Optimal adaptive surface grid generation from B-spline geometric models

by

Bethany Lynne Lumba

A Thesis Submitted to the

Graduate Faculty in Partial Fulfillment of the

Requirements for the Degree of

MASTER OF SCIENCE

Department: Aerospace Engineering and Engineering Mechanics
Major: Aerospace Engineering

Iowa State University
Ames, Iowa
1990

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1. INTRODUCTION

Numerical grid generation distributes grid points over a physical field such that physical phenomena may be represented with sufficient accuracy. Numerical grid generation has applications in many areas where the solution of partial differential equations is of interest. The numerical solution of partial differential equations has reached a high state of development, however the ability to deal with complex geometries, like those found in most physical problems, is still under development. The pacing item in advancing numerical procedures for realistic problems is the development of general techniques for numerically constructing mesh systems about complex configurations [1].

The most common type of grids generated are boundary-conforming. With a boundary-conforming curvilinear coordinate system, the boundary conditions can be represented on a coordinate line, or surface, rather than interpolated onto the boundary [2]. In addition, general codes can be written with the boundary shape specified as input. The intersection of the coordinate lines defines the grid points, making identification of neighbors easy. This allows all computations to be done on a fixed rectangular grid in the computational field with the curvilinear coordinates as independent variables.

One of the most active areas of research in numerical grid generation is adaptive grids. In an adaptive grid generation scheme the physics of the problem must direct the grid points to move. For two-dimensional and three-dimensional grids they move according to the solution variation so that the physical solution can be represented with sufficient accuracy. For three-dimensional grids they can also move in response to geometeric properties such as curvature. The grid points must be distributed over the field in an orderly fashion, yet be allowed to redistribute according to the adaption. To accomplish this, a means of communication between the points must exist and a way to sense the variations must be translated into the motion of the grid points. The grid should not become excessively skewed. It must be smooth and tend toward orthogonality to limit trucation errors introduced by nonuniform grids [3, 4].

This paper examines an adaptive grid generation scheme based on the variational method of Brackbill and Saltzman [5, 6]. The variational method produces elliptic partial differential equations resulting from an application of calculus of variations to an optimal mesh formulation. Both two-dimensional interior grids and three-dimensional surfaces are used to show the adaptive generation. The two-dimensional grids demonstrate the solution adaptive capibilities, and the three-dimensional grids adapt to curvature. Before the adaptive grid generation scheme can be used, the surface must be accurately defined and an initial grid must be created on the surface or interior grid. B-splines are the tool used to model the surfaces and an algebraic grid generation scheme is chosen to create the initial grid. This part is taken from the work of Atwood [10]. The major effort of this paper is the adaptive grid generation scheme.

Five test cases were formulated to validate this method. The first three cases examine two-dimensional flow fields and the last two cases examine three-dimensional surfaces. The first two cases are simple square domains with high gradient regions created within the domain. The third case is an airfoil with a shock. The three-dimensional cases are forebodies of the Boeing 747 and the McDonnel Douglas f-18.

The B-spline surfaces and initial grid generation are described in Chapter Two. Chapter Three presents the adaptive method used and describes its implementation. Chapter Four discusses the results obtained with the test cases. Finally, Chapter Five gives the conclusions of this investigation and recomendations for further research.

CHAPTER 2. INITIAL SURFACE GRID GENERATION

The surface grid generation scheme can be divided into two separate parts. The first part is to define the surface accurately. The second part is to create an initial grid.

Surface Definition

B-splines are used to define the surface. For this case, cubic B-splines are used to produce a curvature continuous surface. A characteristic polyhedron is determined such that the resulting B-spline surface passes through the data points specified by the user. Once the characteristic polyhedron is calculated the surface representation can be changed by shifting its vertices, also called control points. These changes are local, affecting only 16 points per vertex shifted as opposed to other surface-defining techniques, Bezier surfaces for example, where local changes are propagated thoughout the entire surface. This makes the method particularly useful for interactive design and locally changing surface details [7, 8].

The general matrix form for the B-spline surface that approximates an $(m+1)X(n+1)$ rectangular array of points is

$$p_{st}(u,v) = U_k M_k P_{kl} M_l^T V_l^T \tag{2.1}$$

$$s \in [1 : m + 2 - k]$$
$$t \in [1 : n + 2 - l]$$

where k and l control the continuity of the surface, $k = l = 4$ for the cubic case, giving the $C2$ continuity; s and t indentify a particular patch in the surface. The quantity $p_{st}(u, v)$ is the array of data points to be interpolated. U and V are the parametric variables.

$$U_k = [u^{k-1}\, u^{k-2} \cdots 1] \tag{2.2}$$
$$V_l = [v^{l-1}\, v^{l-2} \cdots 1]$$
$$u, v \in [0, 1]$$

Elements in the matrix of control points (the vertices of the characteristic polyhedron) depend on which patch is being evaluated.

$$P_{kl} = p_{ij} \tag{2.3}$$
$$i \in [s - 1 : s + k - 2]$$
$$j \in [t - 1 : t + l - 2]$$

M is the matrix of coefficients which are calculated from the blending functions and remain constant for each k and l.

$$M = \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & 6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix} \tag{2.4}$$

Grid Generation

Once the characteristic polyhedron is calculated the parametric variables are used to produce the desired initial grid structure using an algebraic grid generator. Either a uniform or a two-sided clustering function can be used. The two-sided clustering function is very useful for clustering near a body. This method is devised from Vinokur's one-dimensional stretching functions [9]. The slopes are specified by the user who then controls the clustering. This provides the user with a method to discretize according to the flowfield variables if the characteristics of the solution are known beforehand. This will decrease the amount of movement necessary in the grid to reach the optimum level, thus decreasing the time for convergence of the grid relaxation process. Figure 2.1 shows several examples of clustering for various endpoint slope values.

A uniform parametric grid is an evenly spaced set of parametric variables corresponding to the input grid (Figure 2.2 and Figure 2.3). Note that the parameteric variables are uniform, not the actual physical grid which remains as specified. The uniform clustering is used to create the initial grid because it corresponds to an elliptic grid generator with no adaption.

The entire grid (2-D or 3-D) is contained in the two-dimensional parametric variables. The characteristic polyhedron is the key to the transformation between the parametric variables and the cartesian coordinates. Once the initial grid and charateristic polyhdron are created and stored then the adaptive grid generation is used.

A FORTRAN code called SURFGRID (Surface and Grid Representation with

Interactive Development) provides the surface and initial grid generation described in this chapter [10].

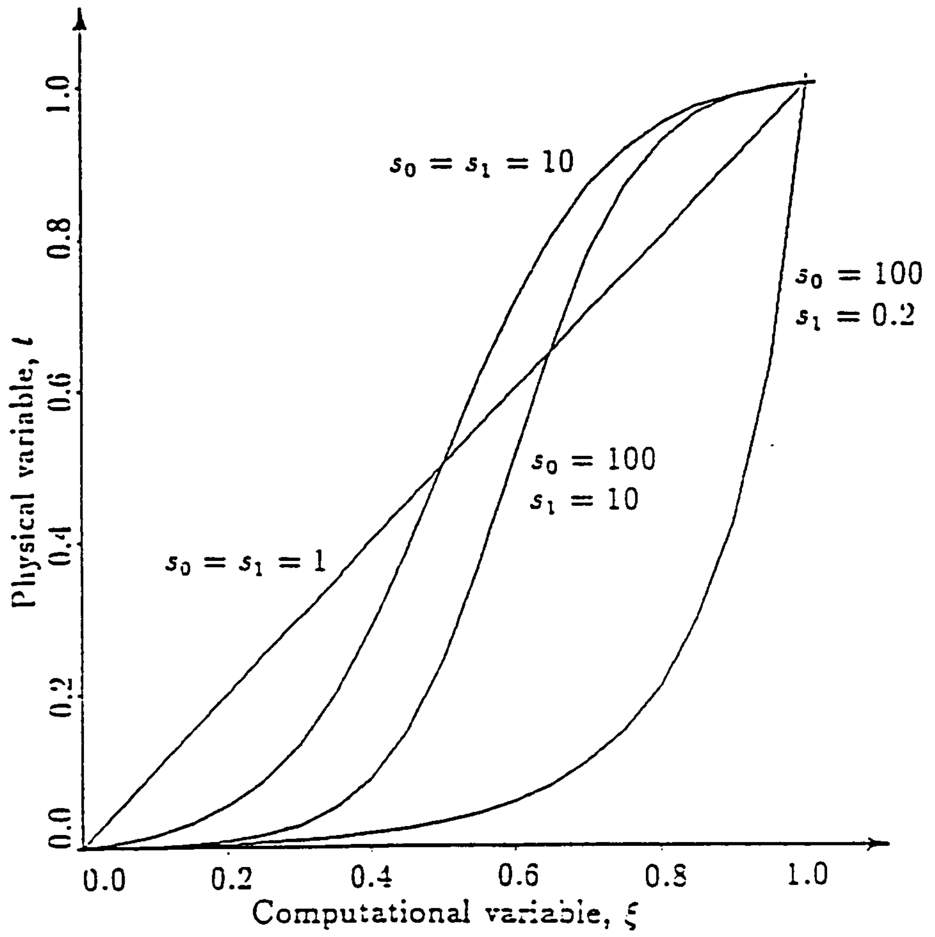

Figure 2.1: Vinokur's parametric clustering

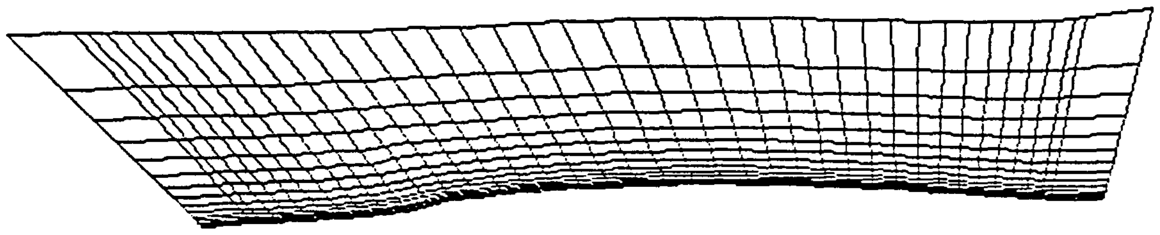

Figure 2.2: Input grid

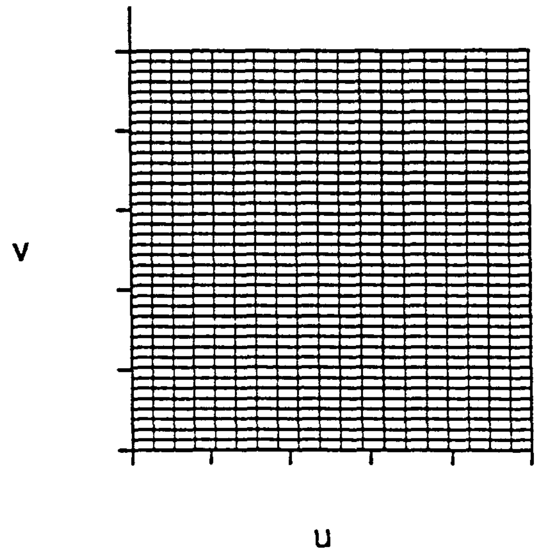

Figure 2.3: Parametric grid

CHAPTER 3. ADAPTIVE GRID GENERATION

A variational approach by Brackbill and Saltzman is used to generate the two-dimensional adaptive boundary fitted coordinates. The system is constructed with three parts, smoothness, orthogonality, and, adaptivity. Each of these criteria is represented by a term in an integral equation which is then minimized using the calculus of variations.

