TRANSONIC AIRFOIL CALCULATIONS USING SOLUTION-ADAPTIVE GRIDS

Terry L. Holst
Ames Research Center, NASA, Moffett Field, California

and

David Brown
Informatics, Inc., Palo Alto, California

Presented at AIAA 5th Computational Fluid Dynamics Conference
Palo Alto, California, June 22-23, 1981

TRANSONIC AIRFOIL CALCULATIONS USING SOLUTION-ADAPTIVE GRIDS

Terry L. Holst*
Ames Research Center, NASA, Moffett Field, California

and

David Brown†
Informatics, Inc., Palo Alto, California

Abstract

A new algorithm for generating solution-adaptive grids (SAG) about airfoil configurations embedded in transonic flow is presented. The present SAG approach uses only the airfoil surface solution to recluster grid points on the airfoil surface, i.e., the reclustering problem is one dimension smaller than the flow-field calculation problem. Special controls automatically built into the elliptic grid generation procedure are then used to obtain grids with suitable interior behavior. This concept of redistributing grid points greatly simplifies the idea of solution-adaptive grids. Numerical results indicate significant improvements in accuracy for SAG grids relative to standard grids using the same number of points.

I. Introduction

The concept of rearranging grid points in a finite-difference calculation so as to improve solution accuracy (i.e., local clustering as opposed to global grid refinement) is not by itself new. Many researchers have used various types of grid clustering to improve solution accuracy for many different applications. A common example is the normal-direction grid stretching, routinely used in boundary-layer calculations. The normal-direction flow-field gradients in the boundary layer are relatively large compared with the normal gradients associated with the outer inviscid flow. Thus the normal distribution of grid points is usually very fine in the boundary layer and relatively coarse in the inviscid region with smooth exponential stretching in between. The rate of stretching used in this situation may be related to the expected boundary-layer thickness (i.e., the Reynolds number) or other user input, but in general is not influenced by the particular solution. Therefore, in the present context, this type of clustering is not considered to be solution adaptive. A solution-adaptive grid (SAG) technique is defined to be a grid generation technique in which the flow-field solution influences the evolving grid.

This flow-field solution influence may be achieved via either "simultaneous" or "nonsimultaneous" iteration. In the simultaneous iteration SAG approach, the finite-difference grid and the flow field are iterated together and therefore converge simultaneously. This approach is usually used only if the grid generation procedure itself is iterative, presumably an iterative procedure similar to the flow solver. For example, an elliptic equation, grid generation technique[1] uses a relaxation algorithm in much the same fashion as that used by the transonic full-potential equation. Therefore, both the grid and flow solver equations could be iterated and converged together.

Of course, a necessary complication of this approach is that a standard "initial condition" grid, which is consistent with the physics of the problem, must be generated. If the grid generation technique is direct, e.g., the hyperbolic solver technique[2] or the algebraic technique,[3] then a nonsimultaneous SAG approach is best. In this case a grid with no a priori influence from the flow solution is first obtained and then used to obtain a standard flow-field solution. Next, a new grid is computed and adapted to the existing standard flow-field solution. The proper levels of clustering at the proper locations are built into the grid so as to reduce the error associated with the particular solution being computed. Finally, using the newly computed grid, the final flow field is computed. If more precise grid adapting is desired, this process could be repeated in an iterative fashion. Because the latter nonsimultaneous approach seems to be somewhat more general for relaxation problems (either an iterative (elliptic numerical) or direct (algebraic or hyperbolic numerical) grid generation procedure could be used), this is the approach used in the present study. In an attempt to minimize computer time only a single iteration will be used.

An additional point regarding the use of the nonsimultaneous SAG procedure is in order. In many relaxation algorithms the solution is obtained on a sequence of grids (coarse, medium, fine). Converged results from the coarse mesh are interpolated onto the medium mesh, and then from the medium mesh onto the fine mesh, thus providing a good initial guess for the fine mesh calculation. The solution from the medium mesh could be used to produce an improved SAG solution on the fine mesh. The additional computer time used to produce the SAG solution for this situation would be minimal. In the present study no attempt to minimize the total amount of computational work has been made. Instead, the main emphasis is on determining the feasibility of using SAG to improve solution accuracy.

The primary motivation for using the SAG approach can be viewed in either of two ways: 1) improved accuracy for a fixed number of grid points, or 2) improved computational efficiency for fixed accuracy, i.e., fewer grid points. Generally speaking, a SAG approach uses some aspect of the evolving solution to recluster or redistribute grid points so as to reduce the solution truncation error. Several researchers have experimented with different SAG algorithms including Dwyer et al.,[4] who developed a simultaneous iteration procedure

*Research Scientist, Applied Computational Aerodynamics Branch. Member AIAA.
†Currently, graduate student, California Institute of Technology, Pasadena, California.

for various time-accurate heat-transfer problems; Glowinski,[5] who experimented with optimal grids for incompressible, inviscid flow using a finite element technique; and Pierson and Kutler,[6] who determined the optimal grid-point distribution for the Blausius boundary-layer solution and an inviscid Burger's equation solution with a steady-state shock wave. In the last two optimization studies the grid point (or element) positions were themselves chosen as decision variables, thus limiting applicability of these procedures to one- or relatively simple two-dimensional problems. Despite this, insight obtained from such theoretical studies can be used to good advantage on more complicated two- and three-dimensional SAG problems. For instance, results from the Ref. 6 study for the one-dimensional inviscid Burger's equation show that a 55.7% reduction in error is possible with the grid points clustered at the shock wave in an optimal fashion. If a significant part of this error reduction can be retained for nonoptimal transformations in two and three dimensions, the overall benefit could be quite substantial.

The present approach uses a simplified formulation for the development of a SAG algorithm. Although the positioning of the grid points in this procedure is not optimal, it is speculated that significant improvements in solution accuracy can be obtained by qualitatively following the same types of clustering as produced by the optimal SAG approaches. The present SAG procedure can be summarized as follows. First, a preliminary solution is computed using a standard solution procedure. The GRAPE grid generation code[7] is used to generate the mesh and the TAIR conservative full-potential code[8] is used to compute the flow-field solution. Information from this solution is used to redistribute grid points on only the airfoil surface. Next, using the newly clustered surface distribution, a new interior mesh is numerically generated using the GRAPE code. Use of this code is important because of its unique capabilities regarding airfoil surface coordinate line control (both angle and normal spacing can be specified). Without the use of this control, improperly skewed meshes would result at the cluster points. After the SAG is generated the final flow field is recomputed with a second application of the TAIR flow-solver code.

The unique concept of redistributing points only on the airfoil surface greatly simplifies the idea of SAG calculations. It essentially reduces a two-dimensional SAG problem to one dimension and a three-dimensional SAG problem to two dimensions. This concept, even though quite simple, works remarkably well, primarily because the gradients generated around a transonic airfoil are generally maximum on the surface and decay exponentially in a radial-like direction away from the airfoil. Likewise, the amount of grid-point clustering used in the present SAG approach is maximum on the airfoil surface and decays exponentially into the mesh interior. This behavior is a consequence of the elliptic solver algorithm used to numerically generate the finite-difference grid.

The next section of this paper presents details of the present SAG algorithm. A review of the grid generation and flow-solver algorithms is also included in this section because of the important role played by these features in the overall SAG algorithm. In the third section several different numerically generated SAG results are shown and compared with standard grid results.

II. Solution-Adaptive Grid (SAG) Algorithm

Governing Equations

The full-potential equation written in strong conservation-law form is given by

$$(\rho\phi_x)_x + (\rho\phi_y)_y = 0 \tag{1a}$$

$$\rho = \left[1 - \frac{\gamma - 1}{\gamma + 1}\left(\phi_x{}^2 + \phi_y\right)\right]^{1/(\gamma-1)} \tag{1b}$$

where the density (ρ) and velocity components (ϕ_x and ϕ_y) are nondimensionalized by the stagnation density (ρ_s) and the critical sound speed (a_*), respectively; x and y are Cartesian coordinates; and γ is the ratio of specific heats.

Equation (1) expresses mass conservation for flows that are isentropic and irrotational. The corresponding shock-jump conditions are valid approximations to the Rankine-Hugoniot relations for many transonic flow applications. A comparison of the isentropic and Rankine-Hugoniot shock polars is given in Ref. 9.

Equation (1) is transformed from the physical domain (Cartesian coordinates) into a computational domain by using a general independent variable transformation. This general transformation, indicated by (see Fig. 1)

$$\left.\begin{aligned}\xi &= \xi(x,y)\\ \eta &= \eta(x,y)\end{aligned}\right\} \tag{2}$$

maintains the strong conservation-law form of Eq. (1) as discussed in Refs. 10-13. The full-potential equation written in the computational domain (ξ-η coordinate system) is given by

$$\left(\frac{\rho U}{J}\right)_\xi + \left(\frac{\rho V}{J}\right)_\eta = 0 \tag{3a}$$

$$\rho = \left[1 - \frac{\gamma - 1}{\gamma + 1}(U\phi_\xi + V\phi_\eta)\right]^{1/(\gamma-1)} \tag{3b}$$

where

$$\left.\begin{aligned}&U = A_1\phi_\xi + A_2\phi_\eta\ , \qquad V = A_2\phi_\xi + A_3\phi_\eta\\ &A_1 = \xi_x{}^2 + \xi_y{}^2\ , \quad A_2 = \xi_x\eta_x + \xi_y\eta_y\ , \quad A_3 = \eta_x{}^2 + \eta_y{}^2\\ &\text{and}\\ &J = \xi_x\eta_y - \xi_y\eta_x\end{aligned}\right\} \tag{4}$$

U and V are the contravariant velocity components along the ξ and η directions, respectively; A_1, A_2, and A_3 are metric quantities; and J is the Jacobian of the transformation.

