A COLLECTION OF TECHNICAL PAPERS PART 2

AIAA 8th APPLIED AERODYNAMICS CONFERENCE

AUGUST 20–22, 1990 / PORTLAND, OREGON

GRID GENERATION USING A POSTERIORI OPTIMIZATION WITH GEOMETRICALLY NORMALIZED FUNCTIONALS

AIAA-90-3048-CP

Brett W. Siebert* and George S. Dulikravich **
Penn State University, Department of Aerospace Engineering
University Park, PA 16802, USA

Abstract

An iterative a posteriori grid optimization procedure is presented. Grid quality is measured by efficient non-orthogonality, non-smoothness, and clustering functionals that are normalized using geometric considerations to provide for the combination of mutually incompatible characteristics in the final grid. Proper formulation and normalization of these functionals yields a scheme which is capable of creating non-overlapping grids in concave to convex regions. An additional functional which aids in untangling already overlapped grids is presented. Applications to solution adaptive grids are presented and the cost effectiveness of the procedure versus an elliptic grid generator is evaluated.

Introduction

The vast variety of grid generation techniques is evidence that the method of developing an acceptable computational grid is problem dependent. In order to provide a more robust grid generation package it is desirable to be able to treat a particular grid a posteriori so as to improve its usefulness. This report presents improvements to an optimization method[1-5] for structured two-dimensional grids which is independent of the initial grid generation algorithm. Expanding to three dimensions can be accomplished by a development analogous to the one given here[3]. The goal of grid optimization is to maximize a certain beneficial feature, or combination of features, possessed by the grid. A prerequisite for upgrading the grid via numerical means is the ability to quantify these features. One conventional measure of grid quality is orthogonality. A grid with nearly orthogonal grid lines produces less numerical diffusion than a highly skewed one. It is also advantageous to have a grid of smoothly varying grid cell sizes. This prevents relatively large Taylor series truncation errors. Concentrating grid points in regions of high variable gradients is necessary to capture the true behavior of the field solution. We will present improved formulations for the functionals that quantify these characteristics. Since different grid qualities are not necessarily mutually compatible, it is necessary to combine them in a weighted fashion. The grid points are iteratively relocated via an inexpensive directional gradient technique in such manner as to minimize the net functional. Application of optimization to solution adaptive [4], block-structured [5-7], and non-structured grid generation [8] was investigated in the past.

Nomenclature

Formed by connecting line segments between the central grid point and its immediate neighbors, the master grid cell serves as the measuring point of grid quality. The following notation is used. A single subscript ($_{E,W,N,S,O}$) denotes quantities at the neighboring grid points to the east $(i+1,j)$, west $(i-1,j)$, north $(i,j+1)$, or south $(i,j-1)$ with respect to the central point (i,j). It can also refer to the line connecting the central point to a neighboring point or a quantity associated with that line. These segments are referred to as cell rays. Subscripts such as ($_{EN,NW,WS,SE}$) correspond to the quadrant or subcell in the specified direction, or to some operation between the indicated cell rays. These vectors, their magnitudes, and supplementary relationships are defined as follows:

$$\vec{r}_E = (x_{i+1,j} - x_{i,j}, y_{i+1,j} - y_{i,j}) = (X_E, Y_E) \quad (2.1a)$$
$$\vec{r}_N = (x_{i,j+1} - x_{i,j}, y_{i,j+1} - y_{i,j}) = (X_N, Y_N) \quad (2.1b)$$
$$\vec{r}_W = (x_{i-1,j} - x_{i,j}, y_{i-1,j} - y_{i,j}) = (X_W, Y_W) \quad (2.1c)$$
$$\vec{r}_S = (x_{i,j-1} - x_{i,j}, y_{i,j-1} - y_{i,j}) = (X_S, Y_S) \quad (2.1d)$$

$$S_E = \vec{r}_E \cdot \vec{r}_E = X_E^2 + Y_E^2 \quad (2.2a)$$
$$S_N = \vec{r}_N \cdot \vec{r}_N = X_N^2 + Y_N^2 \quad (2.2b)$$
$$S_W = \vec{r}_W \cdot \vec{r}_W = X_W^2 + Y_W^2 \quad (2.2c)$$
$$S_S = \vec{r}_S \cdot \vec{r}_S = X_S^2 + Y_S^2 \quad (2.2d)$$

$$D_{EN} = \vec{r}_E \cdot \vec{r}_N = X_E X_N + Y_E Y_N \quad (2.3a)$$
$$D_{NW} = \vec{r}_N \cdot \vec{r}_W = X_N X_W + Y_N Y_W \quad (2.3b)$$
$$D_{WS} = \vec{r}_W \cdot \vec{r}_S = X_W X_S + Y_W Y_S \quad (2.3c)$$
$$D_{SE} = \vec{r}_S \cdot \vec{r}_E = X_S X_E + Y_S Y_E \quad (2.3d)$$

$$C_{EN} = A_{EN} = |\vec{r}_E \times \vec{r}_N| = X_E Y_N - X_N Y_E \quad (2.4a)$$
$$C_{NW} = A_{NW} = |\vec{r}_N \times \vec{r}_W| = X_N Y_W - X_W Y_N \quad (2.4b)$$
$$C_{WS} = A_{WS} = |\vec{r}_W \times \vec{r}_S| = X_W Y_S - X_S Y_W \quad (2.4c)$$
$$C_{SE} = A_{SE} = |\vec{r}_S \times \vec{r}_E| = X_S Y_E - X_E Y_S \quad (2.4d)$$