The linear combination of these integral equations is

$$I = I_s + \lambda_o I_o + \lambda_v I_v \tag{3.1}$$

where I_s, I_o, and I_v represent the integral equations for smoothness, orthogonality, and adaptivity respectively. λ_o and λ_v are scalar factors which determine the weight of the orthogonal and adaptive contributions.

Equation Development

First, smoothness is needed to reduce the truncation error of the solution [11]. To maximize the smoothness of the grid the integral of the quantity $(\nabla\xi \cdot \nabla\xi + \nabla\eta \cdot \nabla\eta)$ is minimized over the parametric domain.

$$I_s = \int\int (\xi_u^2 + \xi_v^2 + \eta_u^2 + \eta_v^2)\, du\, dv \tag{3.2}$$

where (ξ, η) are the computational coordinates and (u, v) are the parametric coordinates. This corresponds to the elliptic grid generation system.

$$\nabla^2 \xi_i = \nabla^2 \eta_i = 0 \tag{3.3}$$

In the above equations (ξ, η) are the dependent variables. However, (u, v) are the preferred dependent variables. After interchanging them, equation 3.1 is written as

$$I_s = \int\int (u_\xi^2 + u_\eta^2 + v_\xi^2 + v_\eta^2)\frac{1}{J}\, d\xi\, d\eta \tag{3.4}$$

with integration over the computational domain. J, the Jacobian, is given by

$$J = u_\xi v_\eta - u_\eta v_\xi \tag{3.5}$$

With this variational formulation the expansion of the Lagrangian functions to include other factors is possible because of the linearity. The second factor Brackbill and Saltzman use is a measure of the orthogonality of the grid, which vanishes as the grid becomes more orthogonal.

$$I_o = \int\int (\nabla\xi \cdot \nabla\eta)^2 J\, du\, dv \tag{3.6}$$

The equation with (ξ, η) as independent variables is

$$I_o = \int\int (u_\xi u_\eta + v_\xi v_\eta)^2 \frac{1}{J^2}\, d\xi\, d\eta \tag{3.7}$$

The $\frac{1}{J^2}$ is there to maintain dimensional consistency between the two integrals. The Jacobian is used because it is always non-zero. However, only the $(\nabla\xi \cdot \nabla\eta)^2$ is necessary to ensure the orthogonality constraint, J^2 is completely arbitrary. Therefore Brackbill and Saltzman drop the J^2 and absorb it into the scale factor λ_o. This makes the Euler-Lagrange equations much less complicated [12], but by absorbing the J^2 into the constant λ_o it is questionable whether the solution to the partial differential equation is actually a minimum to the original intergral. The equation used for the calculations is

$$I_o = \int\int (u_\xi u_\eta + v_\xi v_\eta)^2 \, d\xi \, d\eta \tag{3.8}$$

The third factor is the adaption term,

$$I_v = \int\int w^2(\mathbf{r}) J \, du \, dv \tag{3.9}$$

where $w(\mathbf{r})$ is a specific forcing function.

$$\mathbf{r} = \begin{pmatrix} u \\ v \end{pmatrix}$$

After the change of variables the equation becomes

$$I_v = \int\int w^2(\mathbf{r}) J^2 \, d\xi \, d\eta \tag{3.10}$$

integrated over the computational domain.

The grid generating system is obtained by minimizing the linear combination of these three integrals. Equation 3.1 is shown again for convenience.

$$I = I_s + \lambda_o I_o + \lambda_v I_v \tag{3.11}$$

The choices of λ_o and λ_v will determine the influence of the orthogonality and adaptivity terms. For example, a large λ_o will produce a grid which is nearly orthogonal at the cost of smoothness and adaptivity. But, if λ_o and λ_v are too large the grid becomes skewed and errors may increase.

To minimize this equation, the Euler-Lagrange equations are applied to each of the Lagrangian functions yielding two partial differential equations whose solution minimizes the integral. These calculations are shown in the Appendix. The final results are written here.

$$A\mathbf{r}_{\xi\xi} + B\mathbf{r}_{\xi\eta} + C\mathbf{r}_{\eta\eta} + D\mathbf{r}_{\xi} + E\mathbf{r}_{\eta} = 0 \tag{3.12}$$

This equation is a quasilinear second-order partial differential equation with coefficients which are quadratic functions of the first derivatives. This equation is then discretized with second order central differences for $\mathbf{r}_{\xi}$, $\mathbf{r}_{\eta}$, $\mathbf{r}_{\xi\xi}$, $\mathbf{r}_{\eta\eta}$, and $\mathbf{r}_{\xi\eta}$. The resulting finite difference approximation is solved with a point relaxation scheme for the new grid.

Boundary Points

Since only the interior points are moved with the grid adaption scheme it is necessary to treat the boundary points separately. In order to have the influence of the weight function, the boundary points are calculated from the interior points. The calculations are in parametric space since one of the parametric values is known to be

0 or 1 on all boundaries. The other parametric variable is calculated in one-dimension using the criteria of constant curvature near the boundary (Figure 3.1).

$$q_1 = 2q_2 - q_3 \tag{3.13}$$

where q is either u or v along a constant parametric coordinate line.

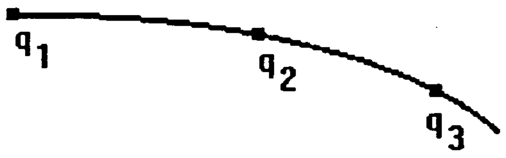

Figure 3.1: Boundary Point

Weight Function

The weight function plays a key role in adaptive grids. The most commonly used form of the weight function is a linear combination of functions given as

$$w = 1 + cM_1 + c_2M_2 + c_3M_3 + \ldots \tag{3.14}$$

where M_i are non-negative functions and c_i are non-negative coefficients to indicate the level of influence attached to them. The '1' guarantees the weight function will never be zero. Any function which produces a non-negative scalar field on the domain is a possibility. For the two-dimensional cases examined in this paper, gradients of scalar quantities in the solution are used to force the adaptive process. The gradient is chosen for the weight function because this will put the most grid points where the solution is changing the most. With three-dimensional surfaces, the emphasis might be on geometric quantities such as curvature, which allows for a better representation of the surface [13]. Because these quantities are associated with the physical location, and not some auxiliary quantity, the weight function is taken to be a function of the physical space, **r**.

Because the grid points are moving with each iteration it is necessary to interpolate the scalar field onto the new grid before the weight function is calculated. This assures that the scalar field remains fixed in physical space while the points move.

The weight function used in this paper is

$$w = 1 + \alpha g \tag{3.15}$$

g can be the magnitude of the gradient of the scalar quantities: velocity, density, or pressure. For surfaces, g is the mean curvature.

$$\begin{aligned} g = H &= \frac{1}{2}(\kappa_1 + \kappa_2) \\ &= \frac{EN + GL - 2FM}{2(EG - F^2)} \end{aligned} \tag{3.16}$$

$$\begin{aligned} E &= \mathbf{p}_u \cdot \mathbf{p}_u \quad & L &= \mathbf{p}_{uu} \cdot \mathbf{n} \\ F &= \mathbf{p}_u \cdot \mathbf{p}_v \quad & M &= \mathbf{p}_{uv} \cdot \mathbf{n} \\ G &= \mathbf{p}_v \cdot \mathbf{p}_v \quad & N &= \mathbf{p}_{vv} \cdot \mathbf{n} \end{aligned} \tag{3.17}$$

$\mathbf{p} = \mathbf{p}(u, v)$ the parametric surface patch

$\mathbf{n}$ is the unit normal to the surface

where κ_1 and κ_2 are the principle curvatures and E,F,G,L,M,and N are coefficients of the first and second fundemental forms.

Alpha, in equation 3.15 is a positive scalar factor, which has the most influence on the movement of the grid. The scale factor should be small enough to ensure stability, and yet large enough to achieve the desired representation [14].

The values of the weight function drive the movement of the grid points. The points are attracted to the regions where the weight function is high. In these regions there will be a better representation of the solution, thus reducing truncation errors.

The adaptive grid generation scheme described in this chapter is provided by a FORTRAN computer code called OptSGrid.

CHAPTER 4. NUMERICAL RESULTS

The method described in the previous chapters was tested on five cases. The first two cases are two-dimensional rectangular domains on which high gradient regions were mathematically created. The third case is from actual data about an f-16. The forebodies of a f-18 and a 747 are examined in the last two cases, which demonstrates the three-dimensional capabilities of this method.

An overview of the method used to create an adapted grid using SURFGRID and OptSGrid is shown in Figures 4.1 and 4.2. This was the method used for the following test cases.

Test Cases I and II

A two-dimensional square domain [-2,2] X [-2,2] was created for these test cases. Using SURFGRID (the program discussed in Chapter Two) each case was parameterized and the characteristic polyhedron was calculated. Since a uniform parameteric grid corresponds to a smooth grid without adaption this initial grid generation technique is preferred. With this choice the grid points will move less during the adaption process therefore achieving convergence in fewer iterations. A 30 X 50 initial square grid was chosen and is shown in Figure 4.4.

A high gradient region was created in the shape of a unit circle for the first case

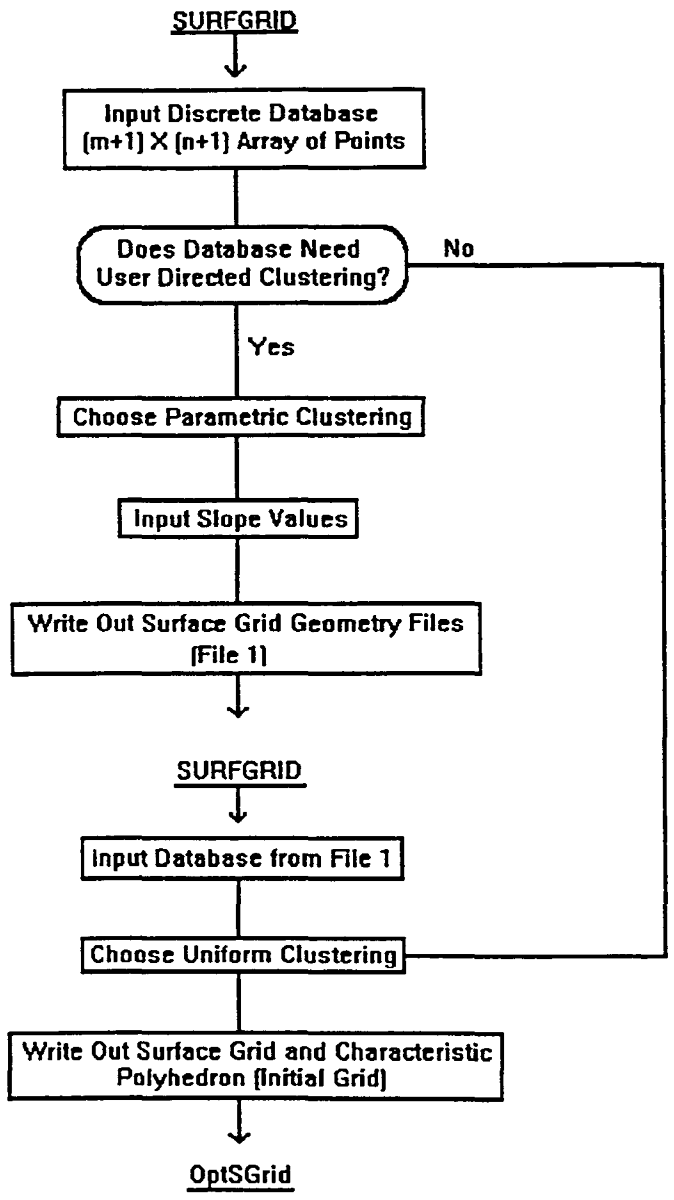
SURFGRID
Input Discrete Database (m+1) X (n+1) Array of Points
Does Database Need User Directed Clustering?
No
Yes
Choose Parametric Clustering
Input Slope Values
Write Out Surface Grid Geometry Files (File 1)
SURFGRID
Input Database from File 1
Choose Uniform Clustering
Write Out Surface Grid and Characteristic Polyhedron (Initial Grid)
OptSGrid