The transformed full-potential equation [Eq. (3)] is only slightly more complicated than the original Cartesian form [Eq. (1)] and offers

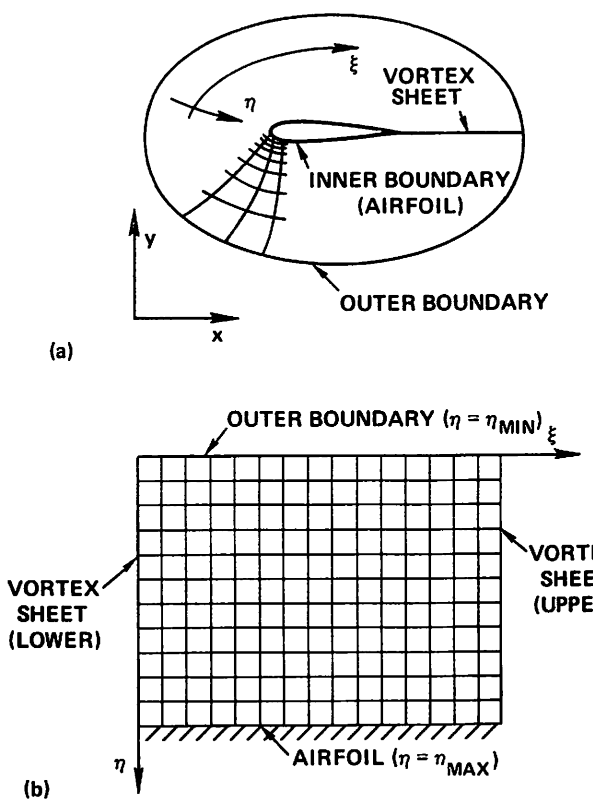


Fig. 1 Numerically generated transformation, $(x,y) \rightarrow (\xi,\eta)$. (a) physical domain; (b) computational domain.

several significant advantages. The main advantage is that boundaries associated with the physical domain are transformed to boundaries of the computational domain. This aspect is illustrated in Fig. 1 where the physical and computational domains for a typical transformation are shown. The inner airfoil boundary becomes the $\eta = \eta_{max}$ computational boundary and the outer physical boundary becomes the $\eta = \eta_{min}$ computational boundary.

Another advantage of this approach is the ability to adjust arbitrarily the mesh spacing on the airfoil surface or in the mesh interior with the provision that the smoothness of the mesh is not disrupted. This feature is the cornerstone of the present SAG algorithm. The next section discusses the method of generating finite-difference grids used in the present study.

Grid Generation

The present SAG algorithm uses the elliptic-solver grid-generation routine introduced by Steger and Sorenson.[14] This procedure is similar to other elliptic-solver routines (e.g., see Refs. 1 and 15) but has one additional highly desirable feature. The Steger-Sorenson algorithm utilizes a novel technique for introducing simple and flexible grid control at domain boundaries. Both normal spacing and cell skewness can be specified by the user. This grid generation flexibility is precisely what makes the present SAG algorithm successful.

Steger and Sorenson use the same elliptic governing equations proposed by Thompson et al.[1]

$$\left.\begin{aligned} \alpha x_{\xi\xi} - 2\beta x_{\xi\eta} + \gamma x_{\eta\eta} &= -J^2(Px_\xi + Qx_\eta) \\ \alpha y_{\xi\xi} - 2\beta y_{\xi\eta} + \gamma y_{\eta\eta} &= -J^2(Py_\xi + Qy_\eta) \end{aligned}\right\} \quad (5)$$

where

$$\alpha = x_\eta^{\ 2} + y_\eta^{\ 2}, \quad \beta = x_\xi x_\eta + y_\xi y_\eta, \quad \gamma = x_\xi^{\ 2} + y_\xi^{\ 2} \quad (6)$$

and

$$J = x_\xi y_\eta - x_\eta y_\xi \quad (7)$$

Appropriate numerical solutions of Eqs. (5) through (7) define the transformation which is functionally indicated by Eq. (2). The primary difference between the two approaches lies in how the P and Q forcing terms are specified. Thompson uses relatively complicated exponential term summations to evaluate P and Q while Steger and Sorenson use

$$\left.\begin{aligned} P &= P_1 \, e^{-a(\eta-\eta_1)} \\ Q &= Q_1 \, e^{-b(\eta-\eta_1)} \end{aligned}\right\} \quad (8)$$

where $P_1 = f(\xi)$, $Q_1 = g(\xi)$ and a and b are all constants. The quantity η_1 represents a mesh boundary which for the present case is the airfoil.

The quantities P_1 and Q_1 are chosen so as to control the spacing between η_1 and $\eta_1 + \Delta\eta$ and the mesh skewness at the airfoil boundary. The rate at which this control decays into the mesh interior is governed by the constants a and b. Typical values used in the present study are 0.5-0.7. Development of the theory behind this control is fully discussed in Ref. 14 and will not be discussed further. Use of a controlling mechanism such as that just presented is imperative for successful operation of the present SAG algorithm. Without proper controls, primarily on cell skewness, the region of a grid generated in the vicinity of a cluster point would be highly skewed. The resulting flow field in this vicinity could be quite inaccurate. An example showing the consequences of no control will be shown in the results section.

The numerical algorithm used to solve Eqs. (5) through (7) with the appropriate P_1 and Q_1 boundary conditions is successive line overrelaxation (SLOR) using a classical coarse-fine mesh sequence. Values of P_1 and Q_1 must be greatly underrelaxed to prevent divergence. This grid generation procedure is available in the GRAPE (Grids About Airfoils Using Poisson's Equation) computer code which was written by Reese Sorenson at Ames (see Ref. 7 for a complete description of this code). Computation times for a typical grid involving 4470 points (149 × 30) requires about 2 sec of CPU time on a CDC 7600 computer using default convergence tolerances.

Full-Potential Equation Algorithm

The full-potential equation algorithm used in the present study is presented and extensively discussed in Refs. 8, 16, and 17. Pertinent details regarding this algorithm are now reproduced for completeness.

Spatial Differencing. A finite-difference approximation to Eq. (1), suitable for both subsonic and supersonic flow regions, is given by

$$\overleftarrow{\delta}_\xi\left[\tilde{\rho}_i\left(\frac{U}{J}\right)_{i+1/2,j}\right] + \overleftarrow{\delta}_\eta\left[\bar{\rho}_j\left(\frac{V}{J}\right)_{i,j+1/2}\right] = 0 \qquad (9a)$$

$$\tilde{\rho}_i = [(1 - \nu)\rho]_{i+1/2,j} + \nu_{i+1/2,j}\rho_{i+k+1/2,j} \qquad (9b)$$

$$\bar{\rho}_j = [(1 - \nu)\rho]_{i,j+1/2} + \nu_{i,j+1/2}\rho_{i,j+\ell+1/2} \qquad (9c)$$

where

$$\left.\begin{aligned} k &= \mp 1 \quad \text{when} \quad U_{i+1/2,j} \gtrless 0 \\ \ell &= \mp 1 \quad \text{when} \quad V_{i,j+1/2} \gtrless 0 \end{aligned}\right\} \qquad (10)$$

and

$$\nu_{i+1/2,j} = \begin{cases} \max[(M^2_{i,j} - 1)C,\ 0] & \text{for} \quad U_{i+1/2,j} > 0 \\ \max[(M_{i+1,j} - 1)C,\ 0] & \text{for} \quad U_{i+1/2,j} < 0 \end{cases} \qquad (11)$$

The operators $\overleftarrow{\delta}_\xi(\ )$ and $\overleftarrow{\delta}_\eta(\ )$ are first-order-accurate, backward-difference operators in the ξ and η directions, respectively; $M_{i,j}$ is the local Mach number; C is a user specified constant (usually between 1.0 and 2.0); and the quantities U and V are computed by standard, second-order-accurate finite-difference formulas. The density (ρ) is computed from the second-order-accurate, discretized version of Eq. (2) and is stored at half points in the finite-difference mesh (i.e., $i + 1/2,\ j + 1/2$). Values needed at $i + 1/2,j$ or $i,j + 1/2$ are obtained by using simple averages.

Use of the density coefficients given by Eqs. (9b) and (9c) is equivalent to the addition of an appropriately differenced artificial viscosity term (Refs. 8 and 16). This effectively maintains an upwind influence in the differencing scheme for supersonic regions anywhere in the finite difference mesh for any orientation of the velocity vector, thus approximating a rotated differencing scheme.

