NASA Technical Memorandum 102198

A Simplified Self-Adaptive Grid Method, SAGE

C. Davies and E. Venkatapathy, Ames Research Center, Moffett Field, California

October 1989

NASA

National Aeronautics and
Space Administration

Ames Research Center
Moffett Field, California 94035

A SIMPLIFIED SELF-ADAPTIVE GRID METHOD, SAGE

Carol B. Davies* and Ethiraj Venkatapathy†
Ames Research Center

SUMMARY

The formulation of the Self-Adaptive Grid Evolution (SAGE) code, based on the work of Nakahashi and Deiwert, is described in the first section of this document. The second section is presented in the form of a user guide which explains the input and execution of the code, and provides many examples. Application of the SAGE code, by Ames Research Center and by others, in the solution of various flow problems has been an indication of the code's general utility and success. Although the basic formulation follows the method of Nakahashi and Deiwert, many modifications have been made to facilitate the use of the self-adaptive grid method for single, zonal, and multiple grids. Modifications to the methodology and the simplified input options make this current version a flexible and user-friendly code.

1. METHODOLOGY

1.1 Introduction

Solution-adaptive grid methods have become useful tools for efficient and accurate flow predictions. In supersonic and hypersonic flows, strong gradient regions such as shocks, contact discontinuities, shear layers, etc., require careful distribution of grid points to minimize grid error and thus produce accurate flow-field predictions. Frequently, the choice of the first grid topology does not adequately capture these flow structures. It has been shown that an effective way of obtaining accurate solutions is by intelligently redistributing (i.e., adapting) the original grid points based on the first flow-field solution and then computing a new solution using the adapted grid. Many adaptive procedures have been proposed based on minimization principles. Some apply only to simple grid topologies and others are extendable to multiple and zonal grid topologies. The cost efficiency of grid adaptions in terms of CPU time depends on the basic formulation of the adaptive-grid solver.

The self-adaptive grid procedure outlined by Nakahashi and Deiwert (1985) has evolved into a flexible tool for adapting and restructuring two-dimensional (2-D) grids. The adaptive-grid methodology is based on variational principles, but the problem is posed in an algebraic, unidirectional manner for multidimensional adaptions. The procedure is analogous to applying tension and torsion spring forces proportional to the local flow gradient at every grid point and finding the equilibrium position of the resulting system of grid points. The multidimensional problem of grid adaption is split into a series of one-dimensional (1-D) problems along the computational coordinate lines. The reduced 1-D problem then requires a tridiagonal solver to find the location of grid points along a given coordinate line. Multidirectional adaption is achieved by the sequential application of the method in each coordinate direction.

*Sterling Software, Palo Alto, CA.
†Eloret Institute, Sunnyvale, CA.

The tension force directs the redistribution of points to the strong gradient regions. To maintain smoothness and a measure of orthogonality of grid lines, torsional forces are introduced that relate information between the family of lines adjacent to one another. The smoothness and orthogonality constraints are direction-dependent, since they relate only the coordinate lines that are being adapted to the neighboring lines that have already been adapted. This implies that the solutions are nonunique and depend on the order and direction of adaption. Nonuniqueness of the adapted grid is acceptable since it makes possible an overall and local error reduction through grid redistribution.

The Self-Adaptive Grid Evolution (SAGE) code has been built with many flexible elements that make it user-friendly. The second section of this report is a user guide, which is independent of the first section and can be used as a separate document. However, the nomenclature and details of the grid adaption procedure within the code consistently follow the analysis, allowing the user to understand the code and implement any individual changes, if desired. The user guide describes the input parameters and their effects and the adaption procedure, execution commands, and subroutines, and provides many examples.

1.2 Formulation of Adaptive Grid Scheme

This section describes in more detail the analysis used in the SAGE code. As stated in the introduction, the 2-D adaption takes place as a series of 1-directional adaptions. Figure 1 illustrates this concept: a flowfield solution has been obtained using the unadapted grid shown in the top diagram, and the points in this grid are now adapted to the flow solution. The first adaption, as shown in the bottom left diagram, marches from left to right; adaption has already taken place along the first and second lines, and line three is currently being adapted. The second adaption, as shown in the bottom right diagram, marches from bottom to top, "adapting" the adapted grid. This example is an arbitrary order of adaption and marching directions; any other order will also produce an adapted grid.

