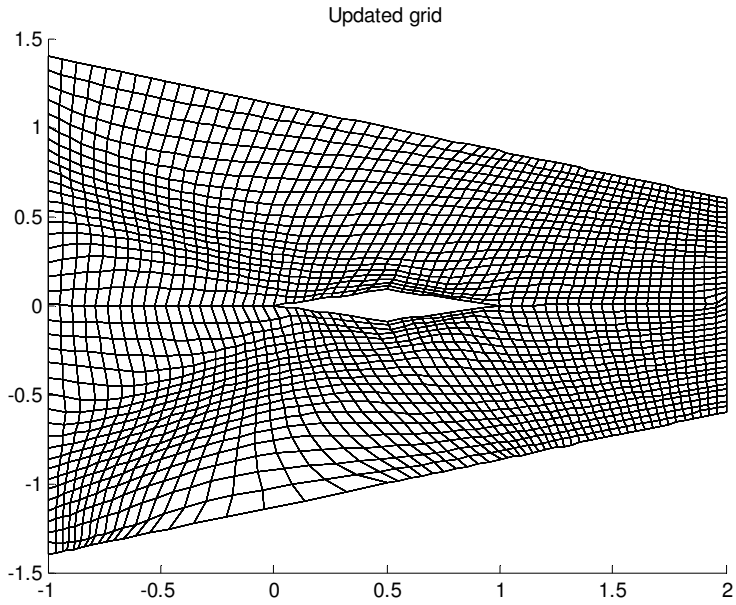


Figure 20: Grid Modified Based On Adjoint Solution

The grid in Fig. 20 is based on the initial flow solution. Like the density adaptive scheme, this grid is used to find a new flow solution. After several adaptations, the final grid is found:

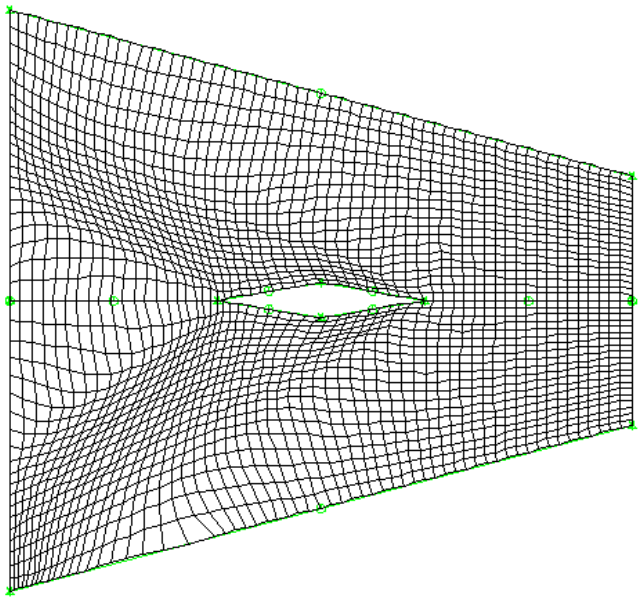


Figure 21: Elliptic Grid Based on Adjoint Solution.

As with the density adaptation, adapting the grid based on the adjoint solution causes the error in the coefficient of lift and drag to decreases after successive adaptations of the grid as shown in Fig. 22:

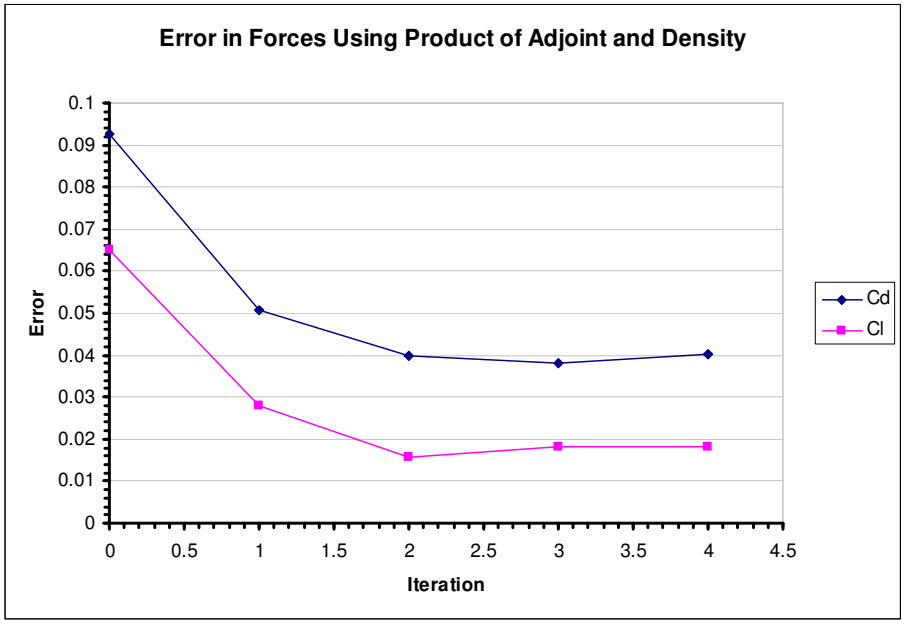


Figure 22: Error in Solution Using Adjoint Adaptation

The error in the lift calculation was compared to the error found using density adaptation:

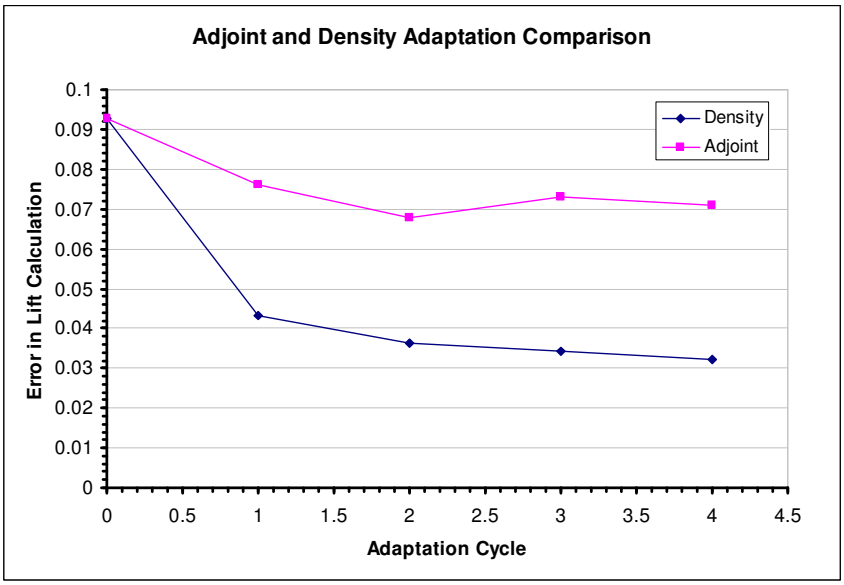


Figure 23: Comparison of Error in C_l of Adjoint and Density Adaptation

Fig. 23 shows that simply using the adjoint solution does not improve the accuracy of the solution as much as the density-difference adaptation. The adjoint results in a less accurate solution because it refines the area far in front of the airfoil. While this flow area has a high effect on the solution, the flow is uniform in this location, so the truncation error in the flow calculation is likely to be very small.

3.4.3 Product Adaptation

Another way to create the refinement parameter is to cluster the grid in the flow area which experiences both high error and has a high impact on the forces acting on the airfoil. To cluster the grid at this location, the product of the density difference and the adjoint solution was found. This value became the new

refinement parameter for product refinement. The initial contours of the equi-distribution function *w* are shown in Fig. 24:

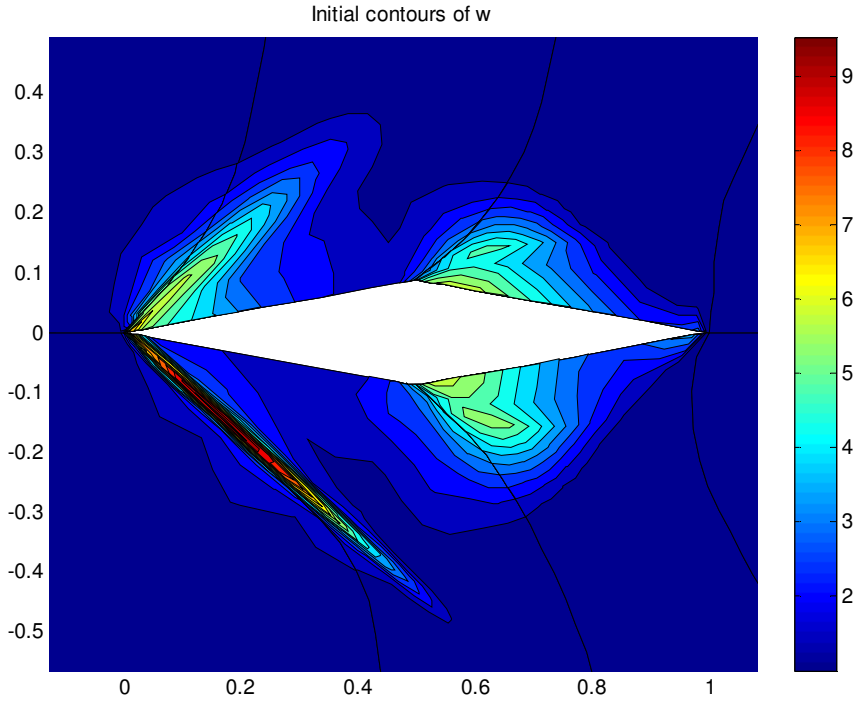
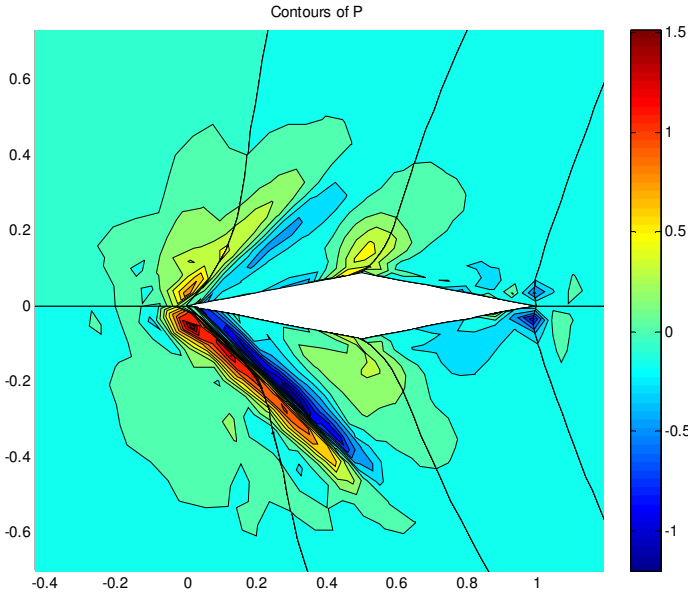
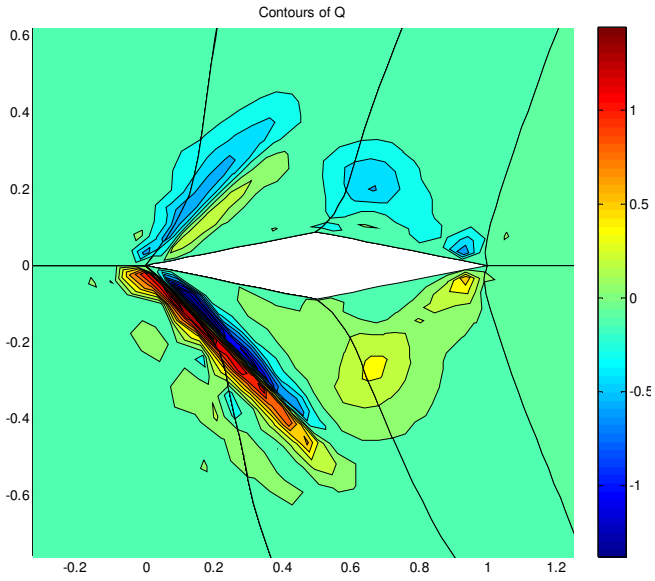