The scheme given by Eq. (9) is centrally differenced and second-order-accurate in subsonic regions. In supersonic regions, the differencing is a combination of the second-order-accurate central differencing used in subsonic regions and the first-order-accurate upwind differencing resulting from the upwind evaluation of the density. As the flow becomes increasingly supersonic the scheme is increasingly retarded in the upwind direction.

AF2 Iteration Scheme. The AF2 fully implicit iteration scheme used in the present study can be expressed in a two-step format given by

Step 1:

$$\left[\alpha - \overrightarrow{\delta}_\eta \bar{\rho}^n_j\left(\frac{A_3}{J}\right)_{i,j-1/2}\right] f^n_{i,j} = \alpha\omega L\phi^n_{i,j} \qquad (12)$$

Step 2:

$$\left[\alpha\overleftarrow{\delta}_\eta \mp \alpha\beta\overleftrightarrow{\delta}_\xi - \overleftarrow{\delta}_\xi \tilde{\rho}^n_i\left(\frac{A_1}{J}\right)_{i+1/2,j} \overrightarrow{\delta}_\xi\right] C^n_{i,j} = f^n_{i,j} \qquad (13)$$

where the n superscript is an iteration index, α is an acceleration parameter (see Ref. 8 for a discussion of α), ω is a relaxation parameter (set equal to 1.8 for all cases), $L\phi^n_{i,j}$ is the nth iteration residual operator (defined by Eq. (9a)), and $f^n_{i,j}$ is an intermediate result stored at each point in the finite-difference mesh. In step 1, the f array is obtained by solving a simple bidiagonal matrix equation for each ξ = constant line. The correction array ($C^n_{i,j} = \phi^{n+1}_{i,j} - \phi^n_{i,j}$) is then obtained in the second step from the f array by solving a tridiagonal matrix equation for each η = constant line. Note that with the AF2 scheme the η-direction difference approximation is split between the two steps. This generates a $\phi_{\eta t}$-type term, which is useful to the iteration scheme as timelike dissipation. (The iterative process is considered as an iteration in pseudotime. Thus, the time derivative is introduced by $(\ )^{n+1} - (\ )^n \sim (\ )_t$.) The split η term also places a sweep direction restriction on both steps, namely, outward (away from the airfoil) for the first step and inward (toward the airfoil) for the second step. No sweep restrictions are placed on either of the two sweeps due to flow direction.

A $\phi_{\xi t}$-type term has been added inside the brackets of step 2 (see Eq. (13)), to provide time-dependent dissipation in the ξ direction. The parameter β is fixed at a value of 0.3 in subsonic regions and specified as needed in supersonic regions (usually between 1.0 and 5.0). The double arrow notation in Eq. (13) on the δ_ξ-difference operator indicates that the difference is always upwind, which on the upper surface is a backward difference and on the lower surface is a forward difference. The sign is chosen in such a way that the addition of $\phi_{\xi t}$ increases the magnitude of the second sweep diagonal.

The full-potential equation solution algorithm just discussed has been coded into a user-oriented computer code called TAIR (Transonic Airfoil Analysis). Details of this code's operation are discussed in Ref. 8. Computation times for a typical transonic airfoil calculation on the default grid containing 4470 points (149 × 30) are about 5 to 15 sec of CPU time on the CDC 7600 computer using the default convergence tolerance.

Grid Clustering Algorithm

As already mentioned the present SAG algorithm redistributes grid points in a solution-adaptive sense by first redistributing points on the airfoil surface. Appropriate interior characteristics of the solution-adaptive grid are then automatically obtained from the elliptic-solver numerical grid generation scheme. The algorithm used to redistribute points on the airfoil surface is quite simple and is now presented. First, determine the normalized arc-length distribution about the airfoil surface (S_i)

$$\bar{S}_1 = 0 \qquad (14a)$$

$$\bar{S}_i = \bar{S}_{i-1} + [(XB_i - XB_{i-1})^2 + (YB_i - YB_{i-1})^2]^{1/2}$$

$$\text{for} \quad i = 2, 3, \ldots, NI \qquad (14b)$$

$$S_i = \bar{S}_i/\bar{S}_{NI} \quad \text{for} \quad i = 1, 2, \ldots, NI \qquad (14c)$$

where XB and YB are the airfoil coordinates and NI is the total number of points on the airfoil surface.

Next, compute values of a test function ($\Delta_{i+1/2}$) designed to simulate (approximately) the surface solution truncation error at the middle of each grid interval, i + 1/2. This test function must be both grid and solution dependent. Several expressions have been tested, with the most suitable given by

$$\Delta_{i+1/2} = (HH_{i+1/2})^3 \times \left| \frac{\left(C_{P_{i+3/2}} - C_{P_{i+1/2}}\right)\Big/H_{i+1} - \left(C_{P_{i+1/2}} - C_{P_{i-1/2}}\right)\Big/H_i}{H_i + H_{i+1}} \right| \quad (15)$$

where

$$H_i = \frac{1}{2}(S_{i+1} - S_{i-1}) \quad (16a)$$

$$HH_{i+1/2} = \max(H_{i+1}, H_i) \quad (16b)$$

and C_p is the standard pressure coefficient. The structure of this expression is partly due to the fact that the TAIR program output involves pressure coefficient data at half points (i.e., i + 1/2) and XB and YB data (and therefore S) at integer points. The $(HH)^3$ quantity, appearing as the lead multiplying term in Eq. (15), is basically an empirical result. Use of $(HH)^2$, which more appropriately models the truncation error associated with the present spatial differencing algorithm, produced too much clustering in the vicinity of the airfoil leading edge. By using $(HH)^3$ a proper balance between the leading edge and shock clustering seemed to result. Use of a test function which puts more emphasis on clustering in supersonic regions, because these regions are only first-order accurate, could be implemented with the present logic but to date has not been investigated.

Once the test function Δ is defined at all airfoil surface points, a new surface grid-point distribution is constructed by adding new points into the old distribution at intervals in which Δ exceeds some user-specified tolerance, T. This process is defined more explicitly as follows:

$$\text{If } \Delta_{i+1/2} \le T \begin{cases} S_k^n = S_i^o \\ \text{increment both } k \text{ and } i \text{ by } 1 \end{cases} \quad (17a)$$

$$\text{If } \Delta_{i+1/2} > T \begin{cases} S_k^n = S_i^o \\ S_{k+1}^n = S_i^o + \dfrac{S_{i+1}^o - S_i^o}{2} \\ \text{increment } k \text{ by } 2 \text{ and } i \text{ by } 1 \end{cases} \quad (17b)$$

where S^n and S^o represent the new clustered and old standard arc-length distributions, respectively. This process is repeated until all values of $\Delta_{i+1/2}$ from i = IMIN to i = IMAX have been tested. IMIN and IMAX set the range of testing which for all cases presented extends from $S_{IMIN} \approx 0.05$ to $S_{IMAX} \approx 0.95$. This eliminates clustering at the airfoil training edge ($S_1 = 0.0$ or $S_{NI} = 1.0$) which was found to be difficult to implement. The difference between the final values of k and i is the number of points added to the arc-length distribution.

The arc-length distribution represented by S_k^n has several slope discontinuities. These discontinuities are removed from S_k^n by a heat equation smoothing process given by

$$\bar{S}_k^n = S_k^n + \varepsilon\left(S_{k+1}^n - 2S_k^n + S_{k-1}^n\right) \quad (18)$$

where the overbar indicates the smoothed distribution and ε is a user-specified smoothing coefficient set to 0.4 for all cases tested. The smoothing process is implemented in an iterative fashion over a limited range, e.g., from $S_k^n \approx 0.05$ to $S_k^n \approx 0.95$, thus excluding the trailing edge. The arc-length distribution near the trailing edge is not smoothed because the smoothing process would destroy the strong clustering needed to cope with the trailing-edge mapping singularity. The boundary conditions applied during this smoothing process force the matching of slope at the endpoints of the smoothing domain. In addition, the smoothed distribution is renormalized after each smoothing iteration to force the matching of function at the smoothing domain endpoints. (This renormalization actually causes the slope-matching boundary condition to be only approximate.) Thus, the transition from a smoothed region into an unsmoothed region is quite well behaved in both function and slope.

The smoothing iteration continues until a maximum iteration limit (~30) is encountered or until a user-specified smoothness constraint is satisfied. The smoothness constraint utilizes a user-specified parameter (RATIO) defined by

$$\text{RATIO} = \frac{\max\limits_k[S_k''(\bar{\xi})]\text{clustered mesh}}{\max\limits_i[S_i''(\bar{\xi})]\text{standard mesh}} \quad (19)$$

where $\bar{\xi}$ is the normalized equally spaced computational coordinate along the airfoil and $S_k''(\bar{\xi})$ is the second derivative of S_k with respect to $\bar{\xi}$. The smoothing iteration continues until $\max\limits_k[S_k''(\bar{\xi})]$ of the clustered distribution is less than RATIO times $\max\limits_i[S_i''(\bar{\xi})]$ of the initial standard arc-length distribution. Values of RATIO between 1.0 and 3.0 have been tested with the present algorithm.