Figure 2 shows the current adaption line in more detail. The equation controlling the redistribution of points along each line consists of two parts: (1) the 1-D approach utilizing tension spring constants and (2) the addition of the torsion term. The tension spring constants (ω) have the effect of clustering the redistributed points into the high gradient regions. The torsion term provides a correction to this redistribution to maintain continuity between sequentially adapted lines.

The current line to be adapted is j, and we are marching from bottom to top: this implies that lines $j-1$ and $j-2$ have already been adapted. For clarity, this analysis always uses j as the stepping (marching) direction and i as the adaption direction. The arc length at A, $s_{i,j}$, is defined (along j) as

$$s_{i+1} = s_i + \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2} \tag{1}$$

A spring-like force is defined to act between each i and $i+1$ nodes such that

$$\omega_i \Delta s_i = K \tag{2}$$

where ω_i, the spring constant, is a weighting function based on the flow gradient, and K is the resultant force. In order to redistribute the points along a line with the minimum solution error, the adaptive procedure defines the weighting function as proportional to the derivative of the flow variable (Nakahashi and Deiwert, 1985). In this formulation, ω is defined as a function of the normalized flow gradient, $\bar{f}$, such that

$$\omega_i = 1.0 + A\bar{f}_i^{\,B} \tag{3}$$

where A and B are constants directly related to the grid spacing and are chosen to maintain the grid intervals to within the limits (Δs_{MIN} and Δs_{MAX}) set by the user on input. They are defined in appendix I of this section.

For this set of equations, with no torsion term, we can solve directly for Δs_i. Taking the sum of both sides of equation (2) gives

$$\sum_{l=1}^{n_i} \Delta s_l = s_{max} = K \sum_{l=1}^{n_i} 1/\omega_l$$

giving

$$K = s_{max} / \sum_{l=1}^{n_i} 1/\omega_l \tag{4}$$

Substituting back in equation (2), we obtain

$$\Delta s_i = s_{max} / \left(\omega_i \sum_{l=1}^{n_i} \frac{1}{\omega_l} \right) \tag{5}$$

In the SAGE code, this solution technique is used only along the initial adaption line. Continuing this approach for successive line-by-line adaptions will not necessarily create a mesh that is sufficiently smooth for input into computational flow-field codes. To ensure a more reasonable grid, there needs to be some connectivity between adjacent j lines to provide a measure of orthogonality and smoothness. To provide this communication, the concept of torsion is introduced that is a function of the $j, j-1$ and $j-2$ lines. The torsion term is evaluated as $-\tau_i(s_i - s_i')$, where the torsion parameter τ defines the magnitude of this torsion force and s' defines its inclination (i.e., orthogonality and smoothness). The derivation of the torsion term is described in a later section.

Equation (2) can be rewritten to represent the force balance, i.e.,

$$\omega_i(s_{i+1} - s_i) - \omega_{i-1}(s_i - s_{i-1}) = 0 \tag{6}$$

To introduce the torsion force to the system of equations, the torsion term is added to equation (6) to obtain

$$\omega_i(s_{i+1} - s_i) - \omega_{i-1}(s_i - s_{i-1}) + \tau_i(s_i' - s_i) = 0 \tag{7}$$

which is rearranged to produce the coefficients of the tridiagonal matrix used to solve for s_i, that is

$$\omega_{i-1}s_{i-1} - (\omega_i - \omega_{i-1} + \tau_i)s_i + \omega_i s_{i+1} = \tau_i s_i' \tag{8}$$

This equation is written for each interval along the adaption line, producing a system of $(n_i - 1)$ equations. Since s_1 and s_n are known, there are $n_i - 2$ rows in the matrix. This equation is solved iteratively until $\sum_{i=1}^{n_i} | s_i^{(n)} - s_i^{(n-1)} | < 10^{-3} \times s_{max}$.