Figure 4.1: Overview of Method (SURFGRID)

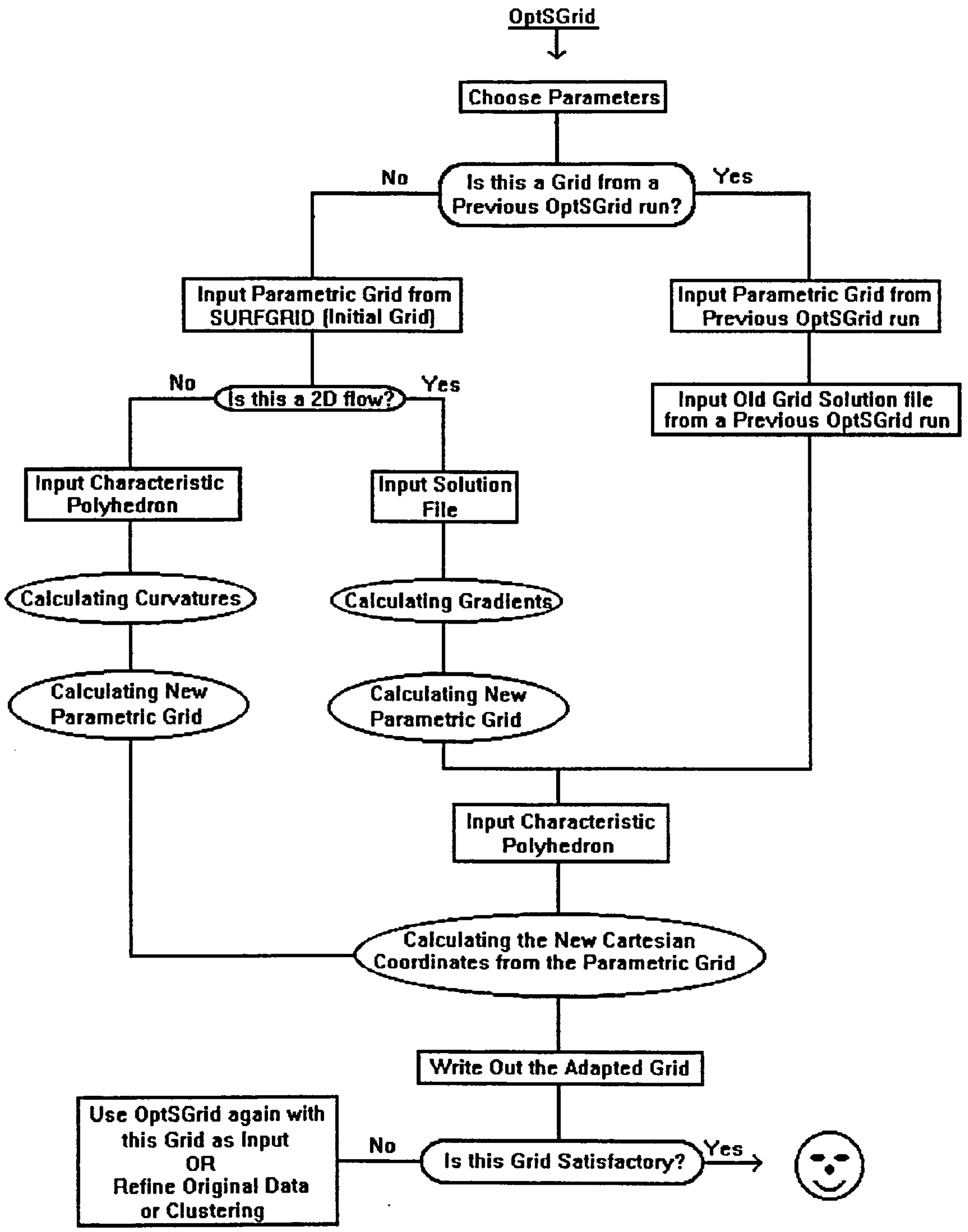
OptSGrid
Choose Parameters
Is this a Grid from a Previous OptSGrid run?
No
Yes
Input Parametric Grid from SURFGRID (Initial Grid)
Input Parametric Grid from Previous OptSGrid run
Is this a 2D flow?
No
Yes
Input Old Grid Solution file from a Previous OptSGrid run
Input Characteristic Polyhedron
Input Solution File
Calculating Curvatures
Calculating Gradients
Calculating New Parametric Grid
Calculating New Parametric Grid
Input Characteristic Polyhedron
Calculating the New Cartesian Coordinates from the Parametric Grid
Write Out the Adapted Grid
Use OptSGrid again with this Grid as Input OR Refine Original Data or Clustering
No
Is this Grid Satisfactory?
Yes

Figure 4.2: Overview of Method (OptSGrid)

and a parabola for the second case. The solution was generated using

$$f(x,y) = (1 + tanh(1 - x^2 - y^2))\frac{1}{2} \tag{4.1}$$

for the unit circle, and

$$f(x,y) = tanh(y - 2x^2 + 1) \tag{4.2}$$

for the parabola, on the grid in Figure 4.4.

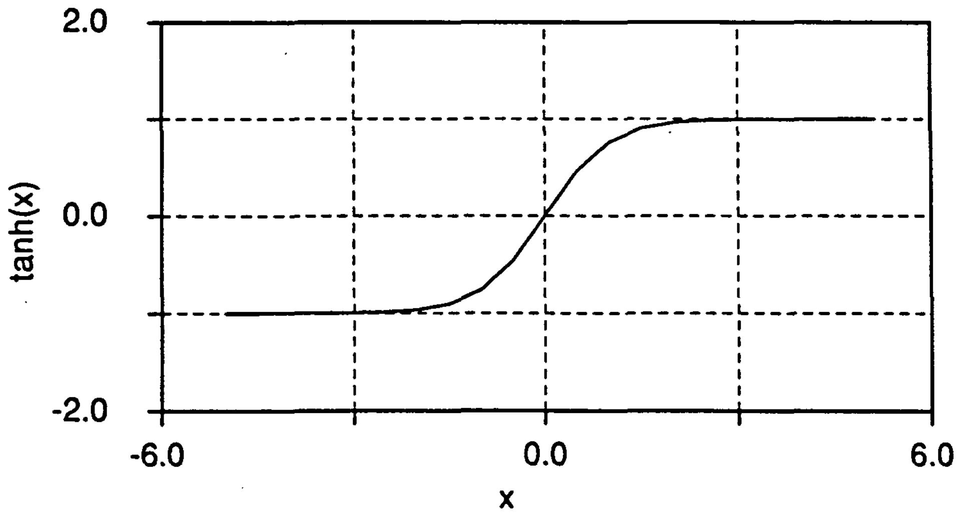

Figure 4.3: Tanh(x)

The value of *tanh* changes rapidly near the origin and is nearly constant everywhere else. That makes the gradient large along the circle and the parabola. Therefore, choosing the weight function to be the gradient of the solution causes points to be attracted to these regions. This leads to the clustering in the conic shapes.

Figures 4.5-4.9 show grids from test cases one and two. Figure 4.5 was created using the the grid in Figure 4.4 as the initial grid and the solution from equation 4.1 as the initial solution. The parameters were $\lambda_o = 0$, $\lambda_v = 140$, and $\alpha = 400$. Figure 4.6 was created using the results from Figure 4.5 as the initial grid and solution, and with parameters $\lambda_o = 0$, $\lambda_v = 150$, and $\alpha = 600$. Using the adapted grid as the initial grid for the second run allows the user to use an iterative process when designing a grid. It also allows the user to change the parameters and check the progress of the adaption. Sometimes it is necessary to keep the parameters low for better stabilty initially, then raise them later to get a more concentrated grid in the high gradient regions. This is the case for the grid in Figure 4.6 which is more tightly clustered than the grid in Figure 4.5.

For this test case the boundary conditions don't play a large part because the adaption is away from the boundaries. But in test case two the high gradient region intersects the boundary so the movement of the boundary points is demonstrated.

Figure 4.7 was created using the grid in Figure 4.4 as the initial grid and the solution calculated in equation 4.2 as the initial solution. The parameters were $\lambda_o = 0$, $\lambda_v = 120$, and $\alpha = 600$. This was used as the initial grid and solution for Figure 4.8. The parameters were increased to $\lambda_o = 0$, $\lambda_v = 200$, and $\alpha = 800$. As seen in the first test case the higher parameters produce the grid with a more concentrated high gradient region. The boundary points move according to the interior points. Constant curvature in parametric space means smooth curvature in physical space. In both these grids the bottom of the parabola is not well defined. Because the gradient of the weight function is much greater in the y-direction the movement in that direction is large compared with the x-direction. Near the high curvature region the gradients are not as large along the parametric coordinate lines therefore the movement is small in comparison. Figure 4.9 shows the same solution on an initial grid of 30 X 30. With spacing the same in both directions the parabola looks similar so the spacing has little influence in this case.

Test Case III

The third test case examines an airfoil. This airfoil is a slice from a solution for a transonic flow about an f-16 [15]. The solution and grid are given on the top of the airfoil only. Using the boundaries of that grid, a simple uniform 5 X 10 grid was created in the region. This was input into SURGRID to create a parametrically clustered 16 X 30 grid. The slopes input were $s_0 = 200$ and $s_1 = 2$. Because of the transonic flow, the clustering is necessary to resolve the boundary layer near the surface. This grid is shown in Figure 4.10 and the corresponding parametric grid is shown in Figure 4.12. Because a uniform parametric grid is preferred for the grid adaption scheme, this grid is input into SURFGRID and the uniform grid generation option is used. This will be the initial grid and the characteristic polyhedron used in OptSGrid. This grid is 20 X 40 and is shown in Figure 4.11. The corresponding parametric grid is shown in Figure 4.13. The grids in Figures 4.10 and 4.11 are very similar, but the corresponding parametric variables are different.

Figures 4.15-4.16 show the grids calculated with OptSGrid. For this test case the weight function is based on the gradient of density. The contours of density in Figure 4.14 show the highest gradients to be near the leading edge. The grid in Figure 4.15 was created with $\lambda_o = 0$, $\lambda_v = 100$ and $\alpha = 500$. The parameters for the grid in

Figure 4.16 were $\lambda_o = 0$, $\lambda_v = 150$, and $\alpha = 800$ with the first grid as the input. Both these grids show movement towards the leading edge. As seen in the preceding section, the second grid has tighter clustering because of the higher parameters. There is also clustering toward the airfoil surface because of the high density gradients in the boundary layer. Without the initial clustering with SURFGRID this would not have been detected.