Non-orthogonality Functional

A truly orthogonal master cell possesses right angles between consecutive cell rays. To quantify the degree of non-orthogonality in a cell, the most obvious and computationally inexpensive method is to take the dot product between each pair of consecutive rays [1-5]. Because it is possible for only three of the rays to be perpendicular, it is necessary to use a summation over all four subcells to fully describe local non-orthogonality. To prevent negative dot products from diminishing the measure of non-orthogonality, the squares of the dot products are used, thereby making each departure from orthogonality cumulative upon the functional.

$$F_O = (D_{EN})^2 + (D_{NW})^2 + (D_{WS})^2 + (D_{SE})^2 \quad (3.1)$$

The purpose of F_O is to measure solely the degree of non-orthogonality, but as the dot product is equal to the product of the cell ray magnitudes and the cosine of the included angle, Eq. 3.1 is biased in favor of large subcells. To remove this intrusion of cell size into the non-orthogonality functional, each scalar product is multiplied by the square of the magnitude of the remaining cell rays.

$$F_O = S_W S_S (D_{EN})^2 + S_S S_E (D_{NW})^2 + S_E S_N (D_{WS})^2 + S_N S_W (D_{SE})^2 \quad (3.2)$$

In order to combine various grid characteristics in a meaningful way, it is necessary to normalize each functional so that a linear combination of component functionals with prescribed weighting factors can be made to form the net expression. Neglecting the normalization results in a failure of the weighting factors to combine the qualities in the desired fashion. Kennon and Dulikravich [2] have suggested that the functional be computed at each point in the grid, then the resulting distribution be scanned for the largest value, which in turn should be used as the nor-

* Graduate Student. Student Member AIAA
** Associate Professor. Senior Member AIAA

malization factor. While this method does normalize the functional, using it in the optimization process results in a failure to benefit from relative improvements among grid qualities as the iterative solution progresses. For example, under such a treatment, F_O will always range from 0 to 1 regardless of how asymptotically close to true orthogonality the grid becomes. Simultaneously, a similarly treated non-smoothness functional will also be in the same range even for a highly uneven cell size distribution. Therefore, a great deal of computational time is wasted in an attempt to adjust the grid to eliminate the non-orthogonality residuals on a nearly orthogonal grid, instead of the bulk of the effort being expended to improve poor cell size distribution.

Geometric considerations can be used to find a more appropriate normalization factor. Rewriting the dot products in Eq. 3.2 in terms of ray size and included angle yields:

$$F_O = S_W S_S S_E S_N \cos^2\theta_{EN} + S_S S_E S_N S_W \cos^2\theta_{NW} + S_E S_N S_W S_S \cos^2\theta_{WS} + S_N S_W S_S S_E \cos^2\theta_{SE} \quad (3.3)$$

The factor $S_E S_N S_W S_S$ is common to each term. At worst the sum $\sum_i^{E,W,N,S} \cos^2\theta_i$ can attain a maximum value of 4. Therefore, the term $4S_E S_N S_W S_S$ is used as the normalization factor. In order to avoid the computational expense of computing the subtended angles and the corresponding cosines, the equation is re-expressed in terms of the nodal coordinates:

$$F_O = \frac{S_W S_S (D_{EN})^2 + S_S S_E (D_{NW})^2}{4(S_E S_N S_W S_S)} + \frac{S_E S_N (D_{WS})^2 + S_N S_W (D_{SE})^2}{4(S_E S_N S_W S_S)} \quad (3.4)$$

Non-smoothness Functional

Grid smoothness is most readily described in terms of the relative areas of neighboring subcells. A grid is considered to be perfectly smooth if the area of one subcell is equal to that of its neighbors. Subcell areas can be measured exactly by taking the cross products of the boundary vectors, but this would result in the added expense of bringing four additional points into the functional calculation. The coordinates of the grid points to the northeast, northwest, southeast, and southwest were ignored while quantifying the degree of non-orthogonality, but for an exact smoothness comparison it is necessary to include them. This leads to a more tedious calculation, so previous investigators [1,2,3,5] have suggested that a particular subcell area be approximated by the absolute value of the cross product of the cell rays which lie along its boundary. For the case of subcells forming parallelograms, this approximation becomes exact, but for arbitrary quadrilateral subcells the accuracy diminishes as the lengths of opposing sides differ, and as the sum of any two neighboring interior angles varies from π. Optimization makes the error in area measure self-regulating, in that the non-smoothness functional tends to lessen the occurrence of widely varying lengths across a subcell and the non-orthogonality term reduces the tendency of consecutive angle sweep to vary from π.

In an effort to prevent grid overlap in adaptive unstructured grids, Kennon [8] has proposed using the signed cross product as the area measure. This minor change is adopted here to help formulate a geometric normalization for the non-smoothness functional, while retaining the tendency to prevent grid overlap. Eqs. 2.4 show the formula for calculating the cross product, and its magnitude can be re-expressed as:

$$|\vec{r}_A \times \vec{r}_B| = r_A r_B \sin\theta_{AB} \quad (4.1)$$

where A and B are any two neighboring cell rays. If the subtended angle, as measured in the counter-clockwise direction, is less than π, the magnitude of the cross product and thus the estimated area will be positive. However, when the angle exceeds this value, the $\sin\theta$ factor makes the area negative. As long as all included angles are less than π, the grid lines cannot overlap. The new non-smoothness functional with given area measures is shown below.

$$F_S = \frac{|(A_{EN} + A_{NW}) - (A_{WS} + A_{SE})|}{2|A_{EN} + A_{NW} + A_{WS} + A_{SE}|} + \frac{|(A_{EN} + A_{SE}) - (A_{NW} + A_{WS})|}{2|A_{EN} + A_{NW} + A_{WS} + A_{SE}|} \quad (4.2)$$