This system of equations is central to the adaption technique used in the SAGE code and most of the description that follows pertains to deriving the coefficients of this equation.

1.3 Auxiliary Equations

1.3.1 Tension spring constant, ω_i. As described in the formulation, ω_i acts as a tension force between the interval $(s_{i+1} - s_i)$ and can be imagined to be a spring (coaligned with the grid line) connecting the two grid points. This tension parameter is defined in equation (3) as

$$\omega_i = 1 + A\bar{f}_i^B$$

where $\bar{f}_i$ is a function of the gradient of the flow variable, q, i.e.,

$$\bar{f}_i = \frac{f_i - f_{min}}{f_{max} - f_{min}} \text{ and } f_i = \frac{\partial q_i}{\partial s} \qquad (9)$$

The standard format of the user input flow-field file contains the conservative flow variables Q, as defined in the plotting software package, PLOT3D, which assumes these flow variables are $\rho, \rho u, \rho v$, and e. Since the adaption is based on a scalar function q, this function can be evaluated as a user-specified combination of flow variables Q. Pressure and Mach number also are included as adaption variables and are internally computed, assuming Q to be the conservative flow variable. However, Q need not be restricted to conservative flow variables, but can be any combination of user-specified flow variables that represent the flow field.

To remove the unwanted oscillations from the discrete evaluation, f_i is smoothed by adding a second derivative term, i.e., $f_i = .75 f_i + .125(f_{i+1} - f_{i-1})$. By default, two smoothing passes are made, but this can be overridden by changing the filter input control parameter. The weighting function, ω_i, is also smoothed in the same way.

1.3.1.1 ***A* and *B*.** As seen in equation (3), ω is also a function of A and B. These variables are important elements in the self-adaptive scheme, since they control the size limits of the grid spacing. A and B are computed from the user-supplied maximum and minimum allowable grid spacings $(\Delta s_{MAX}, \Delta s_{MIN})$. These are relative values of mesh size and allow the user to specify the proportion of largest to smallest Δs in the final adapted grid.

The value of A is constant throughout the grid adaption and is given by

$$A = \frac{\Delta s_{MAX}}{\Delta s_{MIN}} - 1 \qquad (10)$$

The value of B is computed (by an iterative process) for each j line to provide

$$input\, \Delta s_{MIN} = computed\, \Delta s_{min}$$

That is, the computed minimum grid spacing is equal to the user-requested value. Appendix I gives a detailed description of the derivation of these two parameters. It should be noted that the derivation of B given in Nakahaski and Deiwert (1985) is inaccurate and that the formulation given in this appendix is a corrected version.

1.3.1.2 Torsional effect on the evaluation of ω. As noted earlier, the function of variables A and B is to control the size limits of the adapted-grid-mesh spacing and, as shown in appendix I, they are computed from the 1-D equation. The addition of the torsion term will somewhat alter the resulting grid spacings, which now may not exactly correspond to the requested limits. To provide for additional control, a weighting factor is applied to ω such that $\omega_i = (1 + A\bar{f}^B)\omega_{t_i}$, where ω_{t_i} is the weighted tightness factor. When equation (8) is solved, each Δs_i is tested and if it does not lie within the requested spacing limits, ω_{t_i} will be computed; otherwise $\omega_{t_i} = 1.0$. In the case of $\Delta s_i < \Delta s_{MIN}$, we need to relax (decrease) the value of ω_i in order to produce a larger Δs_i in the next iteration. For the case of a too large Δs_i, ω_i should be increased. We therefore use the modifier

$$\omega_{t_i} = \frac{1}{2}\left(\frac{\Delta s_i}{\Delta s_{MIN}} + 1\right) \text{ or } \omega_{t_i} = \frac{1}{2}\left(\frac{\Delta s_i}{\Delta s_{MAX}} + 1\right)$$

depending on whether $\Delta s_i < \Delta s_{MIN}$ or $\Delta s_i > \Delta s_{MAX}$. Equation (8) is therefore solved using the modified values of ω.