Once the smoothing iteration has been completed, new values of Cp are obtained at the new S_k positions, $C_p(S_k)$, from the initial distribution, $C_p(S_i)$, by cubic spline interpolation. Then the whole process is repeated until no additional points are added by the test on $\Delta_{i+1/2}$. For larger values of T this addition of points process will be repeated, perhaps, three or four times. For smaller values of T this process may be repeated as many as a dozen times resulting in much higher levels of clustering. Occasionally, if the T and RATIO values are both specified too low the overall clustering algorithm will never converge to a sensible result. However, a little experience in choosing proper values for T and RATIO will generally prevent this situation from occurring.

After the clustering and smoothing iterations have both converged, the final clustered arc-length distribution will contain more points than the original distribution. If the final number of points is not the desired number — and in general it won't be — a distribution with the desired number of points is obtained via cubic spline interpolation. This interpolation process does not change the clustering built into the arc-length distribution, only the number of points used to define it. During this interpolation step the user may choose a finer mesh with the newly clustered distribution or simply "scale back" to the same number of points as in the original mesh. With the former option the cost of obtaining the clustered mesh would be smaller. This is the recommended procedure for implementation of the present SAG concept for practical applications. Some testing of this implementation has been done with reasonably successful results. However, all results presented herein will be of the latter type, i.e., the initial standard and final clustered grids will have exactly the same number of grid points in both directions. This approach is desirable because for each SAG computation an additional standard mesh computation involving the same number of grid points is available for direct comparison.

After the final interpolated arc-length distribution has been obtained, new x and y airfoil coordinates, which reflect the same clustering as that of the arc-length distribution are computed via cubic spline interpolation. These coordinates are then used as the required Dirichlet boundary conditions for the final grid generation using GRAPE.

During the initial grid generation, in which the standard grid was established, the two airfoil surface boundary conditions utilized for mesh control were

$$\theta = 90° \tag{20}$$

$$\Delta n_i = 1.5\ \Delta S_i \tag{21}$$

where θ is the angle of intersection between the η and ξ coordinates at the airfoil surface and n is the physical arc length along the η coordinate direction. The first condition forces orthogonality and the second condition forces a constant tangential to normal mesh spacing ratio around the airfoil surface. For these conditions all cells around the airfoil surface should be (approximately) similar rectangles with an aspect ratio of 1.5. The second boundary condition given by Eq. (21), however, is not acceptable for the final solution-adapted grid generation. Specification of a constant tangential/normal mesh spacing ratio at the airfoil surface forces a drastic reduction of the normal spacing in regions of large tangential clustering. Thus the ξ lines (η = constant lines) would be drastically pulled toward the airfoil at all cluster points. To eliminate this difficulty the boundary condition of Eq. (21) is changed for the SAG computation. The new boundary condition simply uses the direct specification of the old Δn distribution saved from the previous standard grid calculation. The values of $\Delta n(S_i)$ are interpolated via cubic spline so as to yield $\Delta n(S_k)$ and then directly substituted in place of the Δn values which would have normally resulted from Eq. (21). Use of this boundary condition essentially causes the $\eta = \eta_1 + \Delta\eta$ coordinate to be unaffected by any amount of clustering. Several results presented in the next section illustrate this feature.

III. Computed Results

The SAG algorithm discussed in the previous section is evaluated in this section by presenting a range of numerically computed transonic airfoil solutions. For all calculations the GRAPE computer code[7] has been used to obtain the finite-difference grids and the TAIR computer code[8] has been used to solve the governing full-potential equation.

The first case considered is for the NACA 0012 airfoil at $M_\infty = 0.75$ and $\alpha = 1.5°$. Pressure coefficient distributions computed on 105 × 22 meshes, both standard and solution-adapted versions, are compared in Fig. 2. A blowup of the grid showing detail around the airfoil is plotted in Fig. 3a for the standard grid and in Fig. 3b for the solution-adapted grid. For this SAG calculation T = 0.0006 and RATIO = 1.2. In addition, because no clustering was performed at the airfoil leading edge or on the lower surface, the smoothing interval was reduced to only the upper surface, $(S_{min}, S_{max}) \approx (0.55, 0.95)$. For these conditions, only a moderate amount of clustering centered at the shock wave is introduced into the SAG solution (see Fig. 3b). This results in a sharper shock wave with a slight upstream shift in position (see Fig. 2), as well as a corresponding decrease in lift and drag. The difference between the two results is not large for this case, partly because the mesh is already fine enough for the standard grid solution to do a

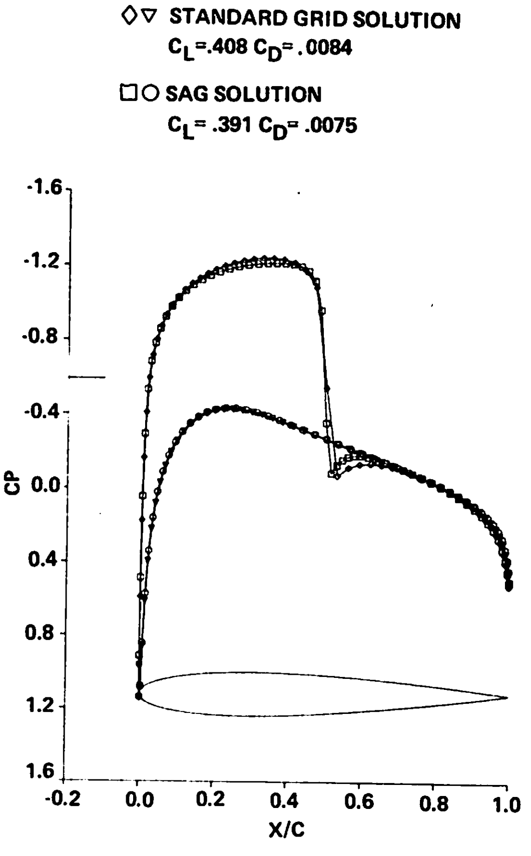
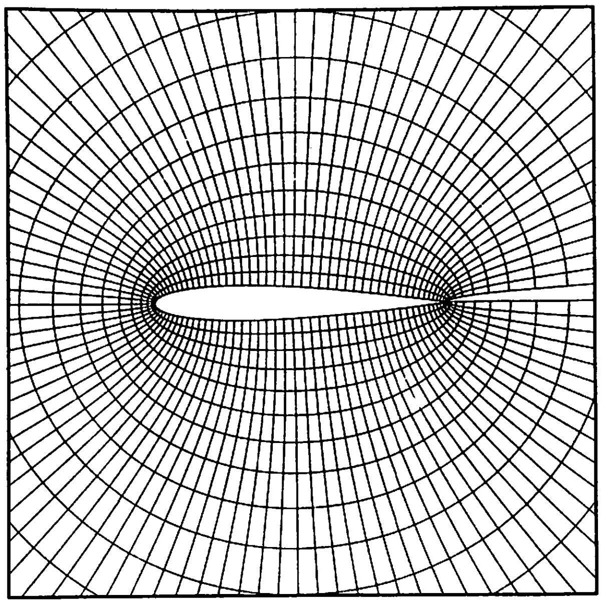
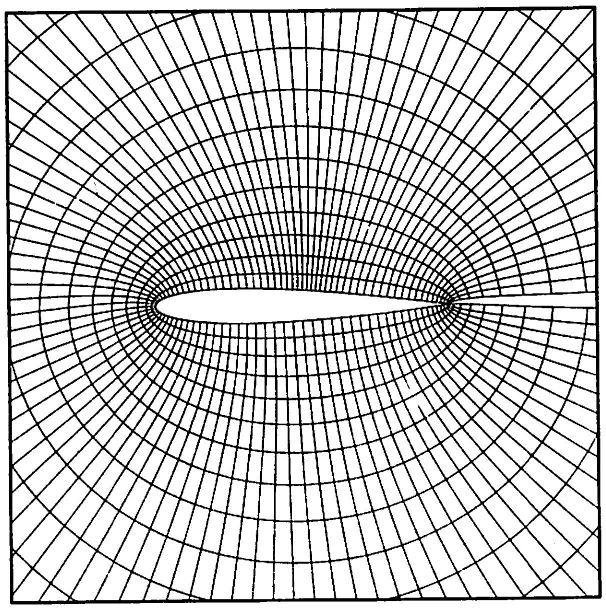


Fig. 2 Pressure coefficient distribution comparison (NACA 0012 airfoil, $M_\infty = 0.75$, $\alpha = 1.5°$).



(a) Standard grid.



(b) Solution-adapted grid.

Fig. 3 Numerically generated finite-difference mesh about an NACA 0012 airfoil (105 × 22).