Only cells possessing a common cell ray are compared in the functional. Incorporating an additional term which compares diagonal cell areas (i.e. $(A_{EN}+A_{WS}),(A_{NW}+A_{SE})$) is unnecessary, since area equality between diagonal cells is assured when the value of the presented form of functional vanishes. Using the form given in Eq. 4.2 has several advantages. First, only two comparisons, the areas above (A_{EN}, A_{NW}) and below (A_{SE}, A_{WS}) the horizontal division of the master cell and the areas to the left (A_{NW}, A_{WS}) and the right (A_{EN}, A_{SE}) of the vertical division, are made in such a manner as to insure equality between all four subcell areas. This is more efficient than making a cell by cell comparison as done in [1-5]. Secondly, overlap is prevented through the use of the given normalization scheme. Since two consecutive angles between the cells cannot each be greater than π, and only cells possessing a common boundary are combined, the consequence of using the signed cross product is that overlapping cells tend to make the functional larger and thus merit greater local optimization. The change in area sign makes both terms in the numerator involving the negative area larger, while decreasing the value of the denominator. Obviously, the functional cannot exceed unity when all cell areas are greater than zero, but when overlap occurs the negative subcell area that results from the crossover can make the functional very large. This insures a greater success in correcting the grid through the optimization process during which the functional is driven to its minimum value. One other advantage of using the present formulation of F_S is that the form is very similar to those of the clustering and gradient functionals (see below), so the area combinations need be calculated only once for all three functionals.

Unlike F_O, which was normalized in such a way that subcell size was not a contributing factor to the magnitude of the functional, the non-smoothness cannot, under the approximations adopted, exhibit independence from cell orthogonality. But the negative effects of this coupling are outweighed by the ability to quantify this grid quality using the same points as the former functional while exhibiting a strong tendency to prevent overlap. Further, equating the subcell area to the cross product of the cell rays is less than half as computationally expensive as calculating the exact area measure.

Field Gradient Functional

For a given flow field variable ϕ, for example Mach number, it is desired that the grid points be concentrated in regions where its gradients are large. Using the existing grid a distribution of ϕ, be it a fully or partially converged field, is found. The values of its spatial gradients are then taken. In clustering a two dimensional grid, equal consideration should be given to derivatives in the x and y directions and each quantity should be cumulative upon the functional. As a measure of the magnitude of the gradient, the following expression is used:

$$G = \sqrt{\left(\frac{\partial\phi}{\partial x}\right)^2 + \left(\frac{\partial\phi}{\partial y}\right)^2} \quad (5.1)$$

These derivatives are best evaluated by mapping the grid into a computational domain where each grid cell becomes a square with unit spacing between the grid points. Eqs. 5.2 and 5.3 give the relationships for evaluating the derivatives:

$$\frac{\partial \phi}{\partial x} = \frac{\partial \xi}{\partial x}\frac{\partial \phi}{\partial \xi} + \frac{\partial \eta}{\partial x}\frac{\partial \phi}{\partial \eta}; \qquad \frac{\partial \phi}{\partial y} = \frac{\partial \xi}{\partial y}\frac{\partial \phi}{\partial \xi} + \frac{\partial \eta}{\partial y}\frac{\partial \phi}{\partial \eta} \tag{5.2}$$

$$\frac{\partial \xi}{\partial x} = \frac{1}{J}\frac{\partial y}{\partial \eta}; \qquad \frac{\partial \xi}{\partial y} = \frac{-1}{J}\frac{\partial x}{\partial \eta} \tag{5.3a}$$

$$\frac{\partial \eta}{\partial x} = \frac{-1}{J}\frac{\partial y}{\partial \xi}; \qquad \frac{\partial \eta}{\partial y} = \frac{1}{J}\frac{\partial x}{\partial \xi} \tag{5.3b}$$

$$J = \frac{\partial x}{\partial \xi}\frac{\partial y}{\partial \eta} - \frac{\partial x}{\partial \eta}\frac{\partial y}{\partial \xi} \tag{5.3c}$$

If more than one flow quantity is required to represent the areas of high gradients, they can be incorporated as follows (ψ representing an additional field variable):

$$G = \sqrt{\left(\frac{\partial \phi}{\partial x}\right)^2 + \left(\frac{\partial \phi}{\partial y}\right)^2} + \sqrt{\left(\frac{\partial \psi}{\partial x}\right)^2 + \left(\frac{\partial \psi}{\partial y}\right)^2} \tag{5.4}$$

With G calculated at each grid point, it is possible to combine the quantities across a cell ray, so that regardless of which grid point at its endpoints is under consideration, the gradients will contribute equally to both. The following relationships are used to ensure the "conservative" coupling of the gradient measures between the grid points:

$$k_E = G_O + G_E; \qquad k_N = G_O + G_N \tag{5.5a}$$

$$k_W = G_O + G_W; \qquad k_S = G_O + G_S \tag{5.5b}$$

Large values of k_i indicate regions of high flow gradients, or regions where the grid points should be concentrated. These quantities can be thought of as stiffness coefficients of springs connecting the grid points. Previous investigators have formulated functionals which tend to minimize the potential energy in the spring system [3,4,5]. That type of formulation suffers from the drawback that the functional never goes to zero. While relocation techniques do acknowledge the existence of non-zero minima in the functional, it is more convenient to have a functional with a minimum at zero to prevent floating point errors from contaminating the net functional. The formulation used in this report is equivalent to specifying equal spring energies in each half of the master cell. The functional goes to zero as the energies become equivalent. After scaling the values of k_i by the maximum value of all k_i's, the following expression is used as the normalized field gradient clustering functional:

$$F_G = \frac{|k_N(A_{EN} + A_{NW}) - k_S(A_{WS} + A_{SE})|}{2|A_{EN} + A_{NW} + A_{WS} + A_{SE}|} + \frac{|k_E(A_{EN} + A_{SE}) - k_W(A_{NW} + A_{WS})|}{2|A_{EN} + A_{NW} + A_{WS} + A_{SE}|} \tag{5.6}$$

If all k_i's were unity, then this functional would be equivalent to F_S. Since the areas contain the length of the cell rays, the grid points are located in such a manner as to have small separation for large stiffness coefficients. However, because it is the areas that are multiplied by the stiffness coefficients, rather than the non-negative cell ray length as previous researches have suggested[3,4,5], grid overlap can be more readily prevented. In theory, once the grid points are relocated, new values of ϕ could be interpolated from the old grid, but this would be exceedingly expensive. Taken in conjunction with the fact that adaptive grids are used with iterative solvers, it is sufficient to assume that k_i does not change during the relocation process.