1.3.2 Torsion term, $\tau(s_i - s_i')$. As mentioned previously, the torsion term is added to the 1-D equation to preserve the required smoothness and orthogonality of the grid. This term is constructed as being analogous to the behavior of a torsion spring. The torsion force, T, relates the node at $(i, j-1)$ to the node (i, j) and is proportional to the turning angle, θ (see fig. 2), and is defined as

$$T_i = C\theta_{i,j-1} \tag{11}$$

where C is the torsion spring constant. This force is thus added to equation (6), giving

$$\omega_i(s_{i+1} - s_i) - \omega_{i-1}(s_i - s_{i-1}) + C\theta_{i,j-1} = 0 \tag{12}$$

It is clear that by doing this, the equilibrium distribution of the grid points has been altered by the magnitude of the torsional force term. The next step is to relate this torsional force to the previously defined parameters on the adaption line. Figure 3 shows in detail the grid segment around the point (i, j) at A. Since θ is generally small, we can approximate θ_i by $(s_i' - s_i)/|DA'|$, where $s_i' - s_i$ is a function of the inclination angle computed from the proportioned orthogonality and straightness vectors, as described later. In addition, we assume that C is proportional to the maximum ω_i and the local aspect ratio of the grid cell. Hence we can write

$$C = \frac{\lambda\omega_{max}(s_{i+1,j-1} - s_{i-1,j-1})}{2|DA'|} \tag{13}$$

where λ is the proportional coefficient and is an input quantity.

The term τ used in equation (8) can thus be defined as

$$\tau_i = \frac{\lambda\omega_{max}(s_{i+1,j-1} - s_{i-1,j-1})}{2|DA'|^2} \tag{14}$$

The evaluation of $|DA'|$ is given in appendix II of this section.

1.3.2.1 Derivation of s'. The influence of smoothness and orthogonality is directly included in equation (8) through s', and the evaluation of s' is an important element in this adaptive-grid procedure.

Figure 3 shows the torsion vector $\vec{t_i}$ intersecting the line AC at s'_i such that

$$s'_i = s_i + AA' \qquad (15)$$

That is, $\Delta s'_i = AA'$. To evaluate AA' we first define the unit torsion vector $\hat{t}$ associated with each nodal point (i, j) [note that $\hat{t}$ *acts* from the point $(i, j-1)$] such that

$$\hat{t}_{i,j} = \frac{1}{|t|}[C_t \hat{e}_i + (1 - C_t)\hat{n}_i] \qquad (16)$$

where

C_t is the input parameter that requests the percentage of straightness to orthogonality

$\hat{e}_i$ is an average straightness vector from $(i, j-2)$ to $(i, j-1)$

$\hat{n}_i$ represents the orthogonality vector between the $j-1$ line and the node i, j and is a function of the $\hat{s}$, $\hat{b}$ and $\hat{u}$ vectors defined below.

For simplicity, figure 3 shows each vector intersecting the j line in the interval $(i, i+1)$; however, this is not necessarily the case. In general, we assume that $\hat{t}_i$ intersects the j line in the interval $(l, l+1)$, where the derivation of l is described in appendix II of this section.

1.3.2.2 Evaluation of vectors $\hat{e}$, $\hat{s}$, and $\hat{n}$.

1.3.2.2.1 Straightness vector $\hat{e}_i$

The unit vector $\hat{e}_i$ (direction ED in fig. 3) is defined as the sum of three straightness vectors, $\hat{d}_{i-1}$, $\hat{d}_i$, $\hat{d}_{i+1}$, where

$$\hat{d}_i = d_{x_i}\vec{i} + d_{y_i}\vec{j}$$

and

$$d_{x_i} = \frac{1}{|d_i|}(x_{i,j-1} - x_{i,j-2}) \text{ and } d_{y_i} = \frac{1}{|d_i|}(y_{i,j-1} - y_{i,j-2})$$

Therefore,

$$\hat{e}_i = \frac{1}{\sqrt{3}}(\hat{d}_{i-1} + \hat{d}_i + \hat{d}_{i+1})$$

If the line $j-2$ does not exist (which will occur at the second line from a physical boundary), it is assumed that $\hat{e} = \hat{n}$.