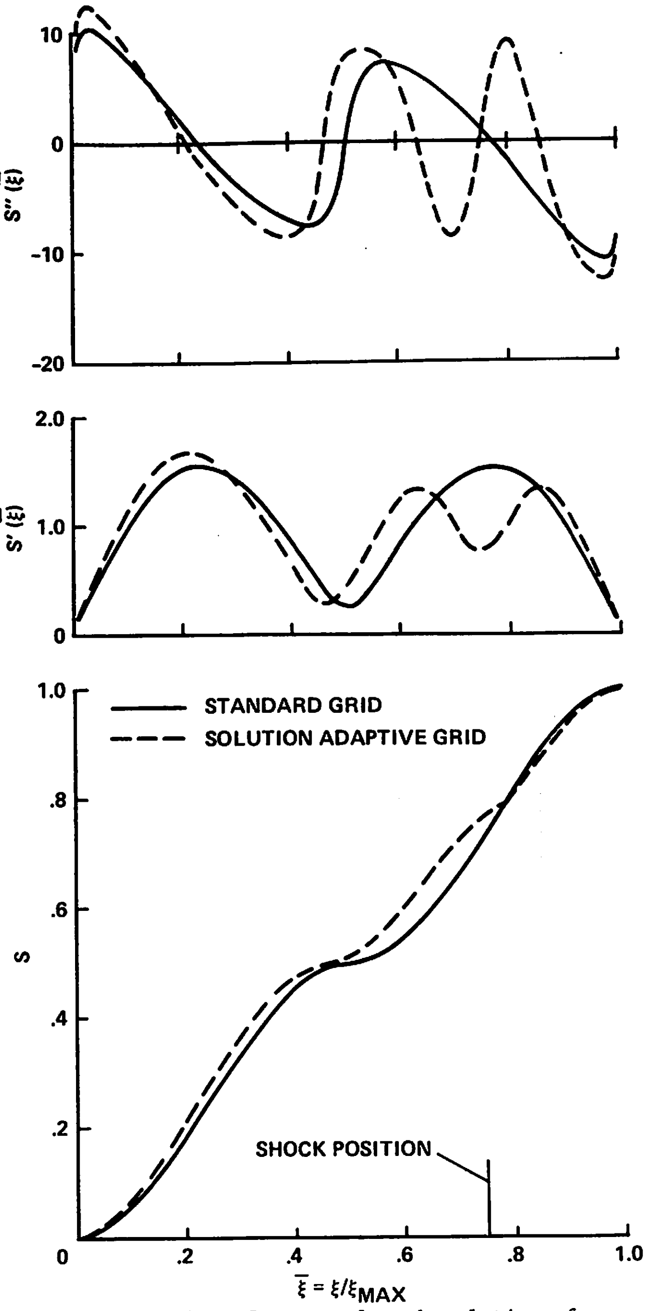


Fig. 4 Airfoil surface arc-length solution, function, slope, and curvature (NACA 0012 airfoil, $M_\infty = 0.75$, $\alpha = 1.5°$).

relatively good job and partly because only a moderate amount of clustering was used for the SAG case. Nevertheless, a significant improvement in accuracy is achieved even for the present case, and will become apparent when the asymptotic behavior of the present solution is discussed.

More complete information about the SAG airfoil surface arc-length distribution for this case is shown in Fig. 4. The arc length, S, and its first and second derivatives, $S'(\bar{\xi})$ and $S''(\bar{\xi})$, are plotted versus the normalized computational coordinate, $\bar{\xi}$. Results for both the standard and SAG grids are shown for comparison. The airfoil leading edge is represented by $S = 0.5$, at least for the present symmetrical NACA 0012 case. From the S vs $\bar{\xi}$ SAG curve it is readily seen that the leading edge occurs at $\bar{\xi} = 0.46$, indicating that for this calculation 46% of the surface grid points exist on the lower surface and 54% on the upper surface. Of course, the symmetrical standard arc-length distribution of grid points is equally divided between the upper and lower surfaces. The redistribution of points about the airfoil in this global sense is a direct result of the present SAG algorithm. In other words, the clustering achieved around any high-frequency detail of the solution — in this case the shock wave on the upper surface — results in a slight coarsening of the solution in all other areas.

The $S'(\bar{\xi})$ quantity physically represents the ratio of the local grid spacing along the airfoil, ΔS, to the average value, ΔS_{avg}. At the shock, $S \cong 0.75$, $S'(\bar{\xi})$ has been reduced by approximately a factor of two for the SAG solution relative to the standard grid solution. This indicates a factor of two reduction in the surface grid spacing at the shock wave. Finally, the $S''(\bar{\xi})$ vs $\bar{\xi}$ curves indicate that the required smoothness requested by the RATIO = 1.2 specification has been achieved. The maximum value of $S''(\bar{\xi})$ for the SAG solution in the range $(S_{min}, S_{max}) = (0.55, 0.95)$ occurs at $\bar{\xi} \cong 0.95$ and is approximately 1.2 times the corresponding value from the standard grid solution.

A most important consideration for any type of finite-difference calculation procedure is determining the accuracy of the solution, i.e., how well does the numerical solution approximate the governing differential equation? This question is most easily answered (for nonlinear equations) by computing a series of solutions on successively finer meshes and then noting the behavior of the solution. For consistency, an asymptotic solution should result, but the rate of approach to the asymptotic solution is also of interest. Figures 5 and 6 show the results of such a study applied to the first example calculation (NACA 0012 airfoil, $M_\infty = 0.75$ $\alpha = 1.5°$). The lift and wave-drag coefficients are plotted as a function of the average grid spacing, $\Delta = 1/NI$, where NI is the number of grid points in the ξ direction. Results for both standard and solution-adaptive grids are presented. For the SAG results the clustering tolerance parameter, T, is scaled by NI, i.e., $T_{actual} = 100\ T_{input}/NI$. With this scaling, a constant value of T_{input} will produce approximately the same amount of clustering regardless of mesh dimensions. The values of T_{input} and RATIO were set to 0.0006 and 1.2, respectively, for all cases computed in Figs. 5 and 6. For all calculations the ratio of the number of grid points in the ξ direction to the number of points in the η direction is approximately held fixed. Figure 5 shows that both the standard and SAG solution sequences approach the same asymptotic value of lift. The most interesting feature, however, is the difference in the rate of approach. The SAG curve is consistently under the standard curve and therefore produces a more accurate value of lift on a coarser mesh. The drag results of Fig. 6 essentially produce the same

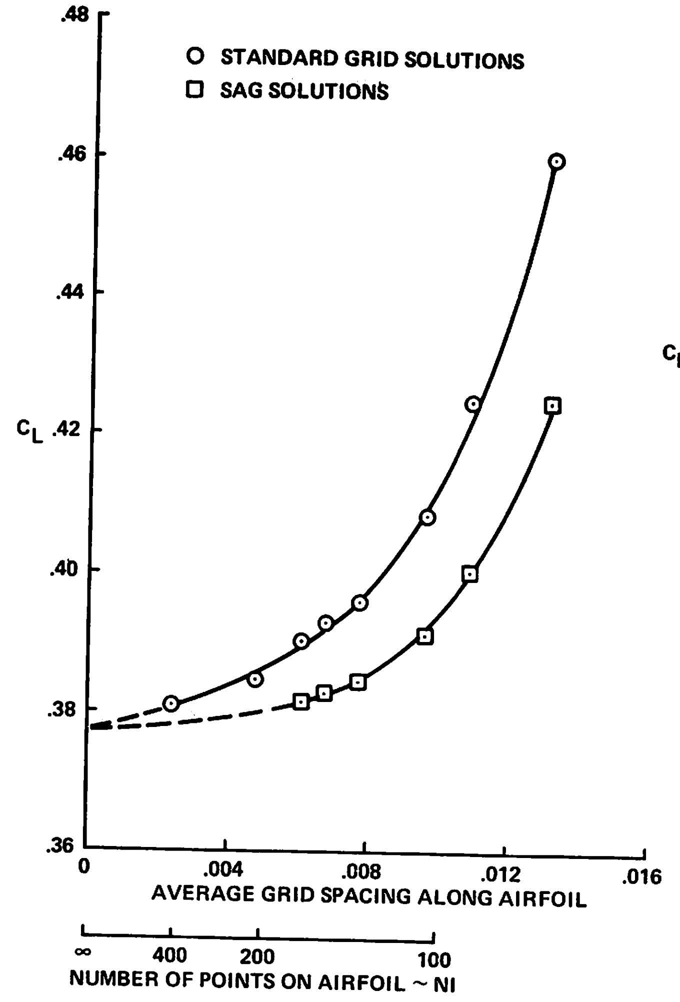


Fig. 5 Asymptotic behavior of the lift on a sequence of meshes (NACA 0012 airfoil, $M_\infty = 0.75$, $\alpha = 1.5°$).