Figure 24: Initial Contours of *w* for Product Adaptation

The values of *w* are highest along the lower edge shock wave, as well as the upper edge shock and both the expansion waves. It is highest here because the shock wave has both a high density gradient and a large affect on the computed forces. Taking the product of the density gradient and the adjoint solution causes *w* to equal zero ahead of the airfoil where the density gradient is zero, and behind the airfoil where the adjoint solution is zero. As a result, *w* is only non-zero at locations close to the airfoil. The values of *P* and *Q* for the product adaptation are shown in Figure 25:



(a)



(b)

Figure 25: Product Adaptation Forcing Functions (a) P (b) Q

Fig. 26 shows the resultant grid when the values of *P* and *Q* are applied to the elliptic grid:

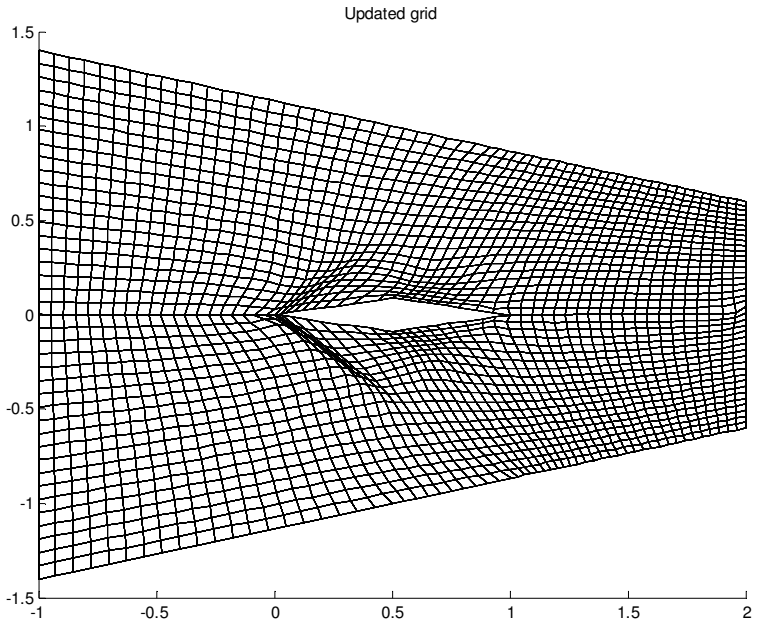


Figure 26: Initial Grid from Product Adaptation

As expected, the grid is only modified near the surface of the airfoil, along the leading edge shocks, and near the expansion wave. The grid is no longer modified near the trailing edge shocks. The grid after several adaptations is shown in Fig. 27:

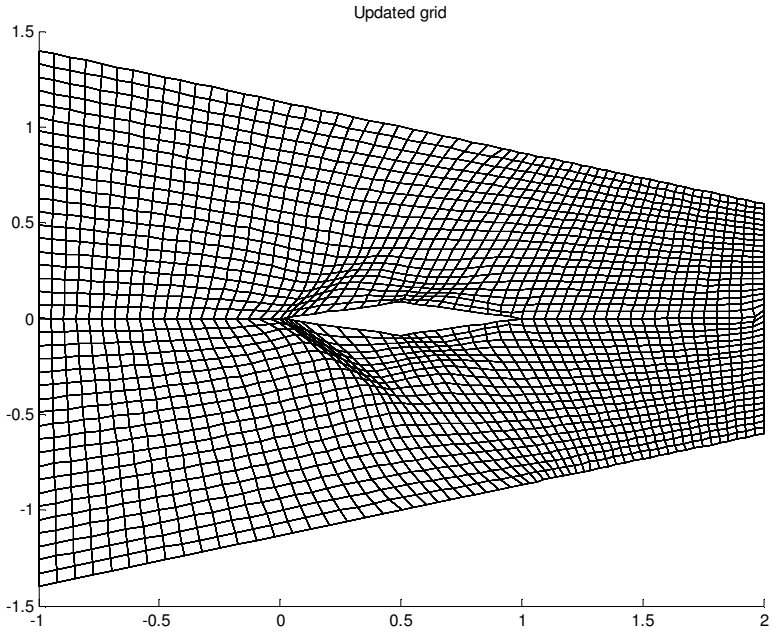


Figure 27: Final Grid after Product Adaptation

The errors in the coefficients of lift and drag after successive adaptations are shown in Figure 28:

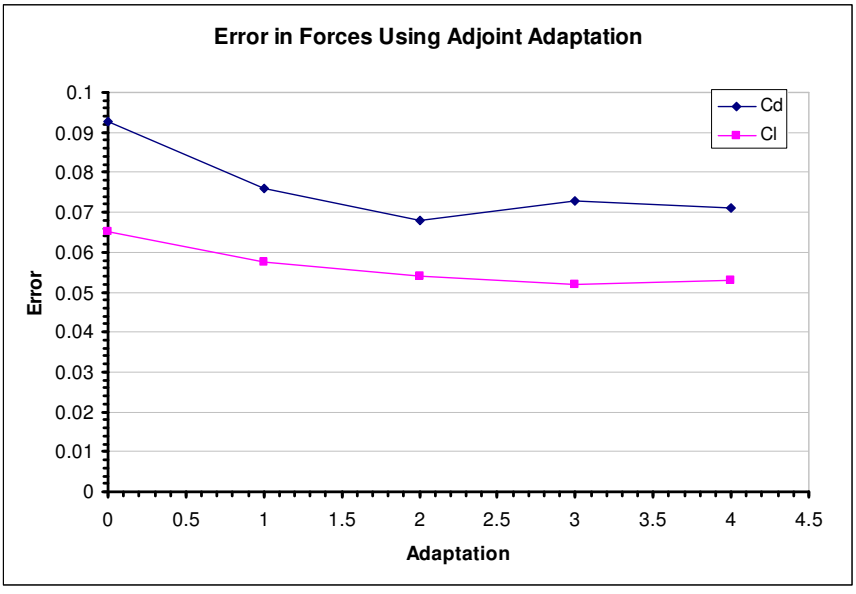


Figure 28: Error in C_l and C_d Using Product Adaptation

The error in the coefficients of lift and drag drops quickly and converges to a finally error value. A comparison with the error in the coefficient of lift of previously mentioned adaptive grids is shown in Fig. 29:

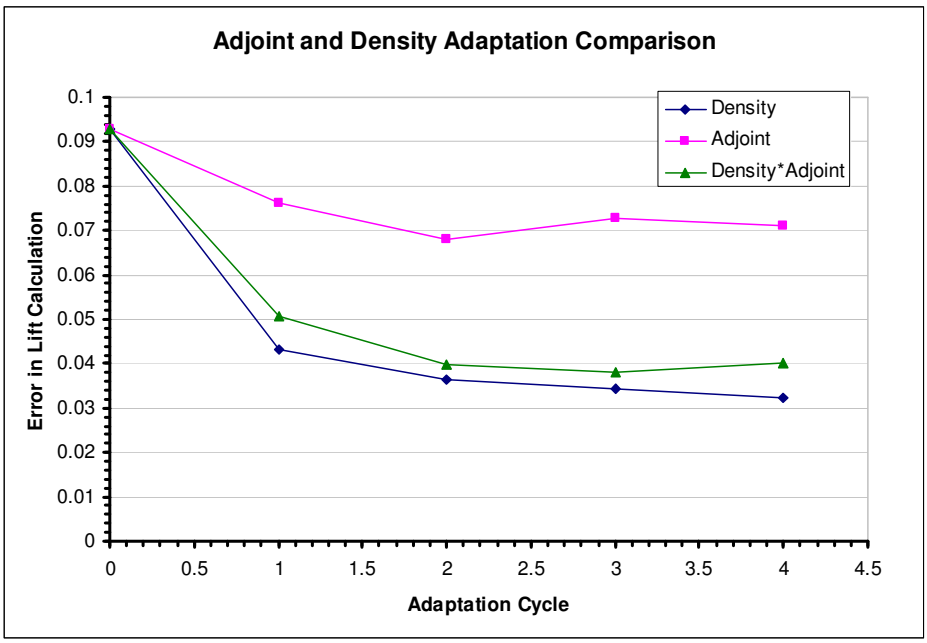
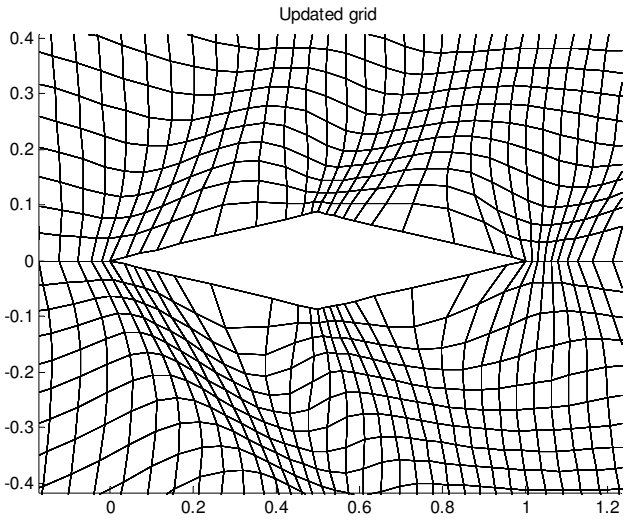
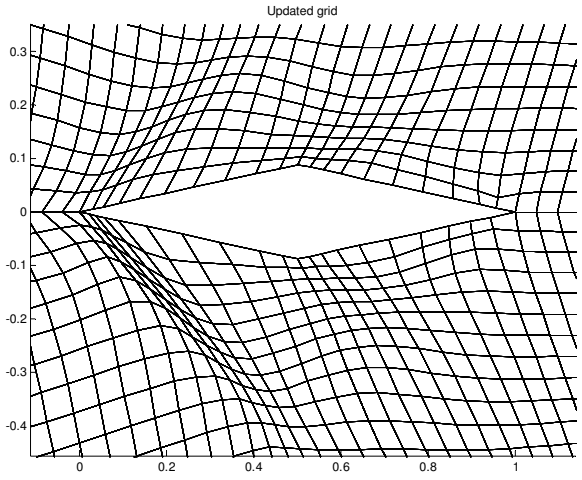