1.3.2.2.2 Streamwise vector $\hat{s}_i$

The unit vector $\hat{s}_i$ (direction AC in fig. 3) is the streamwise vector from $s_i \rightarrow s_{i+1}$ along a j line. We have

$$\hat{s}_{i,j} = s_{x_i}\vec{i} + s_{y_i}\vec{j}$$

where

$$s_{x_i} = \frac{1}{|s_i|}(x_{i+1,j} - x_{i,j}) \text{ and } s_{y_i} = \frac{1}{|s_i|}(y_{i+1,j} - y_{i,j})$$

In order to maintain continuity in the neighborhood of the side-edge boundaries, $\hat{s}_1$ and $\hat{s}_{n_i-1}$ are evaluated as orthogonal to the edge boundaries at i_{st} and i_{end}. To provide a smooth transition into the internal values of $\hat{s}$, the adjacent $\hat{s}_2$, $\hat{s}_3$, and $\hat{s}_4$ are amended by

$$\hat{s}_2 = \frac{3}{4}\hat{s}_1 + \frac{1}{4}\hat{s}_2,\ \hat{s}_3 = \frac{1}{2}\hat{s}_1 + \frac{1}{2}\hat{s}_3,\ \text{and}\ \hat{s}_4 = \frac{1}{4}\hat{s}_1 + \frac{3}{4}\hat{s}_4$$

Similarly amended are $\hat{s}_{n_i-2}$, $\hat{s}_{n_i-3}$, and $\hat{s}_{n_i-4}$.

1.3.2.2.3 Normal vector $\hat{n}_i = f(\hat{b}, \hat{u})$

The vector $\hat{n}$ is a combination of two vectors:

$$\hat{n} = \frac{1}{|n_i|}(t_n\hat{b} + (1 - t_n)\hat{u}) \tag{17}$$

where $\hat{u}_i$ represents orthogonality to the $j - 1$ line, and $\hat{b}_i$ the orthogonality to the j line. The parameter t_n proportions $\hat{b}$ and $\hat{u}$ and is defaulted to 0.5, changing only at the upper and lower marching boundaries, as discussed in the next section. Figure 4 shows the relationship between these three orthogonal vectors. The vector $\hat{u}$ is defined as normal to $\hat{r}_i$, the sum (average) of $\hat{s}_{i-1}$ and $\hat{s}_i$ along the $j - 1$ line such that

$$\hat{r}_i = r_{x_i}\vec{i} + r_{y_i}\vec{j}$$

where

$$r_{x_i} = \frac{1}{\sqrt{2}}(s_{x_i} + s_{x_{i-1}})_{j-1} \text{ and } r_{y_i} = \frac{1}{\sqrt{2}}(s_{y_i} + s_{y_{i-1}})_{j-1}$$

Since $\hat{u}$ is normal to $\hat{r}$ we have

$$\hat{u}_i = \pm(u_{x_i}\vec{i} + u_{y_i}\vec{j}) \text{ where } u_{x_i} = -(r_{y_i})_{j-1} \text{ and } u_{y_i} = (r_{x_i})_{j-1}$$

This normal vector representation, which includes the determination of either a right-handed or left-handed coordinate system, is defined in appendix III of this section.

The vector $\hat{b}_i$ is more difficult to obtain since we need to find a line from $(i, j-1)$ that will be normal to an $\vec{s}_l$ segment of the j line. This value of l will not necessarily equal i, and can change for each iteration. We define

$$\hat{b}_i = \pm(b_{x_l}\vec{i} + b_{y_l}\vec{j})$$

where the direction cosines represent the normal vector to the average of $\hat{s}_{l-1}$ and $\hat{s}_l$, along the j line. These coefficients are derived in a similar manner to those of $\hat{u}$, with $(i, j-1)$ replaced by (l, j), i.e.,