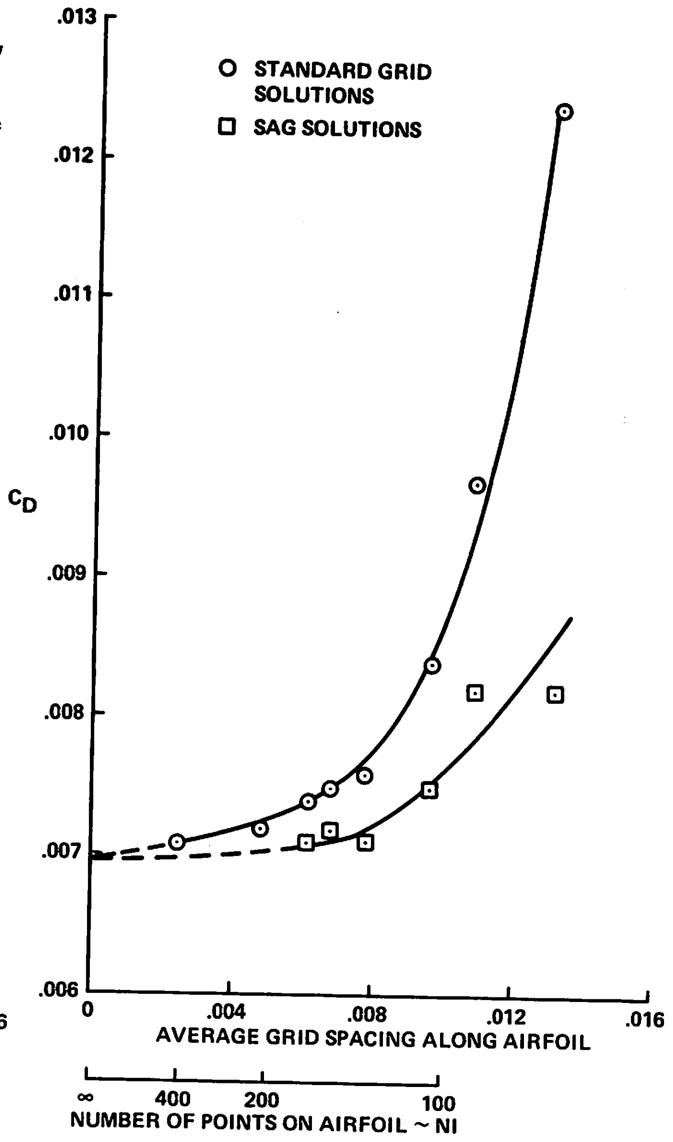


Fig. 6 Asymptotic behavior of the drag on a sequence of meshes (NACA 0012 airfoil, $M_\infty = 0.75$, $\alpha = 1.5°$).

conclusions. However, the drag data is more scattered. This is undoubtedly due to the more sensitive nature of the wave-drag pressure integration. These curves indicate that a SAG solution produces lift and drag accuracy equivalent to a standard grid with two or three times as many grid points. In other words, the SAG solutions have less error than the standard grid solutions containing the same number of grid points.

For the sequence of calculations presented in Figs. 5 and 6, the error associated with the standard grid calculations ranged from about 3% for lift and 7% for drag for the finest mesh (209 × 43) to about 22% for lift and 80% for drag for the coarsest mesh (77 × 16). For two-dimensional potential calculations the mesh size chosen is usually nearer the finer of these two extremes (e.g., 149 × 31). The error associated with this size mesh for many applications is not that significant. Therefore, reduction of this error, by whatever amount, is not that significant. However, for three-dimensional potential calculations as well as two-dimensional Euler/Navier-Stokes calculations, mesh dimensions are considerably smaller and therefore the corresponding solutions may have significant amounts of truncation error. For these types of calculations a means of reducing inherent truncation error would have significant importance.

For the sequence of calculations presented in Figs. 5 and 6, the error reduction provided by the SAG approach for the lift was about 40% to 65% and for the drag about 50% to 75%. The amount of improvement can be increased for higher degrees of clustering, i.e., for smaller values of T. However, some difficulty in forming a smooth asymptotic curve has been encountered for higher degrees of clustering. Thus, establishment of the actual error reduction is difficult.

The second case considered is for the NLR airfoil at $M_\infty = 0.65$ and $\alpha = 2.5°$. Pressure coefficient distributions for both standard and SAG meshes, each consisting of 129 × 31 points, are compared in Fig. 7. Details of both the standard and solution-adapted grids are shown in Figs. 8 and 9. Figures 8a and 9a show blowup views of the standard and SAG grids around the entire airfoil, while Figs. 8b and 9b show further enlargements around the airfoil leading edge. For this case the solution contains two cluster points, one at the leading edge ($S \simeq 0.5$) and one at the shock ($S \simeq 0.58$). Values of T and RATIO used in this calculation were 0.00015 and 1.2, respectively, and the full smoothing interval, (Smin, Smax) = (0.05, 0.95), was used. The smaller value of T produces a more highly clustered grid. For instance, the maximum reduction in $S'(\xi)$ for this case was 2.5 as compared with 2.0 for the previous case. Again, as in the previous case, the result of the clustering procedure is to sharpen the shock and capture more crisply the high-frequency detail of the solution (see Fig. 7). Note that details of the SAG solution away from the clustered areas are in almost perfect agreement with the standard grid calculation.

Results presenting additional characteristics of the present SAG algorithm are given in Figs. 10 and 11. Figure 10 shows a leading-edge blowup of the grid for precisely the same conditions of the previous calculation except that the grid control features imposed by the GRAPE grid generation code have been removed. The clustered arc-length distribution about the airfoil surface is identical to the previous case but the resulting grid is completely different. This difference can be seen by directly comparing Figs. 10 and 9b. The grid with no control is highly skewed, particularly in the vicinity of the leading edge. Such grid skewness does not generally interfere with solution stability but does produce an inaccurate solution. Another parameter which has an influence on solution accuracy is the smoothness control parameter, RATIO. Figure 11 shows an enlargement of the leading-edge grid for precisely the same conditions used to compute the grid of Fig. 9b, except that RATIO has been changed from 1.2 to 3.0. The resulting difference can be seen by directly comparing Figs. 11 and 9b. Allowing large values of $S''(\xi)$ to exist in the surface arc-length distribution does not harm solution quality nearly as much as the skewness shown in the previous example. However, sensitivity studies to determine the effect of grid smoothness on solution accuracy seem to indicate that a value of RATIO below 2.0 (e.g., 1.2 to 2.0) produces the best results. Values of RATIO below 1.2 (e.g., 1.0 to 1.1) can also cause problems, primarily in conjunction with small values of T (e.g., 0.0001 to 0.0002), and should be avoided. For small T and RATIO ~ 1.0, the clustering algorithm either does not converge or converges to a nonsensible answer. Acceptable SAG results can be obtained quite easily and reliably if this combination of parameter values is avoided.

The final case presented is for an experimental airfoil from Gates-Learjet at $M_\infty = 0.8$ and

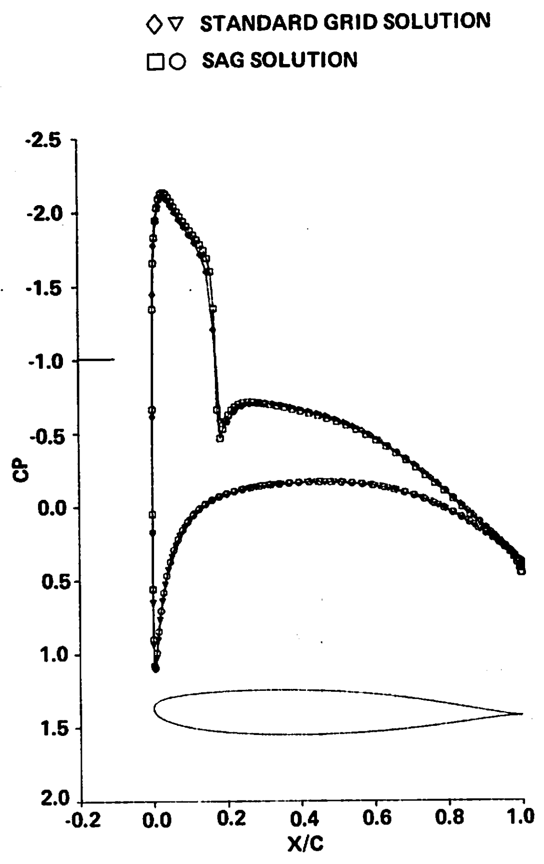
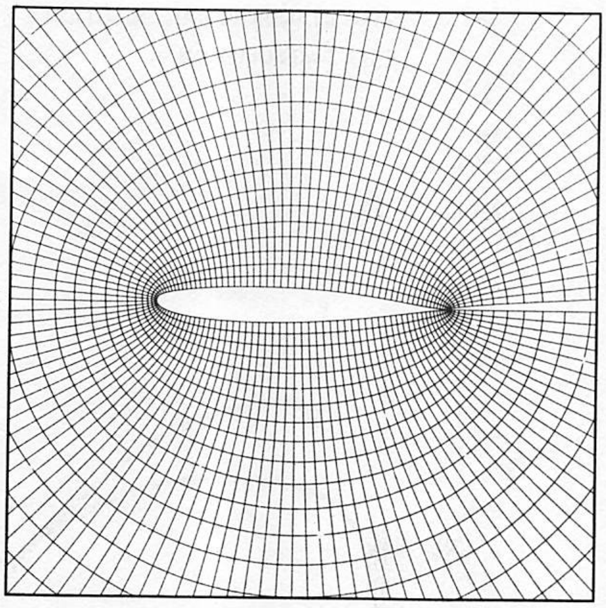
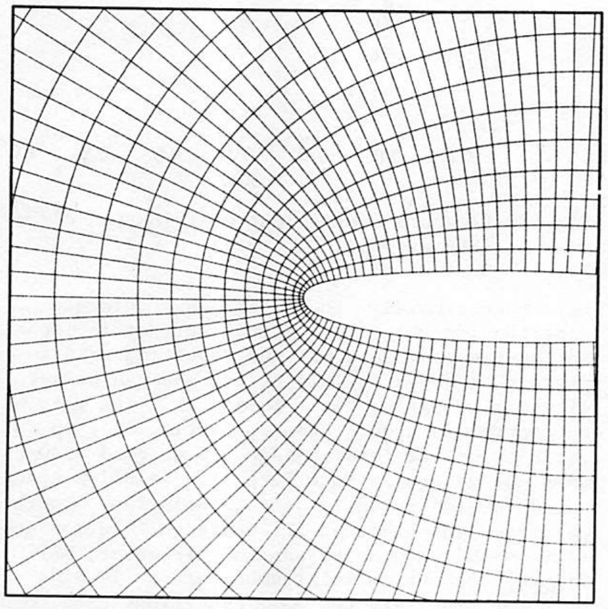


Fig. 7 Pressure coefficient distribution comparison (NLR airfoil, $M_\infty = 0.65$, $\alpha = 2.5°$).