Clustering Functional

In some instances, it can be deduced a priori where the regions of high flow gradients will occur. Judicious use of this knowledge can save the expense incurred by using adaptive grids. Using the same reasoning applied in the formulation of F_G, the clustering functional incorporating a user specified distribution of stiffness coefficients, c_i, is given in Eq. 6.1. The proper distribution of c_i is problem dependent.

$$F_C = \frac{|c_N(A_{EN} + A_{NW}) - c_S(A_{WS} + A_{SE})|}{2|A_{EN} + A_{NW} + A_{WS} + A_{SE}|} + \frac{|c_E(A_{EN} + A_{SE}) - c_W(A_{NW} + A_{WS})|}{2|A_{EN} + A_{NW} + A_{WS} + A_{SE}|} \tag{6.1}$$

Non-overlapping Functional

After investigating the capability of these functionals to treat grid overlap, it became apparent that they adequately prevent it, but are ineffective in removing already existing overlaps. The reason for this lies in the fact that the normalized functionals can actually increase as the grid approaches non-overlap. As previously mentioned, negative subcell area is indicative of grid overlap. This provides the key for formulating a functional which vanishes when overlap is eliminated, but unlike the other functionals, automatically decreases as overlap diminishes:

$$F_N = (|A_{EN}| - A_{EN}) + (|A_{NW}| - A_{NW}) + (|A_{SE}| - A_{SE}) + (|A_{WS}| - A_{WS}) \tag{7.1}$$

Notice that F_N, the negative area or non-overlapping functional, is zero for a master cell which is free from overlap, and is positive but diminishing for a cell which is overlapped but moving towards non-overlap as evidenced by shrinking negative subcell area. Because F_N vanishes on an optimized grid, normalization is not required. When incorporated into the net functional, the unique behavior of this functional and the absolute importance of removing any existing overlap require that it be treated differently from the other component functionals, as discussed in the following section.

Net Functional

Now that the component functionals have been defined, they are incorporated into a net functional, F_{NET}, defined as the linear combination of the individual functionals:

$$F_{NET} = \begin{cases} \omega_O F_O + \omega_S F_S + \omega_G F_G + \omega_C F_C, & \text{if } F_N = 0; \\ 1 + F_N, & \text{if } F_N > 0. \end{cases} \tag{8.1}$$

Here, ω_i act as weighting factors, allowing incompatible qualities to coexist in the converged grid. Weighting factors are not strictly limited in size, but it is easier to define relative qualities if the sum of all of the factors is unity. The formulation of F_{NET} is such that if any local overlap exists, it is removed before any other grid qualities are treated. This is as it should be, since an overlapped grid is useless to numerical solvers, regardless of the other qualities. By adding unity to the F_N and setting the sum of all weighting factors to one, F_{NET} decreases as the overlap is removed, and after the crossover has been eliminated it is less than it was with the overlap regardless of the quality as measured by the other component functionals.

Of course other characteristics may be desirable in the computational grid and their corresponding functionals can be linearly added to supplement F_{NET}. These new functionals may not have physically meaningful normalization factors, but the method of scanning the field and

normalizing it with respect to the largest value will suffice. The weighting factors should be assigned with this in mind. If, for instance, a component functional based on the residuals of the governing field equations [9] is included but not normalized, it may dominate F_{NET}, yielding a grid inappropriate for accurate evaluation of the discretized field equations.

Minimization Scheme

The task of altering the grid so as to minimize the net functional is accomplished by using an inexpensive directional gradient method. The first step in the grid relocation algorithm is to evaluate the net functional and its gradient at the grid point. Due to the complexity of the functionals, their derivatives with respect to the spatial variables cannot be conveniently evaluated analytically. The limit definition of the derivative provides an inexpensive yet accurate method:

$$\frac{\partial F_{NET}}{\partial x} = \frac{F_{NET}(x+\epsilon, y) - F_{NET}(x, y)}{\epsilon} \tag{9.1a}$$

$$\frac{\partial F_{NET}}{\partial y} = \frac{F_{NET}(x, y+\epsilon) - F_{NET}(x, y)}{\epsilon} \tag{9.1b}$$

ϵ is a small number ($\sim 10^{-6}$) assuming that the global dimensions of the entire grid have been used for coordinate scaling. Once the gradient is known, the central grid point is temporarily relocated according to the following relationships:

$$x = x - \alpha \frac{\partial F_{NET}}{\partial x} \tag{9.2a}$$

$$y = y - \alpha \frac{\partial F_{NET}}{\partial y} \tag{9.2b}$$

F_{NET} is reevaluated at this new location. If the net functional decreases, then the multiplicative constant α is repeatedly doubled until F_{NET} begins to increase. The grid point is then moved to the location corresponding to the lowest F_{NET}. If the net functional originally increases, then α is repeatedly halved until the net functional drops below the initial value. The central grid point is then moved to the location corresponding to this α. In order to accelerate the computation of the proper α, the value from the previous iteration serves as a starting point for the current iteration. F_{NET} is not precisely minimized during any one sweep of the grid, but this is not necessary. The added expense of optimally repositioning the central grid point with respect to neighboring grid points, that will in turn be relocated, is prohibitively expensive and inefficient. This simple relocation technique does not affect the final shape of the grid, only the amount of time and number of outer loop iterations it takes to get there. Rather than updating the nodal locations of the entire grid at once, each grid point is moved as soon as a better location is found. This reduces memory requirements and decreases computational expense, since some grid points will have the benefit of being relocated with respect to more optimally placed neighbors. For grids of initially very poor quality, it is more efficient to relocate the grid points in a manner in which large increments can be taken. Bypassing the expense of finding the proper α and gradients of F_{NET}, the field can be swept a few times using the average coordinates of its neighbors to relocate the central grid point:

$$x_{avg} = (x_{i+1,j} + x_{i,j+1} + x_{i-1,j} + x_{i,j-1})/4 \tag{9.3a}$$

$$y_{avg} = (y_{i+1,j} + y_{i,j+1} + y_{i-1,j} + y_{i,j-1})/4 \tag{9.3b}$$

If the net functional is lower at (x_{avg}, y_{avg}), then the grid point is moved, otherwise it is left untouched. This technique conditions the initial grid, resulting in faster optimization. Sweep direction is reversed after each pass to prevent directional biasing due to the frontal characteristic of this algorithm

Boundary Points

A proper master cell does not exist at the boundaries. To overcome this problem, previous researchers have proposed creating a master cell by generating imaginary points outside the computational domain [1-3]. A different approach is taken in this report. At a boundary, only three neighboring grid points are present, thus creating two subcells. Boundary grid point component functionals are formulated in the same manner as their full master cell counterparts, except only two subcells are utilized. This is both cheaper and more accurate than dealing with imaginary points. The boundary functionals making only one comparison between the two existing areas can become quite large during overlap, as there are no other cell areas to compensate for the one that is negative.

When accurate application of boundary conditions require the grid lines to meet the boundaries at right angles, orthogonality locally overshadows the importance of the other qualities. By having a separate set of weighting factors for the boundary points (indeed it is possible to have a different set of weighting factors for each grid point), in which the orthogonality functional is heavily emphasized, this criteria can be conveniently met. This does, however, slow down the relocation of the boundary points, as any shift in these points is closely tied to the location of the interior layer of grid points.

Not only are the interior and boundary grid points treated differently, but different types of boundaries must be handled appropriately. For a free floating boundary, in which the points need not remain on a particular curve, no special attention is merited. At boundaries which are defined by a particular, curve a parametric spline curve is fit through it and the point is relocated according to the negative dot product of the functional gradient and the tangent to the curve. On a periodic boundary one contour is optimized as needed, and the other one set so as to maintain periodicity. In the case of boundary slope discontinuity, grid points pivot about the discontinuity. If grid quality is improved by sliding the grid point on either side of the pivot into the discontinuity and the original discontinuity grid point half way to the successive point, then the grid is updated. The corner grid points remain fixed so that the shape of the boundary is not distorted by the optimization process.

Results

A comparison of the optimization technique versus an elliptic grid generator was made. Figs. 1 and 2 show the initial, optimized, and elliptically generated grids for sizes of 20×20 and 40×40. The quality of the optimized grids, with $\omega_O = 0.45$ and $\omega_S = 0.55$, is equivalent to that of the elliptically generated grids. Only the coarsest optimized grid is different from the elliptic one, and this only near the corner. This is because the boundary points on the optimized grids were kept fixed in order to provide a fair time comparison against the elliptic grid generator which is incapable of relocating the points on the boundary. Fig. 3 shows the cost per grid cell, using a VAX 8550, of generating each of the final grids as a function of the total number of grid cells. The cost of the optimizer is twice that of the elliptic grid generator; this represents 12 averaging sweeps and 12 optimization sweeps for the 20×20 grid and 14 averaging and 8 optimization sweeps on the finer grid. The elliptic scheme requires a memory capacity on the order of $4N$, N being the total number of grid points. The optimizer requires $10N$ storage. However, all but $3N$

come about through the use of the clustering and gradient functionals which have no counterpart in the elliptic scheme that was used.

The robustness of a posteriori optimization is demonstrated in Fig. 4. It shows the results of dealing with existing overlap. The initial grid, Fig. 4a, covers a domain which goes from concave to convex spanning a boundary which has a discontinuity in the slope. Conventional grid generation techniques have trouble preventing overlap in such a domain. Without a means of correcting this, the grids they generate are useless. Optimization prevents this problem. The optimizer produces the grid shown in Fig. 4b after 10 cycles of 4 averaging and 10 optimization sweeps each. A cycle is merely a reinitialization of the update factors, α, to a value of one in order to prevent premature stalling in regions of local minima. All of the overlap has been removed, and the grid lines follow the tip smoothly. Equal weighting was placed on orthogonality and smoothness. The grid points were permitted to slide along each of the boundaries, except on the lower edge where the discontinuity could not be accurately spline fit at this grid size. When the overlapped grid was sent into the elliptic grid generator the grid remained overlapped as shown in Fig. 4c.