$$b_{x_l} = -(r_{y_l})_j \text{ and } b_{y_l} = (r_{x_l})_j$$

Appendix II of this section shows the derivation of l for a general vector $\vec{v}$ acting from node $(i, j-1)$. To obtain l for this case, we replace $\vec{v}$ by $\hat{b}_1, \hat{b}_2, \ldots$ until the correct $\vec{b}_l$ is found. We can now find $\hat{t}_i$ from equation (16). It was explained earlier that s'_i occurs at the intersection of $\vec{t}_i$ and $\vec{s}$ along the j line. This intersection is not necessarily in the i to $i + 1$ segment and the appropriate $\vec{s}_l$ segment must be obtained. The analysis described can again be used to determine l, but in this case the general vector $\vec{v}$ is replaced by $\vec{t}$. Since the calculation of l also calculates the length $|AA'|$, we can now evaluate $s'_i = s_l + |AA'|_l$.

1.4 Enforcement of Boundary Conditions

The ability to modify the adaption techniques in boundary regions substantially improves the flexibility of the adaptive scheme. The adaption domain is defined by the user and may be equal to, or a subset of, the physical grid. A boundary is defined as a line juxtaposed to the limits of the adaption domain defined by the user. There are two types of boundaries: marching boundaries (i.e., all points along the initial and final adaption lines) and edge boundaries ($i = i_{st}$ and $i = i_{end}$). Figure 5 shows an example of an adaption domain as a subset of the physical grid, and illustrates the two types of boundary lines when marching in the j direction. Note that if marching had been in the i direction, the boundary definitions would have been reversed.

By amending the previously defined variables of C_t, t_n, and λ in the boundary regions, it is possible to maintain physical characteristics at boundaries, allow for multiple passes, provide continuity across grid boundaries, simplify multiple grid adaptions, etc.

1.4.1 Treatment of initial marching boundary line. If the initial adaption line is within the physical domain, a smooth transition will be required across the starting internal boundary. As an example, this situation occurs when adapting in zones: each zone has different flow features and the user may wish to march up to a certain line using one set of parameters, then continue marching using a new set of control values. The common boundary across the two zones must remain unchanged when starting the second adaption pass. This feature is controlled by the input parameter, m_g. When $m_g > 0$, a smooth transition from external grid lines (e.g., those in the already adapted zone) to internal lines is created by maintaining the same grid points along the initial line, and then proportionally introducing the input adaption parameters. For example, when stepping to the second adaption line, we maintain as much straightness as possible by setting $C_t = 1$ (see eq. (16)). At each subsequent line, C_t is gradually decreased until it equals the user-supplied input value. The number of lines stepped before the full effect of the new parameters is felt depends on m_g: C_t will linearly decrease until m_g lines have been adapted. An example of this can be seen in figure 5, where $j_{st} = 4$ and $m_g = 5$. The grid points along $j = 4$ will remain unchanged while those along $j = 8$ will be fully adapted to the input control parameters. The code controls the adaption parameters for lines in between. Consequently, we define a variable, n_m, as

$$n_m = \frac{m_g + j_{st} - j}{m_g} \tag{18}$$

and then replace the value of C_t used in equation (16) by

$$C_{t_m} = C_t(1 - n_m) + n_m$$

At the same time, the value of λ is increased to $\lambda(1 + 5n_m)$ (thus increasing the magnitude of the torsion term) so that this amendment to C_t is effective. After m_g lines, $n_m = 0$, thus returning C_t and λ to their original values. Note that the computation of $\hat{e}$ along the $j_{st} + 1$ line is a function of the $j_{st} - 1$ line (if it exists), and this will also help in the merging process.

1.4.2 Treatment at final boundary line. Another discontinuity will occur between the final adaption line and any subsequent lines external to the adaption domain. It is not possible to use the same merging concept described earlier. If requested (through the MARCH input parameter), the grid points in the remaining external lines will be redistributed with the same proportions as the points on the final adapted line. This is not an adaption to the flow field, but provides a more aesthetic result.