Test Cases IV and V

The fourth and fifth test cases demonstrate the three dimensional capabilities of this code. The weight function for these three dimensional surfaces is based on the mean surface curvature. The adaption is toward the high curvature regions.

The first surface examined is the forebody of the 747. Because of the symetric nature only half the data were used. The nose portion was not resolved accurately therefore it was not used either. In order to have more grid points at the front, the data were preprocessed with SURFGRID. The parametric clustering option was used for the grid generation with one breakpoint in the i-direction at $i = 10$ [10]. The slopes were $s_0 = 2$ and $s_1 = 2$. To create the initial grid and characteristic polyhedron, SURFGRID was used with the uniform parametric grid generation option. Figure 4.17 shows the initial grid with the uneven spacing while Figure 4.18 is the corresponding parametric grid.

Once the initial grid is known and the characteristic polyhedron is calculated, OptSGrid is run. The parameters for the grid in Figure 4.19 are $\lambda_o = 0$, $\lambda_v = 100$, and $\alpha = 500$. The area about the windshield provides the highest curvature therefore the points have moved toward this region. Figure 4.20 was created from this grid with parameters $\lambda_o = 0$, $\lambda_v = 150$, and $\alpha = 800$.

The grids in Figures 4.19 and 4.20, as well as all the grids calculated with the uniform clustering option in SURFGRID, have larger spacing near the boundary. This comes from the formation of the characteristic polyhedron. Since all the calculations in OptSGrid are done in parametric space (which is uniform throughout) this doesn't affect them, only the final cartesian grid is affected. Unfortunately the boundary for this case falls in the middle of a high curvature region. The problem should be set up with the high curvature or high gradient regions away from the boundary.

The fifth test case is the front section of the f-18. The data extend from the nose back to behind the cockpit. There are several high gradient regions in this area.

For this test case the database was not ordered smoothly on the surface, therefore the leading edge of the wing was not explicitly defined. In order to have a sharp leading edge a straight line was put through the points at the edge of the data. This does not guarantee an accurate leading edge. A double line of data was input along this line, making the continuity C1 at these points. A uniform 20 X 40 grid was used as the initial grid (from SURFGRID). This grid is shown in Figure 4.21. Figure 4.22 shows a cutaway of this initial grid, looking at the bottom half of the wing. This grid was taken from the data supplied, and therefore retained some of its characteristics making it less uniform than the 747 initial grid. The grid in Figure 4.23 was calculated from the initial grid with the parameters $\lambda_o = 0$, $\lambda_v = 150$, and $\alpha = 800$. The same cutaway as before is shown. In this grid the points have slid towards the leading edge. There is also movement along the edge towards those areas with the highest curvatures. These curvatures arise because some points slid off the leading edge when the initial grid was generated. Some may be above or below the edge, not on the edge as desired. This influenced the calculation of the curvatures and, therefore the weight function. Though there are several regions in the grid which have high gradients, the region near the leading edge of the wing seems to have the most activity. This is because the curvature here is the highest. The other high curvature areas, such as those near the nose and the cockpit, though small in comparison, also show some activity.

In the solution of these five test cases the procedure has been robust. The adapted grids look smooth and adapt as expected.

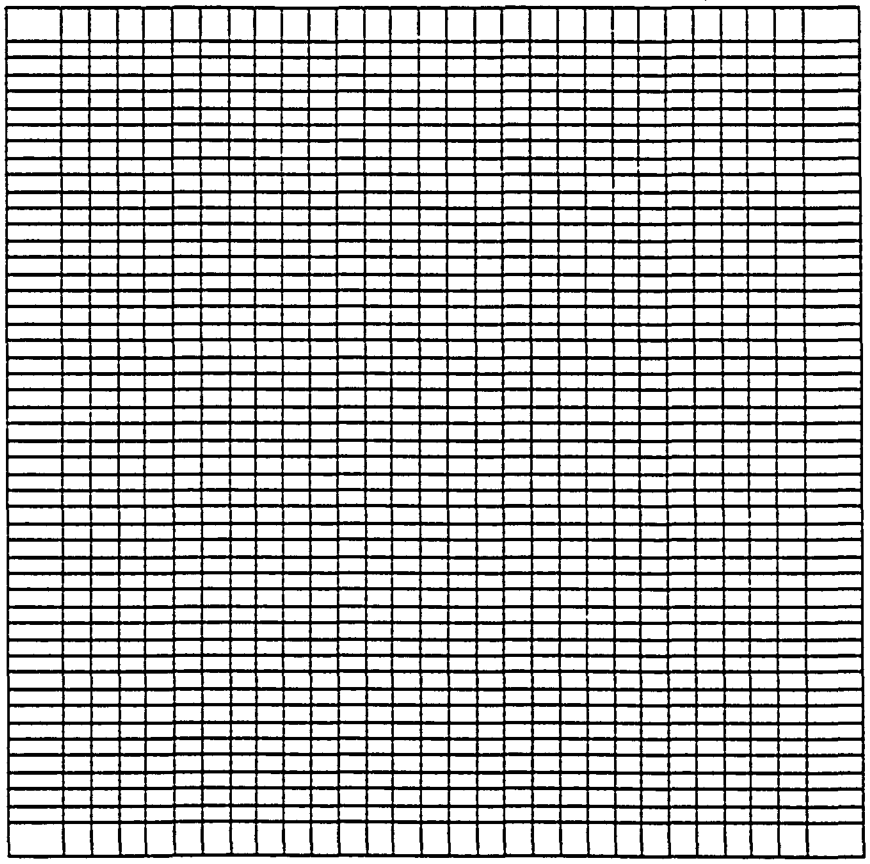

Figure 4.4: The 30 X 50 initial grid

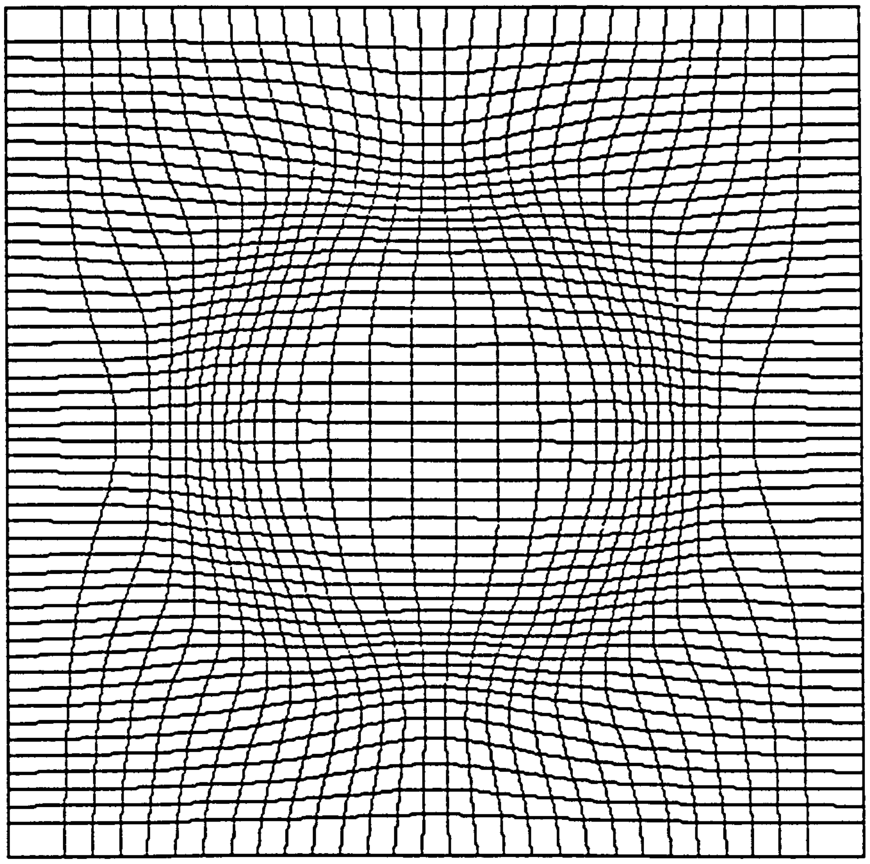

Figure 4.5: Grid adapted to unit circle with parameters $\lambda_o = 0$, $\lambda_v = 140$, and $\alpha = 400$

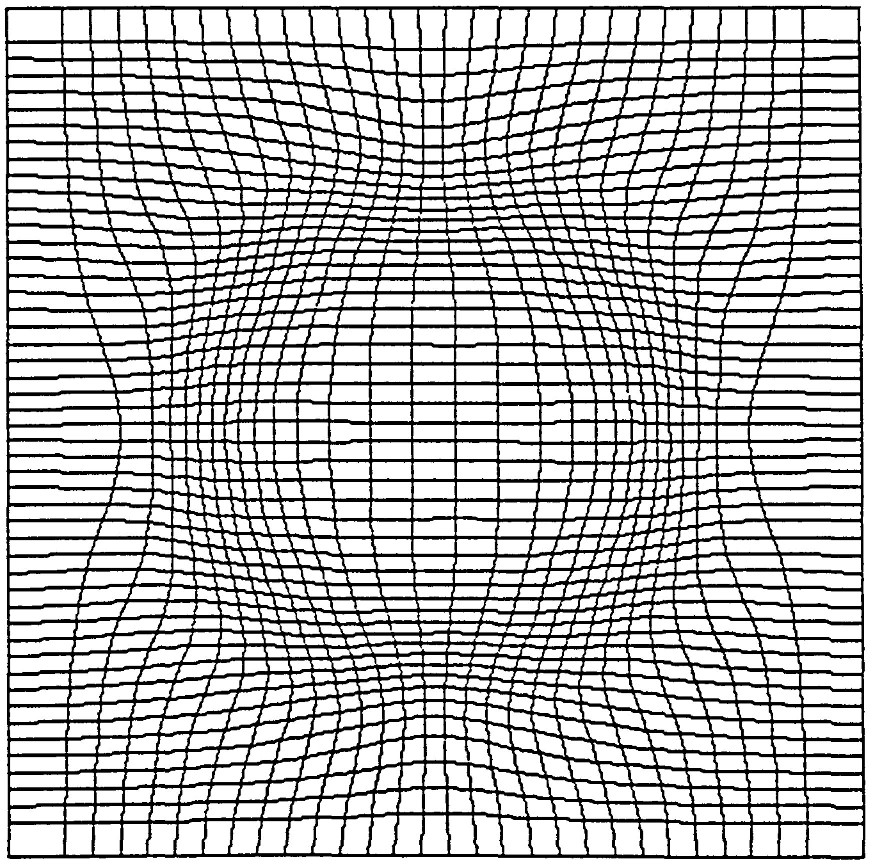

Figure 4.6: Grid adapted to unit circle with parameters $\lambda_o = 0$, $\lambda_v = 150$, and $\alpha = 600$