Figure 29: C_l Error Resulting from 3 Different Adaptive Grids

Using the product of the adjoint and the density provided a much more accurate solution than the adjoint alone. The solution is not as accurate as the grid adapted based on density gradient, but the two solutions are very similar in accuracy. The results did not confirm that the adjoint solution improved the accuracy of the solution. This can be explained by the construction of the grid. Since the leading and trailing edge of the airfoil are held constant, the grid behind the airfoil cannot be used to better refine the grid along the surface. As a result, the refinement behind the airfoil does not harm the accuracy of the density adaptive technique. A comparison of the grid near the surface of the airfoil shows the grid to be similar for both the density and product adaptations.



(a)



(b)

Figure 30: Grid around the Airfoil: (a) Density (b) Product

The product adaptation clustered the grid slightly closer to the airfoil than the density, but it did not improve the accuracy. One explanation as to why the product refinement is not as accurate as the density refinement is the type of airfoil used. The wedge airfoil was selected because its analytical solution is known for supersonic flows. It is possible that the case is not representative of actual airfoils such as the NACA0012. For standard airfoil shapes in a transonic

flow, the shock does not occur at the leading edge, but rather in the middle of the airfoil. As a result, the product of the adjoint solution and the density may work better than simply density gradient in that case.

The adaptive technique is better than simple global refinement regardless of which property is used for the refinement parameter.