1.4.3 Treatment of side-edge boundary. A smooth transition will also be needed at the side-edge boundaries if the adaption domain is internal to the given grid, i.e., $i_{st} > 1$ or $i_{iend} < imax$. Figure 6 shows an example where the fixed external grid spacing is denser than the first redistributed points, giving a discontinuous effect across the boundary. If the input parameter $n_g \neq 0$, a modification is requested to maintain some continuity of grid spacing across the side-edge boundaries. Consider the start-edge boundary at $i = i_{st}$ along line j. We wish to enforce some value on ω_1 that will give a value of Δs_1 close to the value of $\Delta s_{i_{st}-1}$. To do this, we find the average $\omega\Delta s$ along the converged $j - 1$ line and replace ω_1 by $\overline{\omega\Delta s}/\Delta s_{i_{st}-1,j}$. This value remains fixed during the iteration, but is merged into the updated values of ω_2, ω_3, and ω_4. The end-edge boundary is handled in a similar manner.

There is often the need to improve the side-edge spacing, even when the adaption domain coincides with the physical domain. If an adaption pass generates inappropriate side-edge spacing, a rerun of the code with n_g set will attempt to improve the spacing by using the above technique. However, in this case we do not have an external Δs, and $\Delta s_{2,j}$ is used instead of $\Delta s_{i_{st}-1,j}$. This will usually prevent the spring constants from pulling the lines too far off the boundary.

The variable n_g can be set to initiate computation for either or both side edges.

1.4.4 Treatment of orthogonality at boundaries. The code assumes that the grid lines should be as orthogonal as possible to a marching boundary. To accomplish this, the normal vector $\hat{n}$ is emphasized over the straightness vector (by decreasing C_t) when close to a marching boundary line. For even greater control, the coefficient t_n is modified to emphasize either $\hat{u}$ or $\hat{b}$, depending on whether the initial-line or end-line boundary is being considered. To ensure that the modifications to these torsion coefficients sufficiently effect the computation, the value of λ is simultaneously increased to accentuate the torsion term. Orthogonality at side-edge boundaries has been covered earlier in the definition of $\vec{s}$.

Since not all marching boundaries are physical boundaries, an input option is available that will override this emphasis on orthogonality for either boundary.

1.5 Discussion

1.5.1 Flexibility and segmentation. The vectorial approach used in the analysis provides for great flexibility in the use of the SAGE code. The user has complete choice of adaption direction and order of sequential adaptions without concern for the computational data structure. Multiple passes are available with no restraint on stepping directions: for each adaptive pass (or sweep) the user can choose a completely new set of adaptive parameters. This facility, combined with the capability of edge-boundary control, enables the code to handle multidimensional multiple grids. Zonal grids can be adapted while maintaining continuity along the common boundaries. For patched grids, the multiple-pass capability enables complete adaption. It was mentioned in the introduction that this adaptive technique does not produce a unique grid. This nonuniqueness enables the user to choose the most appropriate solution to the grid adaption.

1.5.2 Limitations and future improvements. The current version of the code does not always preserve input-wall boundaries. Each line is adapted in the computational domain and the points are interpolated back to the physical domain; in certain applications, interpolation error from sparse grid points or large surface gradients can prevent the wall boundary from resuming its initial shape. An option for inputting or algebraically defining physical boundaries will be available in the next version of the code.

Another area that will be addressed is the special needs of applications with periodic boundaries. These include C- and H-grids for compressor/turbine, rotor-stator problems. The requirement is that grid points on the initial and final boundary lines be equivalent. Since this adaptive procedure is a marching scheme, new iteration procedures will be required to enforce this condition.