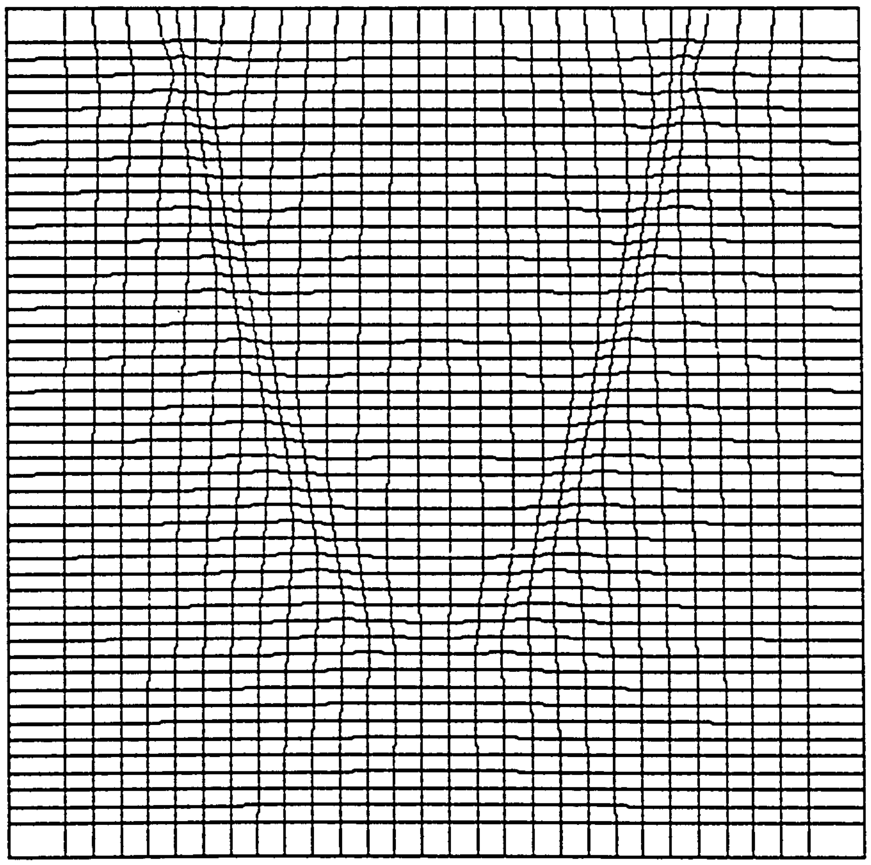

Figure 4.7: Grid adapted to parabola with parameters $\lambda_o = 0$, $\lambda_v = 120$, and $\alpha = 600$

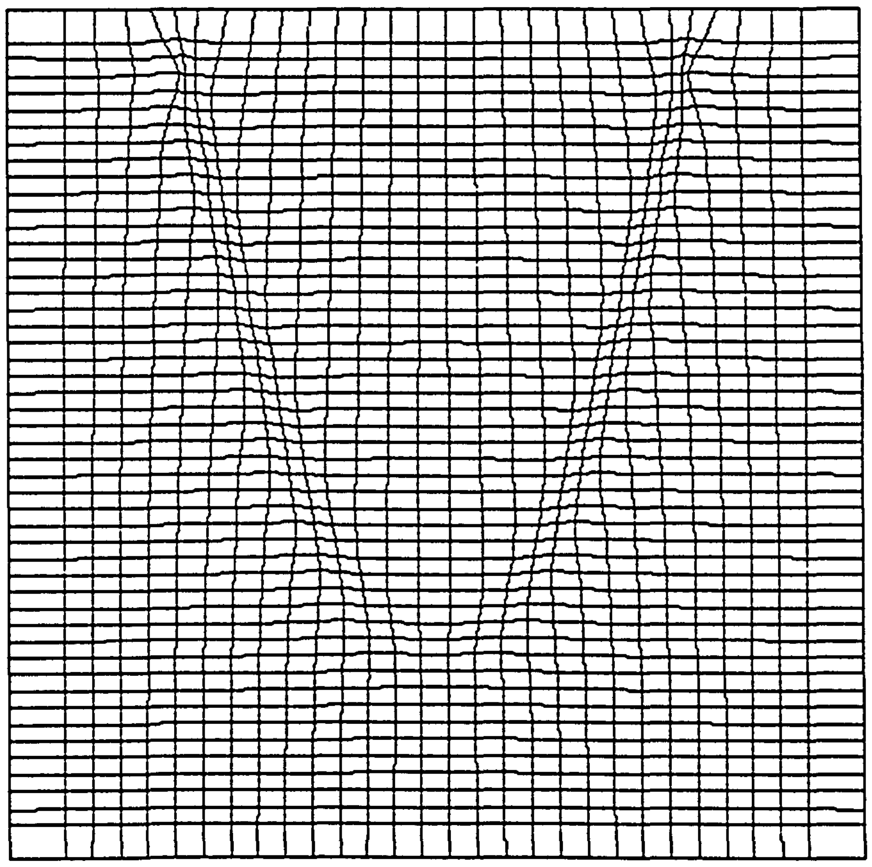

Figure 4.8: Grid adapted to parabola with parameters $\lambda_o = 0$, $\lambda_v = 200$, and $\alpha = 800$

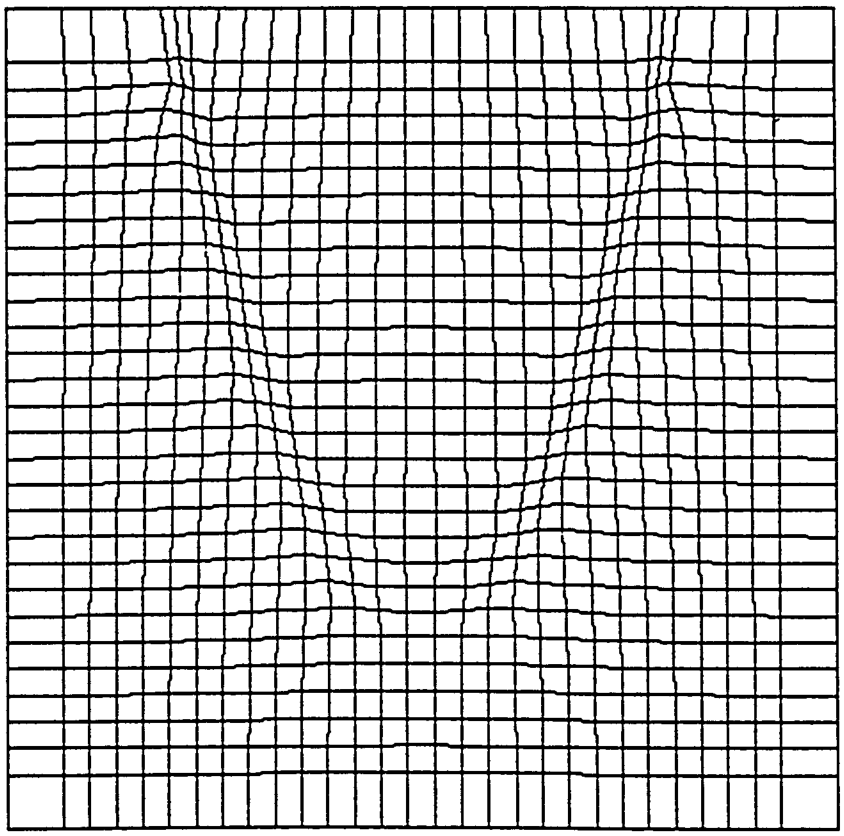

Figure 4.9: Grid adapted to parabola with parameters $\lambda_o = 0$, $\lambda_v = 120$, and $\alpha = 600$ (30 X 30 grid)