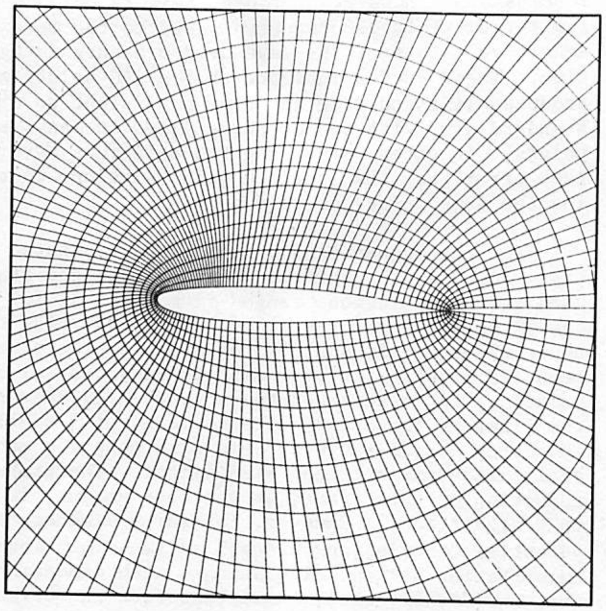


(a) Airfoil view.

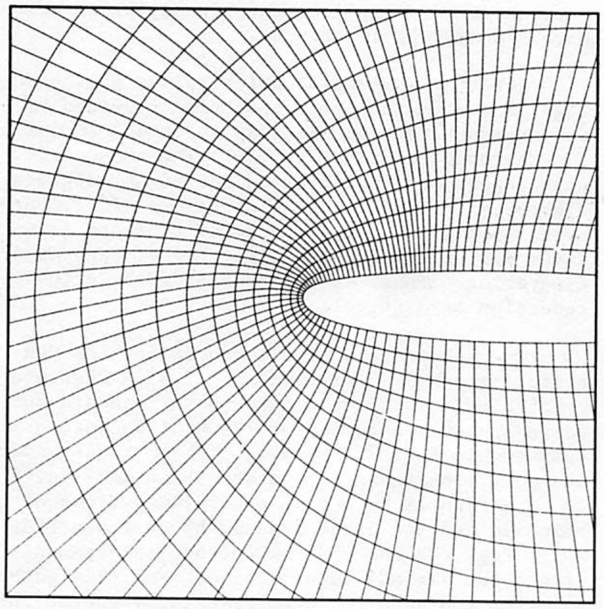


(b) Leading edge detail.

Fig. 8 Numerically generated finite-difference mesh about an NLR airfoil (standard grid).



(a) Airfoil view.



(b) Leading edge detail.

Fig. 9 Numerically generated finite-difference mesh about an NLR airfoil (solution-adapted case, $M_\infty = 0.65$, $\alpha = 2.5°$).

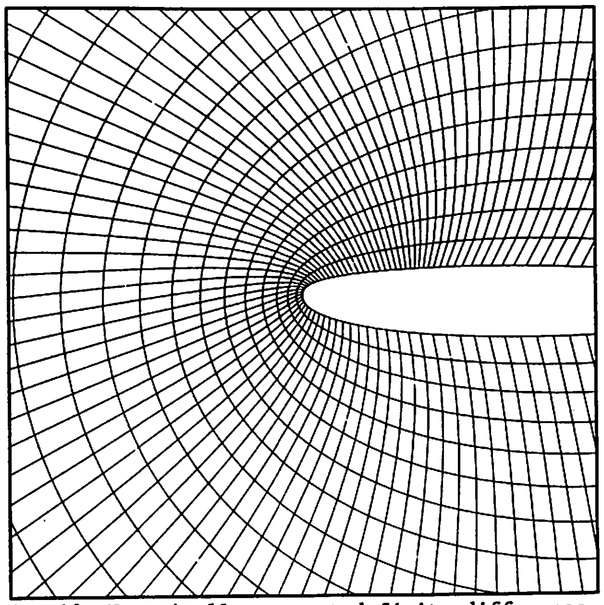


Fig. 10 Numerically generated finite-difference mesh about an NLR airfoil (solution-adapted case with no grid control, $M_\infty = 0.65$, $\alpha = 2.5°$).

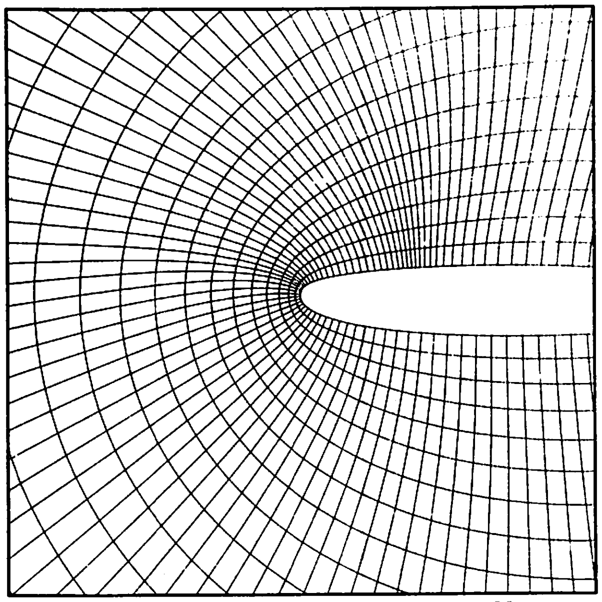
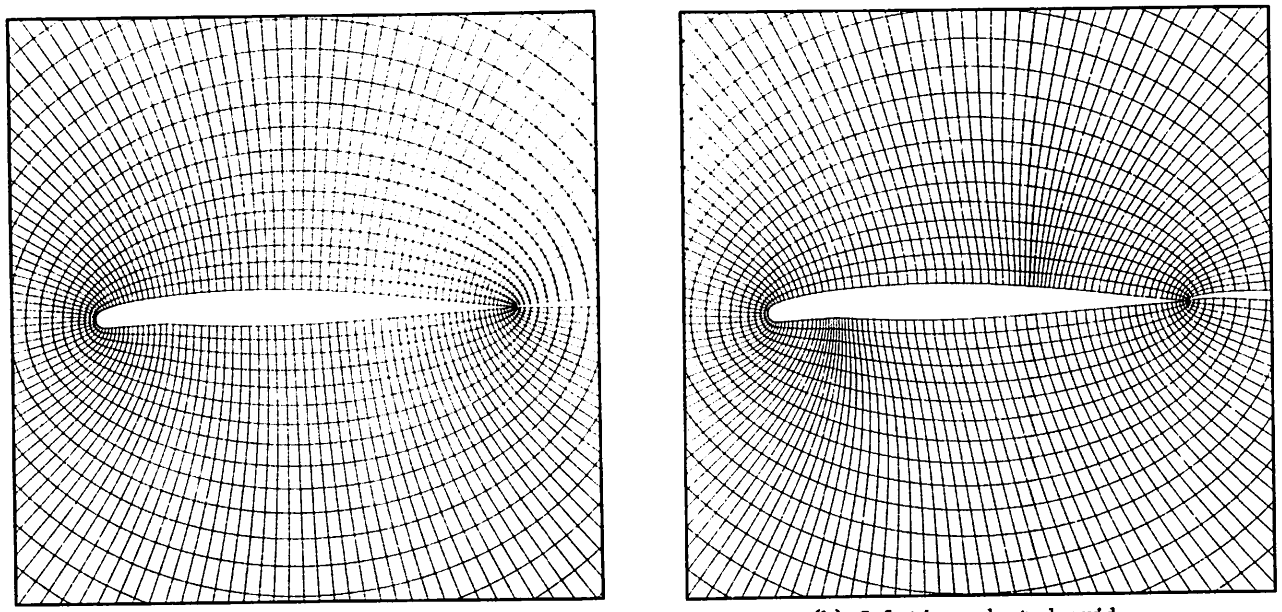


Fig. 11 Numerically generated finite-difference mesh about an NLR airfoil (solution-adapted case with little smoothing, $M_\infty = 0.65$, $\alpha = 2.5°$).