Grid optimization was also applied to achieve a solution adaptive grid. Fig. 5a shows the initial grid about a bump in a channel. A steady state compressible Euler solver[10] was applied to the grid to solve the flow in a transonic regime. After 200, 500, 1000, and 2000 iterations of the Euler solver, the grid was sent into the optimization code with Mach number serving as the ϕ field in the gradient clustering stiffness coefficients. Equal weighting was given to F_O and F_G, and each optimization process entailed 4 cycles of 8 averaging and 8 optimization sweeps. Fig. 5b shows the clustering about the developing shock wave after 500 iterations. Fig 5c. shows the converged grid at 2000 iterations. At this point the shock is fully developed and dramatic changes in the solution no longer occur. Fig. 6 shows an enlarged region of the grid near the trailing edge shock wave. While the points are concentrated about the shock and towards the bump, the leading and trailing edges are poorly represented. This is due to the inadequacy of the Mach number alone to represent all the regions of high flow gradients. Figs. 7a and 7b show Mach contours when the flow was converged using solution adaptive and fixed grids. The solution adaptive grid has a sharper shock wave due to the better resolution provide by the clustering of points in that region. Fig. 8 shows a comparison of convergence rates between the solution adaptive and fixed grids. Increases in the residual on the solution adaptive grid occur because the bilinear interpolation of the flow solutions from the old to optimized grid cannot be accomplished accurately enough in the high gradient regions near the leading and trailing edges and at the shock. This is demonstrated in Fig. 9, which is a contour plot of the two highest orders of magnitude of the residual after the 2000 iteration optimization process. The time requirements for the grid optimization and solution interpolation are negligible compared to those of the Euler solver.

Conclusions

The proposed grid optimization component functionals are more efficient and more reflective of the geometric quantification of grid quality than those given by previous investigators. Overlap is successfully prevented and eliminated by using the presented normalization factors and supplemental functional, thus enhancing the robustness of a posteriori grid optimization for complex configurations. Solution adaptive grids can also be generated using a posteriori optimization.

Acknowledgements

We would like to acknowledge the contributions of Peter Grul for generating some of the initial grids, James Withington for the use of his elliptic grid generation code, and Dr. Seungsoo Lee for the use of his Euler solver.

References

1. Kennon, S.R., and Dulikravich, G.S., *A Posteriori Optimization of Computational Grids*, AIAA Paper 85-0483, 25th Aerospace Sciences Meeting, Reno, NV, January 1985.
2. Kennon, S.R., and Dulikravich, G.S., *Generation of Computational Grids Using Optimization*, AIAA Journal, Vol. 24, No. 7, July 1986, pp. 1069-73.
3. Carcaillet, R., Kennon, S.R., and Dulikravich, G.S., *Optimization of Three-dimensional Computational Grids*, AIAA Journal of Aircraft, Vol. 23, No. 5, May 1986, pp. 415-421.
4. Carcaillet, R., Dulikravich, G.S., and Kennon, S.R., *Generation of Solution Adaptive Computational Grids Using Optimization*, Computer Meth. in App. Mech. and Eng., Vol. 57, Sept. 1986, pp. 279-295.
5. Kennon, S.R., and Dulikravich, G.S., *Composite Computational Grid Generation Using Optimization*, Proceedings of the First International Conference on Numerical Grid Generation in CFD, ed. J. Haeuser, Landshut, West Germany, July 14-17, 1986.
6. Martin, C.W., Soni, B.K., and McClure, M.D., *Experience in Grid Optimization*, AIAA Paper 87-0201, Reno, NV, January 1987.
7. Mani, K.K., *Generation of Three-dimensional Computational Grids Independent of Topology*, AIAA Paper 87-0274, 27th Aerospace Sciences Meeting, Reno, NV, January 1987.
8. Kennon, S.R., *Supersonic Inlet Calculations Using an Upwind Finite-Volume Method on Adaptive Unstructured Grids*, AIAA Paper 89-0113, 27th Aerospace Sciences Meeting, Reno, Nevada, January, 1989.
9. Demkowicz, L., and Oden, J.T., *On a Mesh Optimization Method Based on a Minimization of Interpolation Error*, Int. J. Eng. Sci., Vol. 24, No. 1, 1986, pp. 55-68.
10. Lee, S., *Acceleration of Iterative Algorithms for Euler and Navier-Stokes Equations*, Ph.D. Dissertation, Dept. of Aerospace Engineering, Penn State University, University Park, PA, May, 1990.

Figures

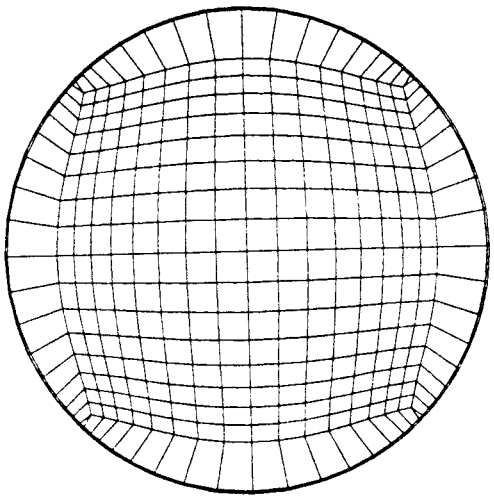

Fig. 1a

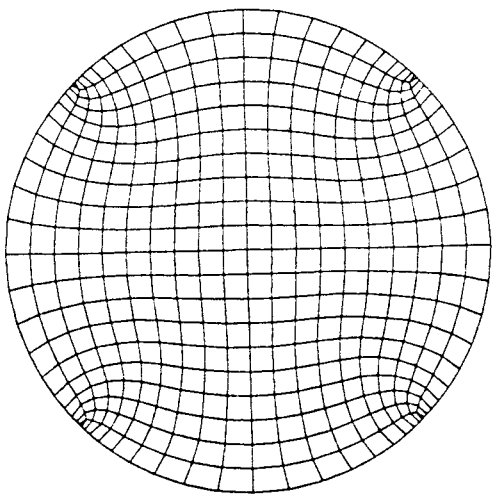

Fig. 1b

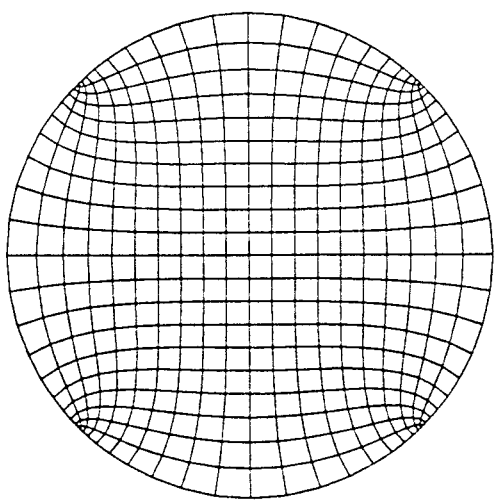

Fig. 1c

Fig. 1 H-grid on a circle (20 × 20 cells) a) initial grid; b) optimized grid; c) elliptic generator grid.