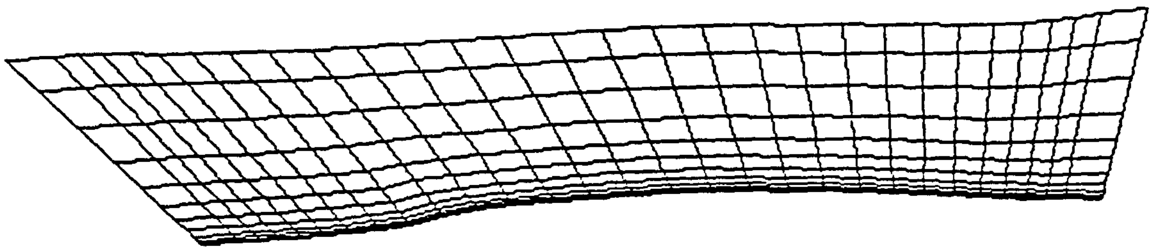

Figure 4.10: Parametrically clustered grid around the airfoil

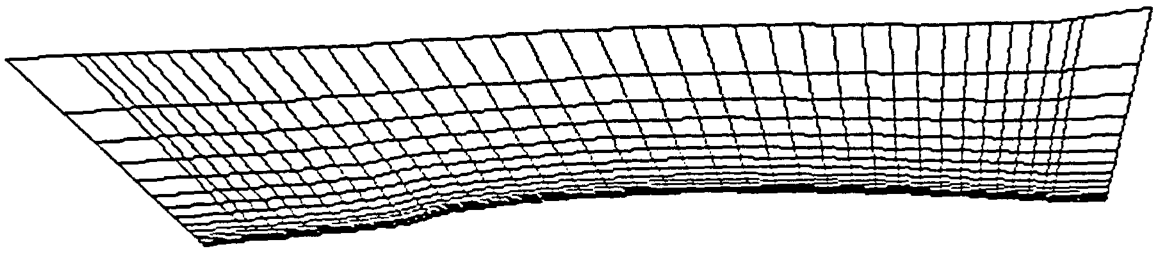

Figure 4.11: Initial grid for airfoil

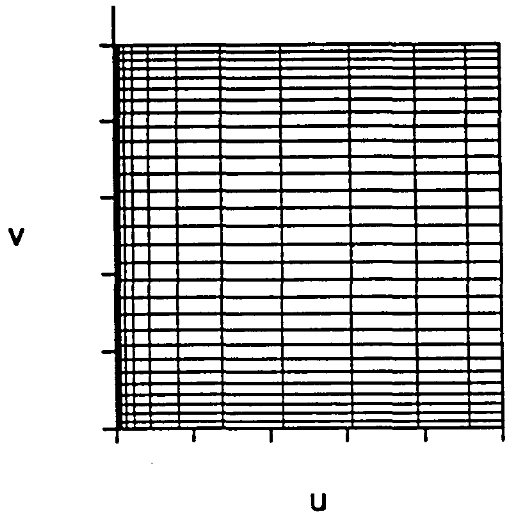

Figure 4.12: Parametric variables corresponding to Figure 4.10

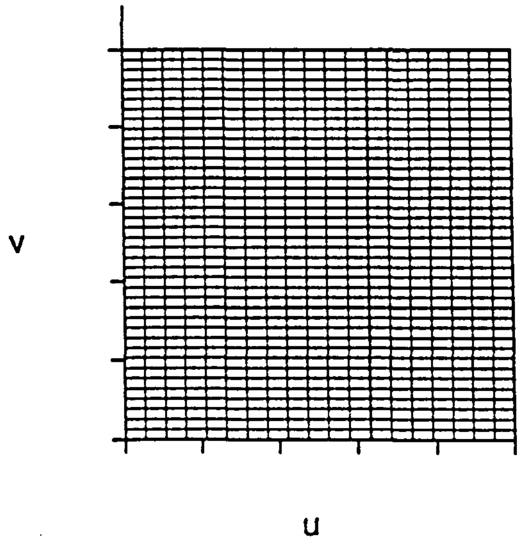

Figure 4.13: Uniform parametric variables corresponding to Figure 4.11

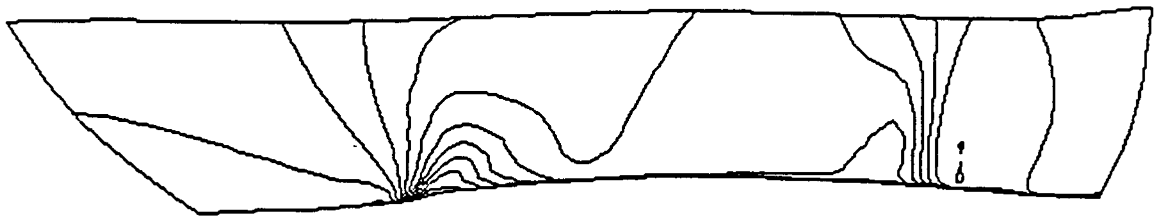

Figure 4.14: Density contours above the airfoil

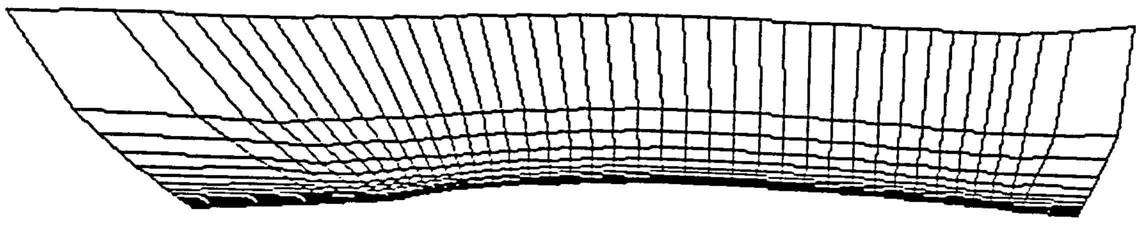

Figure 4.15: Grid adapted about the airfoil with parameters $\lambda_o = 0$, $\lambda_v = 100$, and $\alpha = 500$

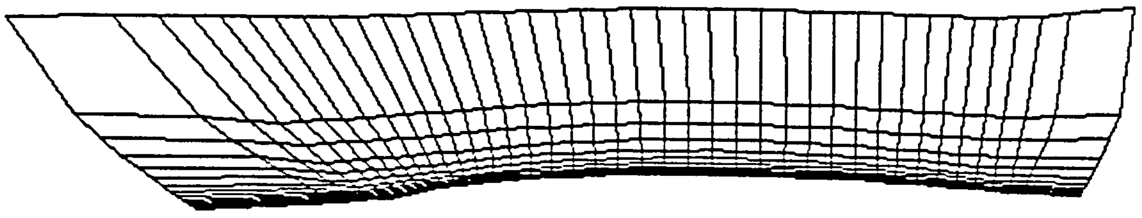

Figure 4.16: Grid adapted about the airfoil with parameters $\lambda_o = 0$, $\lambda_v = 150$, and $\alpha = 800$

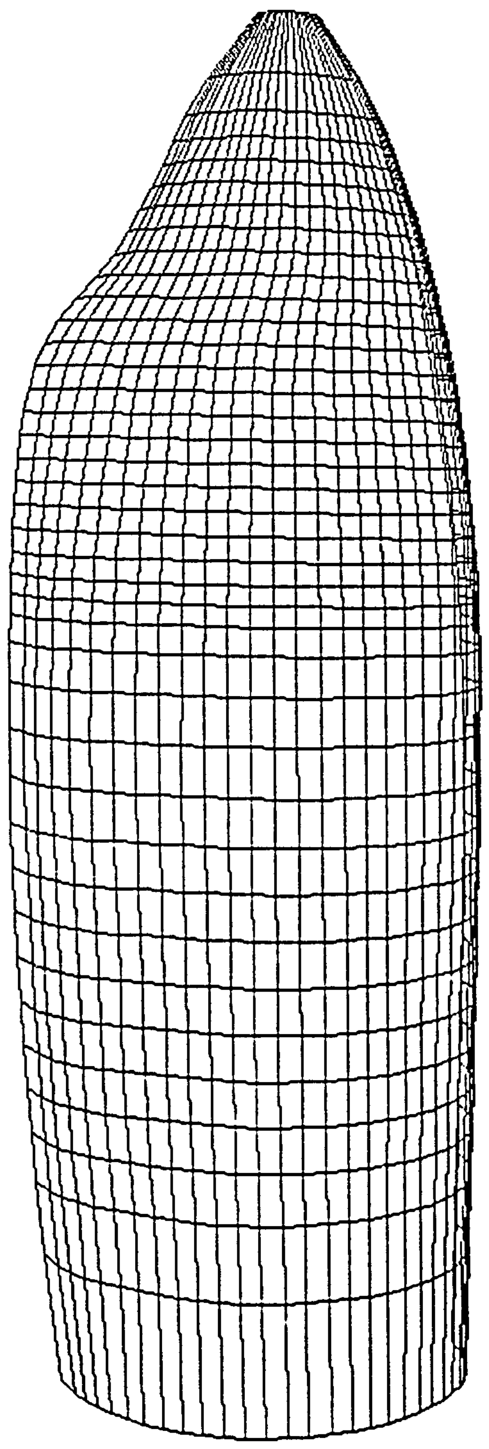

Figure 4.17: Initial grid for 747

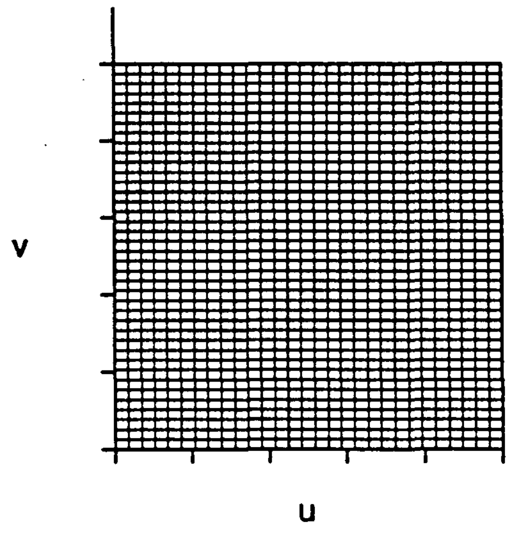

Figure 4.18: Parametric variables corresponding to Figure 4.17

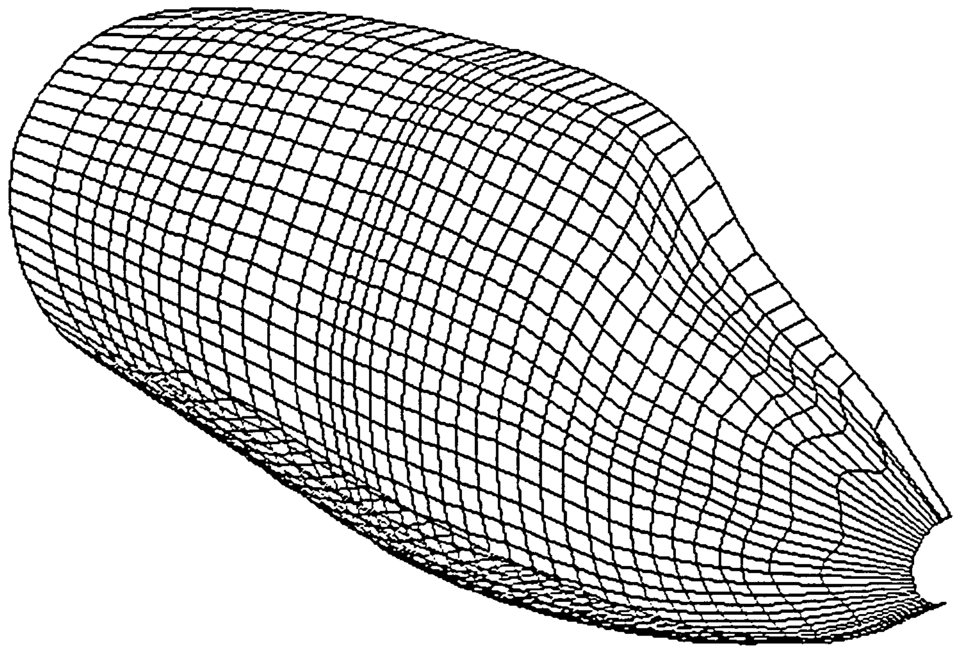

Figure 4.19: Grid adapted on a 747 surface with parameters $\lambda_o = 0$, $\lambda_v = 100$, and $\alpha = 500$

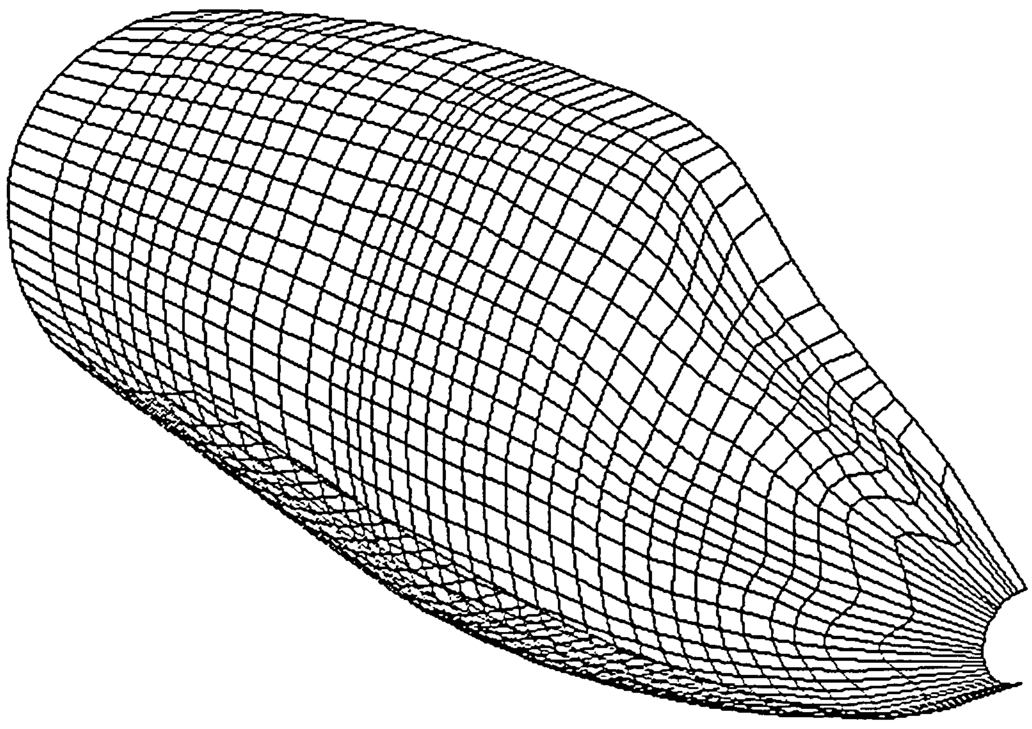

Figure 4.20: Grid adapted on a 747 surface with parameters $\lambda_o = 0$, $\lambda_v = 200$, and $\alpha = 800$