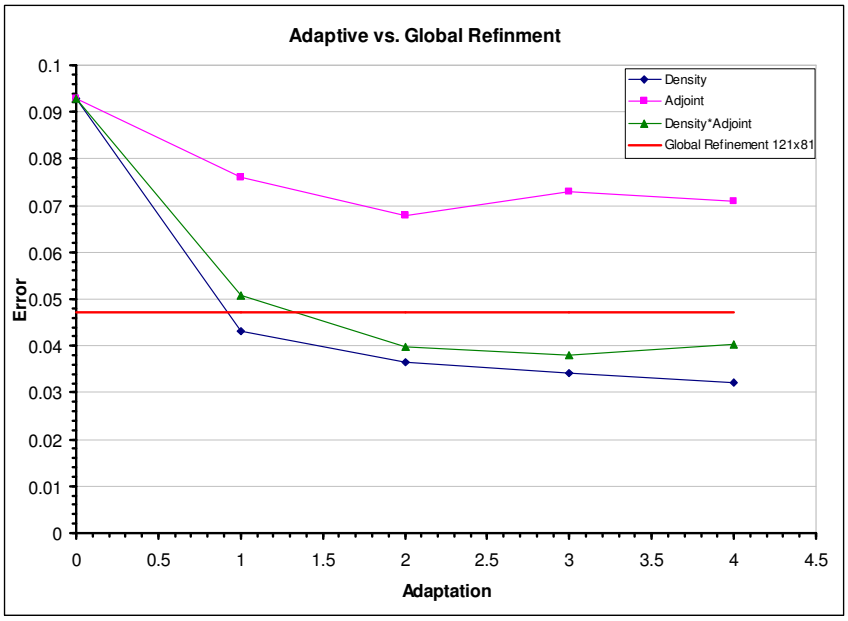


Figure 31: Error in Adaptive Schemes Compared to Error in Global Refinement

After a single adaptation of the density adaptive scheme and after two iterations of the density and adjoint adaptive scheme the solution is more accurate than applying a global grid refinement from 61x41 grid to a 121x81 grid. Since the adaptive grid accomplishes this accuracy without increasing the computational time dramatically, it is preferred over the global refinement.

4 Summary

A study was performed that examines the accuracy with which a CFD simulation could predict the lift and drag coefficients on an airfoil. A double

wedge airfoil in a supersonic free-stream was used because an analytical solution could be found for comparison. Three different grid modifications were made in this study: global refinement, local clustering, and adaptive refinement. For the adaptive refinement, three different functions of the flow solution were used to define the equi-distribution function.

Global refinement found that increasing the resolution of the grid also increased the accuracy of the solution. Unfortunately, the computational time increased significantly as the error decreased. The global refinement was also limited in the accuracy level it could obtain. After the second refinement, the accuracy had only increased slightly, while the computational time increased by a factor of 6.

Local refinement found that the accuracy could be increased greatly by clustering the grid at the leading and trailing edges, as well as at the middle of the airfoil. The accuracy continued to improve as the clustering was increased, until a certain point was reached. After too much clustering, the accuracy decreased because it left other areas of the grid with poor refinement. This method worked the best for the double wedge airfoil in a supersonic flow because the locations of flow features such as shock and expansion waves were known. It would not work as well for a transonic airfoil such as the NACA0012 where the shock location is not known.

Adaptive refinement found that using a refinement parameter based on the density difference produced the most accurate adaptive solution. The density difference scheme adapted the grid well along the shocks and expansion waves. It

also refined the grid behind the airfoil, where the flow has little effect on the forces acting on the airfoil. The adaptive grid based on the adjoint solution caused the grid to be refined in the areas where the flow had the greatest effect on the force calculations. This grid was clustered in front of the airfoil where the flow was uniform, and had very little error. As a result, the adjoint adaptation was less accurate than the density difference adaptation. The final adaptive scheme examined used the product of the density difference and the adjoint solution. This scheme clustered the grid around the flow features near the airfoil and eliminated the refinement of the flow behind the airfoil. The product adaptation resulted in a grid slightly less accurate than the density-difference grid.

Care must be taken in generalizing the results of this study. First, only a 2-D double wedge airfoil in an inviscid supersonic flow was examined. It is quite likely that the details of the adaptation will differ in the presence of strong boundary curvatures and boundary layers. Second, only one CFD scheme was used. Schemes with higher inherent accuracy might also show different behaviors, especially with the adaptive technique. Finally, only isolated airfoil calculations were considered. Internal flows might exhibit different kinds of behaviors not seen here.

With all the warnings described above, it is clear that the adaptive grid refinement is a powerful technique that should become a standard part of all CFD calculations.

5 Bibliography

[1]Giles, Michael B. "On Adjoint Equations for Error Analysis and Optimal Grid Adaptation in CFD"

[2]Thompson, J.F., Thames, F.C and Mastin, C.W., "TOMCAT – A Code for Numerical Generation of Boundary-Fitted Curvilinear Coordinate Systems on Fields Containing Any Number of Arbitrary Two-Dimensional Bodies", *Journal of Computational Physics,* Vol 20, pp 274-302, 1977.

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

[4]Dannenhoffer, J.F., "MBANS2 Program", private communication, 2008.