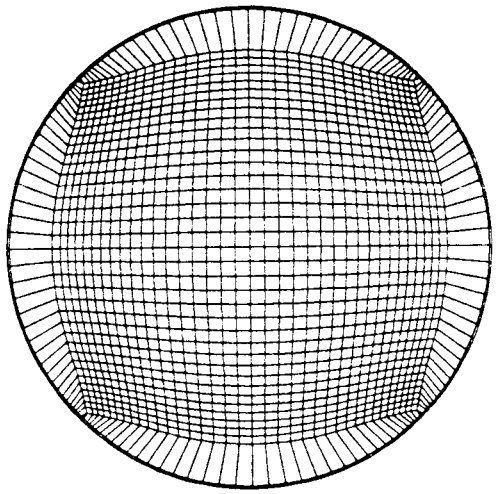

Fig. 2a

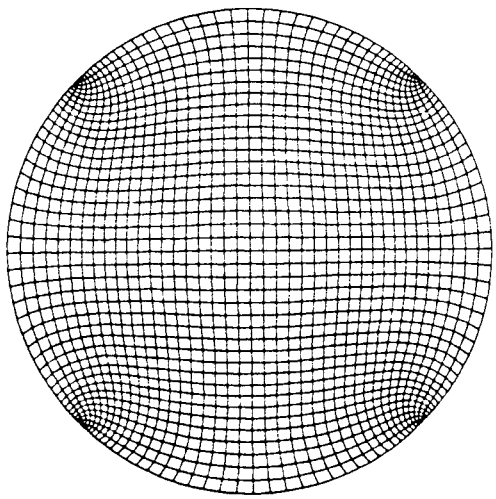

Fig. 2b

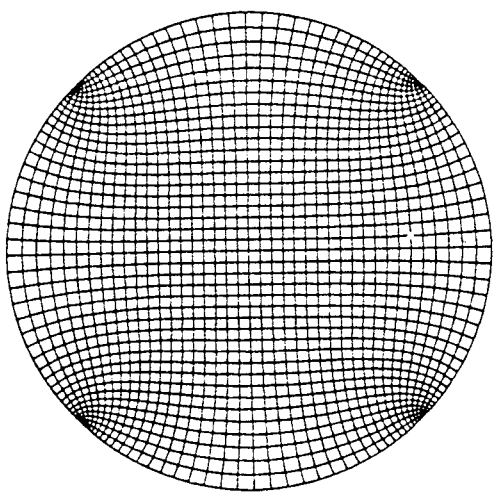

Fig. 2c

Fig. 2 H-grid on a circle (40 × 40 cells) a) initial grid; b) optimized grid; c) elliptic generator grid.

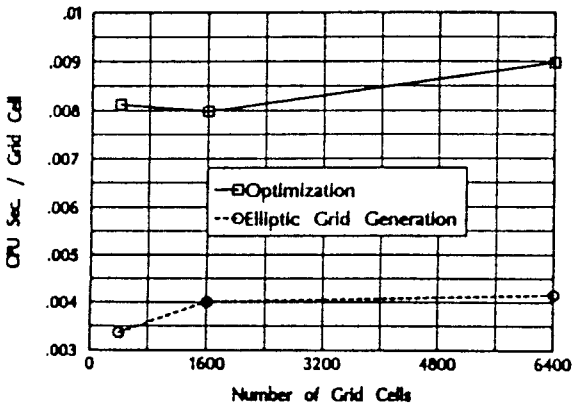

Fig. 3 Computational expense comparison of optimization vs. elliptic grid generation.

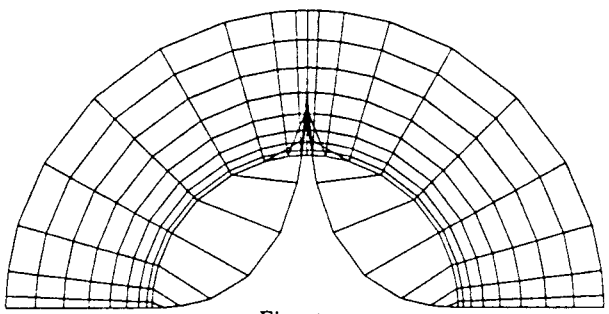

Fig. 4a

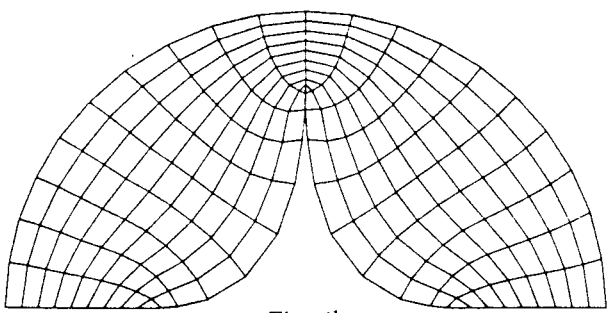

Fig. 4b

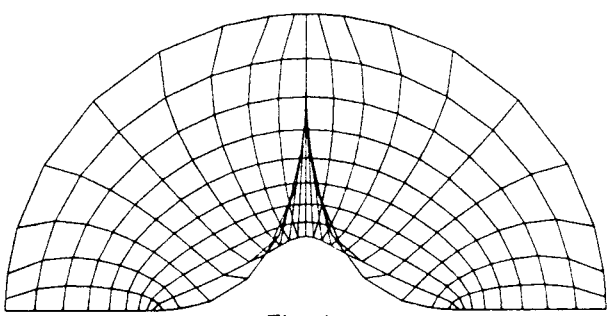

Fig. 4c

Fig. 4 H-grid over a concave-convex boundary (20 × 9 cells)
a) initial overlapped grid; b) optimized grid;
c) elliptically generated grid.