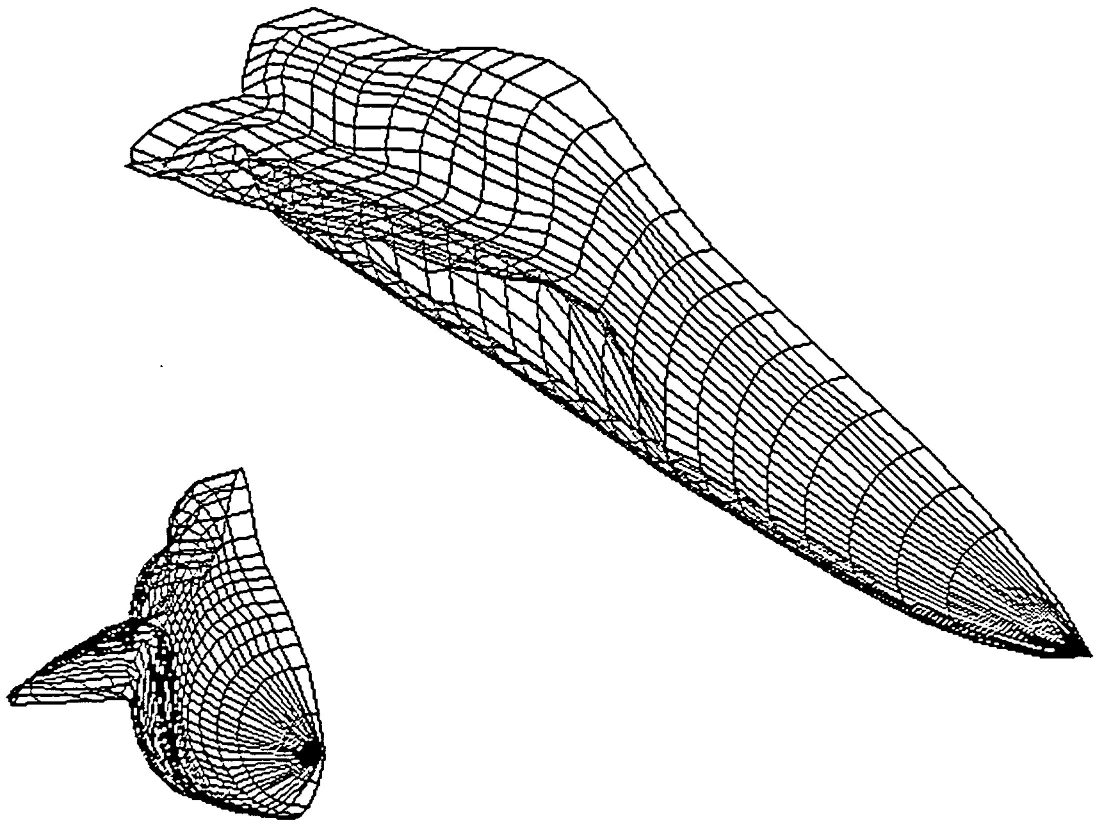

Figure 4.21: The f-16 initial grid, side view and front view

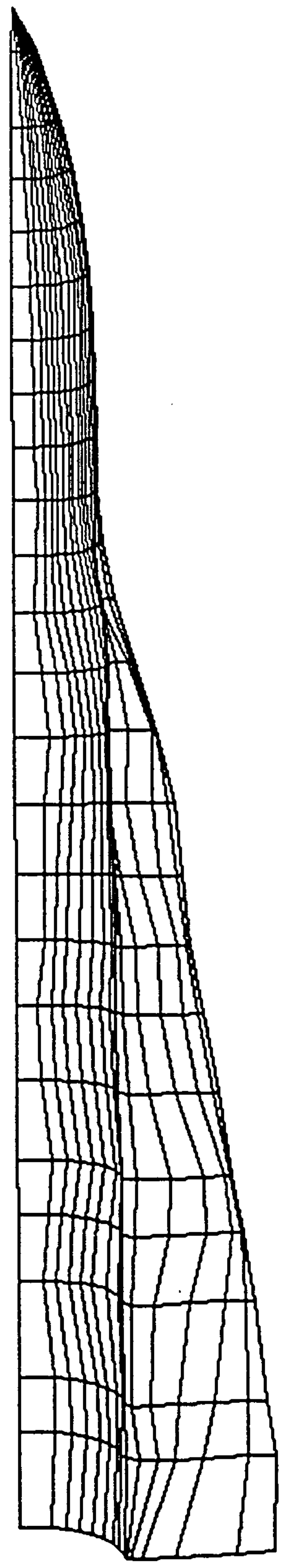

Figure 4.22: The f-18 initial grid, cutaway showing the bottom of the wing

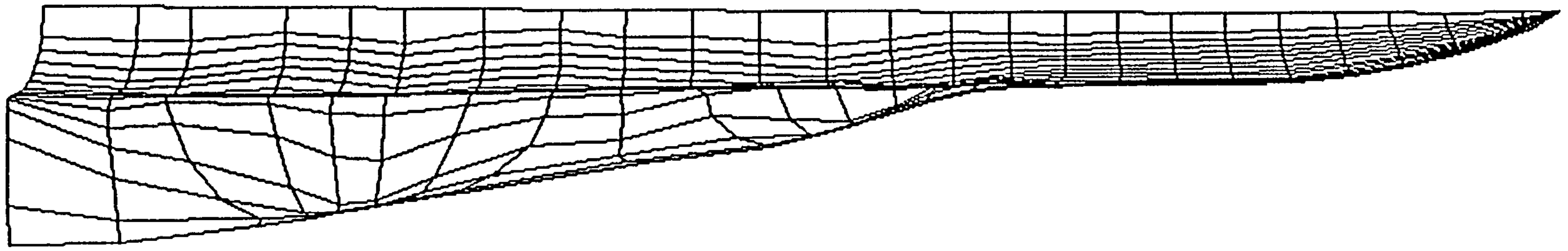

Figure 4.23: Grid adapted on a f-18 surface with parameters $\lambda_a = 0$, $\lambda_o = 150$, and $\alpha = 800$, cutaway showing the bottom of the wing

CHAPTER 5. CONCLUSIONS AND RECOMMENDATIONS

A procedure for adaptive grid generation using B-spline surfaces is presented. The Brackbill and Saltzman variational technique used with B-spline surfaces allows two-dimensional flows and three-dimensional surfaces to be treated similarly. The B-spline surfaces are used to obtain the parametric variables for the grid and the characteristic polyhedron. These are then input (along with the solution for two-dimensional grids) into OptSGrid, a code developed to adapt the grid to the solution for a two-dimensional case, or curvature of the surface for a three-dimensional case.

One of the major weaknesses of this technique is that the adjustments on the grid are ultimately limited by the basic parameters which are specified by the user [14]. This procedure calculates the optimum grid for the given parameters, but the user chooses the parameters. Further investigation into optimizing the parameters, as well as the grid is needed. In fact, further research in several directions is possible.

In order to get a full three-dimensional grid the resulting surface grid is designed to be input as a boundary for 3DGRAPE [16]. A fully adaptive three-dimensional grid is the ultimate goal of this research.

A short term goal is to localize the adaption. As seen in test case five, with large grids there may be more than one high curvature area. Only the highest weight function value significantly attracts the points. If the weight function could be localized, allowing the points to be attracted to the highest weight function value in the area, then a large grid could be resolved without having to break it into zones.

Another area of research is to develop a method to tell a 'good' grid from a 'bad' one. With so much research in the area of grid generation this would be a useful tool. For the generation method presented in this thesis a method for testing the grid is needed to verify that this is the optimum grid for the given parameters. This investigation needs to be interfaced with a flow solver to verify an improvement in the time and solution on one of the adapted grids as opposed to the base grids.

BIBLIOGRAPHY

[1] Anderson, Dale A. "Adaptive Grid Methods for Partial Differential Equations." *Advances in Grid Generation*, Ed. Kirti N. Ghia and Urmila Ghia. The American Society of Mechanical Engineers, New York, 1983, 1-15.

[2] Halsey, N.D. "Use of Conformal Mapping in Grid Generation for Complex Three-Dimensional Configurations." *AIAA Journal*, 25, No. 10 (1987),1286-1291.

[3] Jacquotte, Olivier-P. "A Mechanical Model for a New Grid Generation Method in Computational Fluid Dynamics." *Computer Methods in Applied Mechanics and Engineering*, 66, (1988), 323-338.

[4] Thompson, J.F., Warsi, Z.U.A.,and Mastin, W.C. *Numerical Grid Generation, Foundations and Applications.* Elsevier/North-Holland, New York, 1985.

[5] Brackbill, J.U. "Coordinate System Control: Adaptive Meshes" *Numerical Grid Generation*, Ed. Joe Thompson. Elsevier/North-Holland, New York, 1982.

[6] Brackbill, J. and Saltzman, J. "Adaptive Zoning for Singular Problems in Two Dimensions." *Journal of Computational Physics*, 46, (1982), 342-368.

[7] Mortenson, M.E. *Geometric Modeling.* John Wiley and Sons, New York, 1985.

[8] Faux, I.D. and Pratt, M.J. *Computational Geometry for Design and Manufacture.* Ellis-Horwood, Chichester, England, 1980.

[9] Vinokur, M. "On One-Dimensional Stretching Functions for Finite Difference Calculations." NASA-CR-3313, Oct. 1980.

[10] Atwood, Christopher. "Geometric Modeling." Thesis, Iowa State University, Ames, Iowa, 1988.

[11] Thompson, Joe F. "Grid Generation Techniques in Computational Fluid Dynamics." *AIAA Journal*, 22, No. 11 (1984), 1505-1523.

[12] Hindman, R.G. "Static and Dynamic Grid Generation Course Notes." Unpublished notes. Department of Aerospace Engineering, Iowa State University, Ames, Iowa, 1985.

[13] Eisman, P.R. "Adaptive Grid Generation." *Computer Methods in Applied Mechanics and Engineering*, 64 (1987), 321-376.

[14] Chen, S.C. "Three-Dimensional Adaptive Grid Generation for a Boundary-Fitted Coordinate System." NASA-CR 182192, 1988.

[15] Flores, Jolen and Chaderjian, Neal "The Numerical Simulation of Transonic Flow about the Complete F-16A." AIAA-88-2506, 1988.

[16] Sorenson, Reese L. "The 3DGRAPE Book: Theory, User's Manual, Examples." NASA TM-102224.

ACKNOWLEDGEMENTS

The author would like to express her gratitude to Dr. Richard G. Hindman, whose guidence and encouragement was essential for the completion of this thesis.

The author also wishes to acknowledge the Applied Computational Fluids Branch of NASA Ames Research Center for financial support and direction in the early stages of this work.

This work has been supported under Joint Research Interchange NCA2-453.

APPENDIX . CALCULUS OF VARIATIONS

Calculus of variations is applied to equation 3.1 to produce equation 3.12, which is the basis for the adaptive grid generation scheme. The following equations show the manipulations in detail.