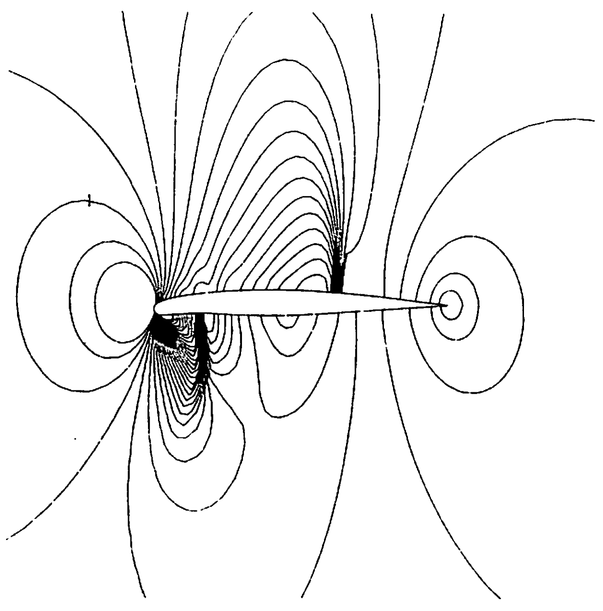
$\alpha = 0°$. Both standard and solution-adaptive grids involving 149 × 37 points are shown in Fig. 12. Mach number contours about the airfoil for both standard and SAG solutions are shown in Fig. 13. These contour maps were plotted for a Mach number range of 0.7 to 1.72 in increments of 0.03. The SAG calculation was computed with T and RATIO values of 0.0002 and 2.0, respectively. For this solution three cluster points exist, the leading edge and both the lower and upper surface shocks. Notice how the solution-adaptive grid of Fig. 12b precisely adapts to the solution given in Fig. 13. Direct comparison of the two Mach number contour maps shown in Fig. 13 shows several interesting differences. First, the width of both shock waves is much smaller for the SAG solution than for the standard solution. This indicates a superior, sharper shock capture as a result of the automatic clustering process. In addition, the lower surface leading-edge expansion which is followed almost immediately by the lower-surface shock wave produces a large suction pressure covering only the first 15% of chord. The sonic line produced by this sharp leading-edge expansion/shock interaction extends 30% further into the solution interior for the SAG solution than for the standard solution. The more extensive development of this region for the SAG solution is due to the grid clustering around both the leading-edge expansion and the shock wave. This example shows that even when multiple cluster points are involved, significant solution improvements can be obtained using the present SAG approach.



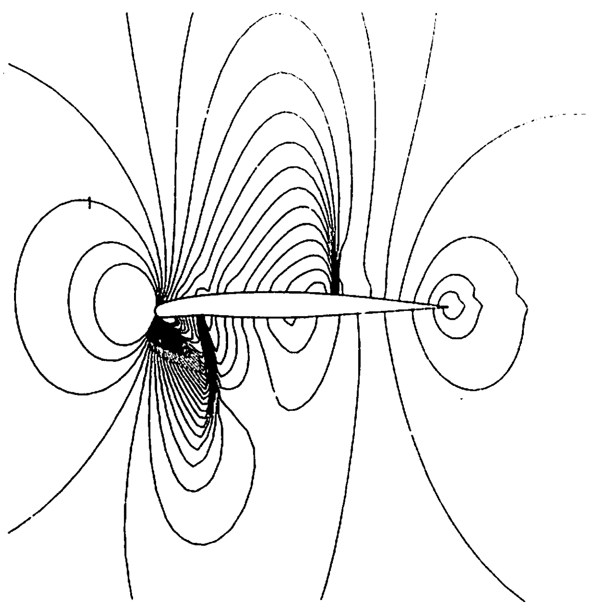
(a) Standard grid.

(b) Solution-adapted grid.

Fig. 12 Numerically generated finite-difference mesh about an experimental Gates-Learjet airfoil (149 × 37).



(a) Standard grid solution.



(b) Solution-adapted grid solution.

Fig. 13 Mach number contours about an experimental Gates-Learjet airfoil ($M_\infty = 0.65$, $\alpha = 2.5°$).

IV. Conclusions

A new algorithm for generating solution-adaptive grids about airfoil configurations embedded in transonic flow is presented. The present SAG approach uses only the airfoil surface solution to recluster grid points on the airfoil surface. Therefore, the reclustering problem is one dimension smaller than the flow-field calculation problem. Special controls automatically built into the elliptic grid generation procedure are then used to obtain grids with suitable interior behavior. This concept of redistributing grid points greatly simplifies the idea of solution-adaptive grids.

Computed results on solution-adaptive grids indicate significant reductions in the error relative to standard grids using the same number of grid points. Results computed on mesh sequences indicate that both standard grid and SAG calculations approach the same asymptotic values of lift and drag. However, the rate of approach of the SAG sequence is much faster than that of the standard grid sequence. For this sequence of calculations the error reduction provided by the SAG approach was about 40% to 65% for the lift and about 50% to 75% for the drag.

In the present formulation the full-potential equation in conservative form was used as the flow-field governing equation. The present SAG algorithm, however, could be used with other formulations, e.g., the Euler or even the Navier-Stokes equations. For three-dimensional full-potential or Euler/Navier-Stokes applications where both execution time and storage place stringent requirements on almost any computer, the present SAG algorithm could provide even more significant benefits than those obtained in the present study.

Acknowledgment

The authors express their gratitude to Reese Sorenson of the Applied Computational Aerodynamics Branch at Ames Research Center for his assistance in running the GRAPE grid generation code.

References

[1]Thompson, J. F., Thames, F. C., and Mastin, C. M., "Automatic Numerical Generation of Body-Fitted Curvilinear Coordinate System for Field Containing Any Number of Arbitrary Two-Dimensional Bodies," Journal of Computational Physics, Vol. 15, 1974, pp. 299-319.

[2]Steger, J. L. and Chaussee, D. S., "Generation of Body Fitted Coordinates Using Hyperbolic Partial Differential Equations," FSI Report 80-1, Jan. 1980.

[3]Eiseman, P. R., "Three-Dimensional Coordinates About Wings," Proceedings of AIAA Fourth CFD Conference, July 1979, pp. 166-174.

[4]Dwyer, H. A., Kee, R. J., and Sanders, B. R., "Adaptive Grid Method for Problems in Fluid Mechanics and Heat Transfer," AIAA Journal, Vol. 18, No. 10, Oct. 1980, pp. 1205-1212.

[5]Glowinski, R., "On Grid Optimization for Boundary Value Problems," Stanford University Report No. STAN-CS-79-720, Feb. 1979.

[6]Pierson, B. L. and Kutler, P., "Optimal Nodal Point Distribution for Improved Accuracy in Computational Fluid Dynamics," AIAA Journal, Vol. 18, No. 1, Jan. 1980, pp. 49-54.

[7]Sorenson, R. L., "A Computer Program to Generate Two-Dimensional Grids About Airfoils and Other Shapes by the Use of Poisson's Equation," NASA TM-81198, May 1980.

[8]Dougherty, F. C., Holst, T. L., Gundy, K. L., and Thomas, S. D., "TAIR — A Transonic Airfoil Analysis Computer Code," NASA TM, in preparation, 1981.

[9]Steger, J. L. and Baldwin, B. S., "Shock Waves and Drag in the Numerical Calculation of Isentropic Transonic Flow," NASA TN D-6997, 1972.

[10]Lapidus, A., "A Detached Shock Calculation by Second-Order Finite Differences," Journal of Computational Physics, Vol. 2, 1967, pp. 154-177.

[11]Viviand, H., "Conservative Forms of Gas Dynamic Equations," La Recherche Aerospatiale, No. 1, Jan.-Feb. 1974, pp. 65-68.

[12]Vinokur, M., "Conservative Equations of Gas Dynamics in Curvilinear Coordinate Systems," Journal of Computational Physics, Vol. 14, Feb. 1974, pp. 105-125.

[13]Steger, J. L., "Implicit Finite Difference Simulation of Flow About Arbitrary Geometries with Application to Airfoils," AIAA Paper 77-665, June 1977. (See also AIAA Journal, Vol. 16, No. 7, July 1978, pp. 696-686.)

[14]Steger, J. L. and Sorenson, R. L., "Automatic Clustering Near a Boundary in Grid Generation with Elliptic Partial Differential Equations," Journal of Computational Physics, Vol. 33, No. 3, Dec. 1979, pp. 405-410.

[15]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. 24, 1977, pp. 274-302.

[16]Holst, T. L., "An Implicit Algorithm for the Conservative, Transonic Full Potential Equation Using an Arbitrary Mesh," AIAA Paper 78-1113, July 1978. (See also AIAA Journal, Vol. 17, No. 10, Oct. 1979, pp. 1038-1045.)

[17]Holst, T. L., "A Fast, Conservative Algorithm for Solving the Transonic Full-Potential Equation," Proceedings of AIAA Fourth Computational Fluid Dynamics Conference, July 1979, pp. 109-121. (See also AIAA Journal, Vol. 18, No. 12, Dec. 1980, pp. 1431-1439.)