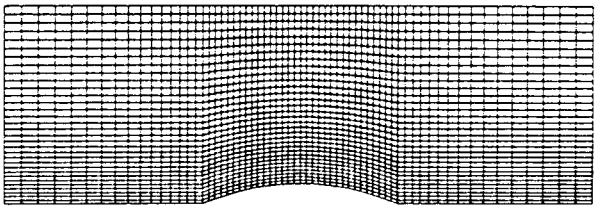

Fig. 5a

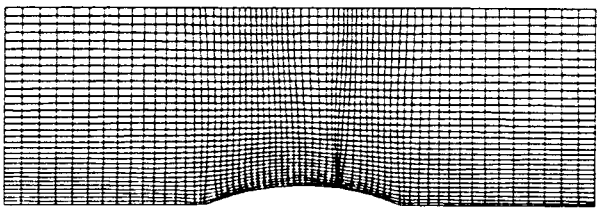

Fig. 5b

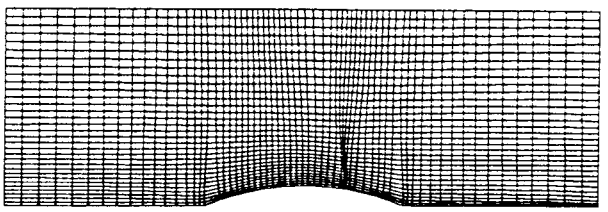

Fig. 5c

Fig. 5 Solution adaptive grid evolution (64 × 32 cells)
a) initial grid;
b) intermediate grid, 500 iterations of Euler solver;
c) converged grid, 2000 iterations of Euler solver.

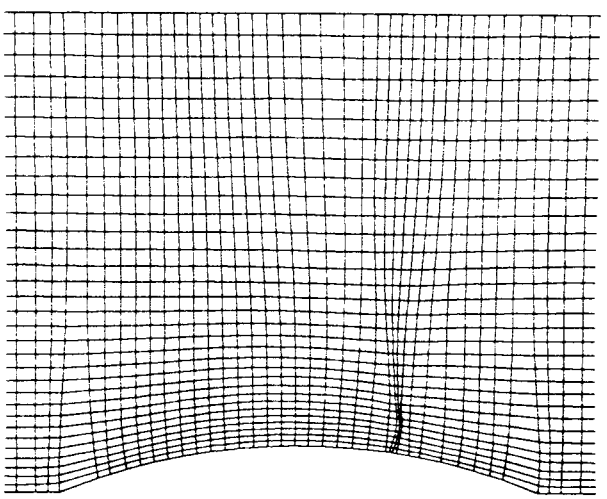

Fig. 6 Shock region of converged solution adaptive grid.

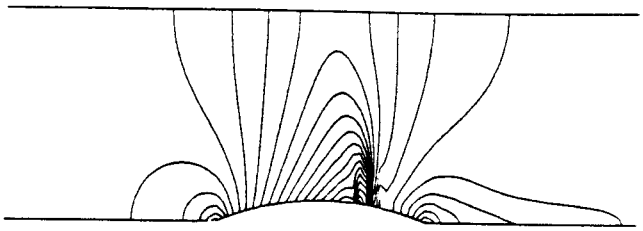

Fig. 7a

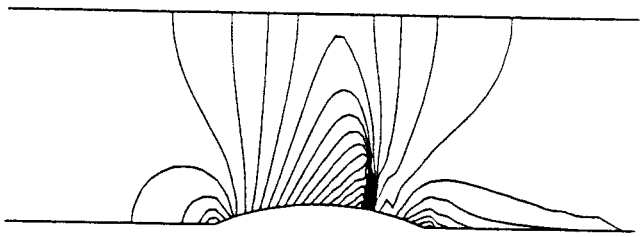

Fig. 7b

Fig. 7 Mach contours a) on fixed grid;
b) on solution adaptive grid.

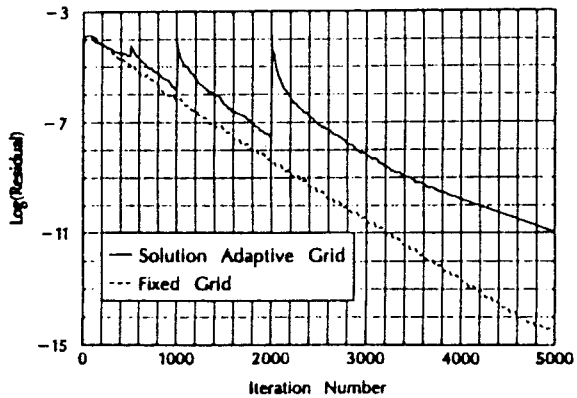

Fig. 8 Euler solver convergence rates for
fixed and solution adaptive grids.

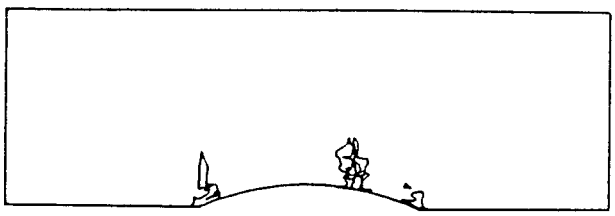

Fig. 9 Residual contours on solution adaptive grid.