The calculations begin with the smoothness, orthogonality, and adaptivity terms from Chapter 3 in their integral equations.

$$I_s = \int_R [(\nabla\xi)^2 + (\nabla\eta)^2]\, dR = 0 \tag{A.1}$$

$$I_o = \int_R (\nabla\xi \cdot \nabla\eta)^2 J^3\, dR = 0 \tag{A.2}$$

$$I_v = \int_R (w^2 J)\, dR = 0 \tag{A.3}$$

(ξ, η) Computational Space

(u, v) Parametric Space

In the above equations (ξ, η) are the independent variables. As stated in Chapter 3, (u, v) are the preferred dependent variables. The equations are converted using the metrics.

$$\begin{aligned} \xi_u = \tfrac{v_\eta}{J} \quad \xi_v = \tfrac{-u_\eta}{J} \\ \eta_u = \tfrac{-v_\xi}{J} \quad \eta_v = \tfrac{u_\xi}{J} \end{aligned} \tag{A.4}$$

where J is the Jacobian,

$$J = u_\xi v_\eta - u_\eta v_\xi \tag{A.5}$$

The converted equations are the same as equations 3.4, 3.8, and 3.10 from Chapter Three.

$$\begin{aligned} I_s &= \int \frac{(u_\eta^2 + u_\xi^2 + v_\xi^2 + v_\eta^2)}{J} \, d\xi d\eta & \text{(A.6)} \\ I_o &= \int (u_\xi u_\eta + v_\xi v_\eta)^2 \, d\xi d\eta & \text{(A.7)} \\ I_v &= \int w^2 J^2 \, d\xi d\eta & \text{(A.8)} \end{aligned}$$

Because of the linearity, calculus of variations can be applied to each equation separately, then the resulting equations are combined linearly.

The Euler equations are

$$-\frac{\partial L}{\partial u} + \frac{\partial}{\partial \xi}\left(\frac{\partial L}{\partial u_\xi}\right) + \frac{\partial}{\partial \eta}\left(\frac{\partial L}{\partial u_\xi}\right) = 0 \tag{A.9}$$

$$-\frac{\partial L}{\partial v} + \frac{\partial}{\partial \xi}\left(\frac{\partial L}{\partial v_\xi}\right) + \frac{\partial}{\partial \eta}\left(\frac{\partial L}{\delta v_\xi}\right) = 0 \tag{A.10}$$

where L is the Lagrangian.

First, the contribution from the smoothness term is calculated.

$$I_s = \int \frac{(u_\eta^2 + u_\xi^2 + v_\xi^2 + v_\eta^2)}{J} \, d\xi d\eta \tag{A.11}$$

The Euler equation for u is

$$\frac{\partial}{\partial \xi}\left[\frac{2u_\xi J-(u_\eta^2+u_\xi^2+v_\xi^2+v_\eta^2)v_\eta}{J^2}\right] + \frac{\partial}{\partial \eta}\left[\frac{2u_\eta J+(u_\eta^2+u_\xi^2+u_\xi^2+v_\eta^2)v_\xi}{J^2}\right] = 0 \tag{A.12}$$

After the calculations the above equation can be written

$$\frac{2b\alpha}{J^3}u_{\xi\xi}+\frac{-4b\beta}{J^3}u_{\xi\eta}+\frac{2b\gamma}{J^3}u_{\eta\eta}+\frac{-2a\alpha}{J^3}v_{\xi\xi}+\frac{4a\beta}{J^3}v_{\xi\eta}+\frac{-2a\gamma}{J^3}v_{\eta\eta}=0 \tag{A.13}$$

The Euler equation for v is

$$\frac{\partial}{\partial \xi}\left[\frac{2v_\xi J+(u_\eta^2+u_\xi^2+v_\xi^2+v_\eta^2)u_\eta}{J^2}\right] + \frac{\partial}{\partial \eta}\left[\frac{2v_\eta J-(u_\eta^2+u_\xi^2+v_\xi^2+v_\eta^2)u_\xi}{J^2}\right] = 0 \tag{A.14}$$

After the calculations the above equation can be written

$$\frac{-2a\alpha}{J^3}u_{\xi\xi}+\frac{4a\beta}{J^3}u_{\xi\eta}+\frac{-2a\gamma}{J^3}u_{\eta\eta}+\frac{2c\alpha}{J^3}v_{\xi\xi}+\frac{-4c\beta}{J^3}v_{\xi\eta}+\frac{2c\gamma}{J^3}v_{\eta\eta}=0 \tag{A.15}$$

where the α, β, γ, a, b, and c are given by

$$\begin{aligned} a &= u_\xi v_\xi + u_\eta v_\eta & \alpha &= u_\eta^2 + v_\eta^2 \\ b &= v_\xi^2 + v_\eta^2 & \beta &= u_\xi u_\eta + v_\xi v_\eta \\ c &= u_\xi^2 + u_\eta^2 & \gamma &= u_\xi^2 + v_\xi^2 \end{aligned} \tag{A.16}$$

Now the orthogonality term is calculated.

$$I_o = \int (u_\xi u_\eta + v_\xi v_\eta)^2 \, d\xi d\eta \tag{A.17}$$

The Euler equation for u is

$$\frac{\partial}{\partial \xi}[2(u_\xi u_\eta + v_\xi v_\eta)u_\eta] + \frac{\partial}{\partial \eta}[2(u_\xi u_\eta + v_\xi v_\eta)u_\xi] = 0 \tag{A.18}$$

$$\begin{aligned} &4u_{\xi\eta}(u_\xi u_\eta + v_\xi v_\eta) + 2(u_\eta^2 u_{\xi\xi} + 2u_\xi u_\eta u_{\xi\eta} + u_\xi^2 u_{\eta\eta} \\ &+u_\eta v_\eta v_{\xi\xi} + (u_\eta v_\xi + u_\xi v_\eta)v_{\xi\eta} + u_\xi v_\xi v_{\eta\eta}) \end{aligned} \tag{A.19}$$

The Euler equation for v is

$$\frac{\partial}{\partial \xi}[2(u_\xi u_\eta + v_\xi v_\eta)v_\eta] + \frac{\partial}{\partial \eta}[2(u_\xi u_\eta + v_\xi v_\eta)v_\xi] = 0 \tag{A.20}$$

$$\begin{aligned} &4v_{\xi\eta}(u_\xi u_\eta + v_\xi v_\eta) + 2(v_\eta^2 v_{\xi\xi} + 2v_\xi v_\eta v_{\xi\eta} + v_\xi^2 v_{\eta\eta} \\ &+u_\eta v_\eta u_{\xi\xi} + (u_\eta v_\xi + u_\xi v_\eta)u_{\xi\eta} + u_\xi v_\xi u_{\eta\eta}) \end{aligned} \tag{A.21}$$

Now, the adaptive term is calculated.

$$I_v = \int w^2 \, J^2 \, d\xi d\eta \; = \; 0 \tag{A.22}$$

The Euler equation for u is

$$-\, 2ww_u J^2 + \frac{\partial}{\partial \xi}\left[w^2 2Jv_\eta\right] + \frac{\partial}{\partial \eta}\left[-w^2 2Jv_\xi\right] = 0 \tag{A.23}$$

$$\begin{aligned} = \quad & -2ww_uJ^2 + 2w^2v_\eta^2u_{\xi\xi} - 4w^2v_\eta v_\xi v_{\xi\eta} + 2w^2v_\xi^2u_{\eta\eta} \\ & -2w^2u_\eta v_\eta v_{\xi\xi} + 2w^2(v_\eta u_\xi + v_\xi u_\eta)v_{\xi\eta} - 2w^2v_\xi u_\xi v_{\eta\eta} \end{aligned} \tag{A.24}$$

Since the ξ, η are the independent variables w_u is converted using the metrics.

$$-2ww_uJ^2 = -2wJ(w_\xi v_\eta - w_\eta v_\xi) \tag{A.25}$$

The Euler equation for v is

$$-2ww_vJ^2 + \frac{\partial}{\partial\xi}\left[-w^2 2Ju_\eta\right] + \frac{\partial}{\partial\eta}\left[w^2 2Ju_\xi\right] = 0 \tag{A.26}$$

$$\begin{aligned} = \quad & -2ww_vJ^2 + 2w^2u_\eta^2v_{\xi\xi} - 4w^2u_\eta u_\xi v_{\xi\eta} + 2w^2u_\xi^2v_{\eta\eta} \\ & -2w^2u_\eta v_\eta u_{\xi\xi} + 2w^2(v_\eta u_\xi + v_\xi u_\eta)u_{\xi\eta} - 2w^2v_\xi u_\xi u_{\eta\eta} \end{aligned} \tag{A.27}$$

Since the ξ, η are the independent variables w_v is converted using the metrics.

$$-2ww_vJ_2 = -2wJ(-w_\xi u_\eta + w_\eta u_\xi) \tag{A.28}$$

All the equations are combined. Using matrix notation the final equation can be written:

$$A\mathrm{r}_{\xi\xi} + B\mathrm{r}_{\xi\eta} + C\mathrm{r}_{\eta\eta} + D\mathrm{r}_\xi + E\mathrm{r}_\eta = 0 \tag{A.29}$$

with

$$a = u_\xi v_\xi + u_\eta v_\eta \qquad \alpha = u_\eta^2 + v_\eta^2$$
$$b = v_\xi^2 + v_\eta^2 \qquad \beta = u_\xi u_\eta + v_\xi v_\eta$$
$$c = u_\xi^2 + u_\eta^2 \qquad \gamma = u_\xi^2 + v_\xi^2$$

$$A_s = \begin{pmatrix} b & -a \\ -a & c \end{pmatrix} \tfrac{2\alpha}{J^3} \quad B_s = \begin{pmatrix} -2b & 2a \\ 2a & -2c \end{pmatrix} \tfrac{2\beta}{J^3} \quad C_s = \begin{pmatrix} b & -a \\ -a & c \end{pmatrix} \tfrac{2\gamma}{J^3}$$

$$A_o = \begin{pmatrix} u_\eta^2 & v_\eta x_\eta \\ v_\eta u_\eta & v_\eta^2 \end{pmatrix} 2\lambda_o \quad B_o = \begin{pmatrix} 2(\beta + u_\eta u_\xi) & v_\eta u_\xi + u_\eta v_\xi \\ v_\eta u_\xi + u_\eta v_\xi & 2(\beta + v_\xi v_\eta) \end{pmatrix} 2\lambda_o$$

$$C_o = \begin{pmatrix} u_\xi^2 & u_\xi v_\xi \\ u_\xi v_\xi & v_\xi^2 \end{pmatrix} 2\lambda_o$$

$$A_v = \begin{pmatrix} v_\eta^2 & -v_\eta u_\eta \\ -v_\eta u_\eta & u_\eta^2 \end{pmatrix} 2w^2\lambda_v \quad B_v = \begin{pmatrix} -2v_\xi v_\eta & v_\eta u_\xi + u_\eta v_\xi \\ v_\eta u_\xi + u_\eta v_\xi & -2u_\xi u_\eta \end{pmatrix} 2w^2\lambda_v$$

$$C_v = \begin{pmatrix} v_\xi^2 & -u_\xi v_\xi \\ -u_\xi v_\xi & u_\xi^2 \end{pmatrix} 2w^2\lambda_v$$

$$D_v = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} 2Jww_\eta\lambda_v \quad E_v = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} 2Jww_\xi\lambda_v$$

where

$$\begin{aligned}
A &= A_s + A_o + A_v \\
B &= B_s + B_o + B_v \\
C &= C_s + C_o + C_v \\
D &= D_v \\
E &= E_v
\end{aligned} \tag{A.30}$$