1.6 Appendices

1.6.1 Appendix I: Derivation of A and B.

1.6.1.1 Calculation of A. We wish to relate A to the input values of Δs_{MIN} and Δs_{MAX}. A is constant throughout the entire mesh, and hence the 1-D relationship given in equation (2) holds. From its original definition in equation (9), $\bar{f}_{max} = 1$ and $\bar{f}_{min} = 0$; therefore, from equation (3) we have $\omega_{max} = 1 + A$ and $\omega_{min} = 1$. From equation (2) (rewritten as $\Delta s_i = K/\omega_i$) we can see that the minimum Δs will occur at K/ω_{max}. Similarly, the maximum Δs occurs at K/ω_{min}. We therefore wish to enforce $\Delta s_{MIN} = K/(1 + A)$ and $\Delta s_{MAX} = K$. These can be solved simultaneously: by eliminating K we have the expression given in equation (10)

$$A = \frac{\Delta s_{MAX}}{\Delta s_{MIN}} - 1$$

1.6.1.2 Calculation of B. This calculation is more complex and B is found by an iterative procedure. In addition, B will change for each j line. The objective is to determine which value of B will give the minimum computed Δs_i equal to the requested minimum, Δs_{MIN}. For each value of B there exists a minimum Δs_i. (An example of a plot of B vs. Δs_{min} is shown in fig. 7). We need to find B at the requested Δs_{MIN}. To do this, we assume an initial value of $B(= 1.0)$, evaluate ω, and then solve for new Δs using equation (5). Although equation (5) is true only for the initial line, it is assumed to hold for all j in order to simplify the B calculation. If $|\Delta s_{MIN} - min\Delta s_i^{(n)}|$ is small, an acceptable value of B has been found. If not, a new B is computed from $B^{(n+1)} = B^{(n)} + \Delta B^{(n)}$, ω is reevaluated, and the procedure repeated.

$\Delta B^{(n)}$ can be found from the definition

$$\frac{\partial}{\partial B} min(\Delta s_i) = \lim_{\partial B \to 0} \frac{(\Delta s_{MIN} - min\Delta s_i^{(n)})}{\Delta B^{(n)}} \tag{19}$$

As mentioned in the calculation of A, we know that Δs_i is a minimum when $\omega_i = 1 + A$, and by substituting this in equation (5) and differentiating, we obtain

$$\frac{\partial}{\partial B} min(\Delta s_i) = \frac{s_{max}}{1 + A} \frac{\partial}{\partial B}\left(\frac{1}{\psi}\right) \tag{20}$$

where

$$\psi = \sum_{l=1}^{n_i} \frac{1}{\omega_l}$$

We know that

$$\frac{\partial}{\partial B}\left(\frac{1}{\psi}\right) = -\frac{1}{\psi^2}\frac{\partial \psi}{\partial B}$$

Hence, the next step is to evaluate $\partial\psi/\partial B$. With this definition of ψ, we can take the summation sign out of the differential and obtain

$$\frac{\partial}{\partial B}\left(\sum_{l=1}^{n_i}\frac{1}{\omega_l}\right) = \sum_{l=1}^{n_i}\frac{\partial}{\partial B}\left(\frac{1}{\omega_l}\right)$$

$$= \sum_{l=1}^{n_i}\frac{\partial}{\partial B}\left(\frac{1}{1+A\bar{f}_l^B}\right)$$

$$= -A\sum_{l=1}^{n_i}\frac{\bar{f}_l^{\,B}\,log\,\bar{f}_l}{\omega_l^2}$$

Equation (20) can now be solved, and after substituting for

$$\sum_{l=1}^{n_i}\left(\frac{1}{\omega_l}\right) = \frac{s_{max}}{\Delta s_{min}(1+A)}$$

from equation (5), we obtain

$$\frac{\partial}{\partial B}min(\Delta s_i) = \frac{A(1+A)}{s_{max}}[\,min(\Delta s_i)\,]^2\sum_{l=1}^{n_i}\frac{\bar{f}_l^B log\,\bar{f}}{\omega_l^2}$$

Finally ΔB can be obtained from equation (19), giving B.

After the calculation of A and B, the correct distribution of ω is available for the general solution procedure.

1.6.2 Appendix II: The intersection of a vector, $\vec{v}$, with the streamwise vector, $\vec{s}$. This technique is used to obtain the $\hat{b}$ vector used in the orthogonality term, and to find the location of s_i'. To determine s_i' we need the streamwise segment in which the torsion vector $\hat{t}$ from the grid point $(i, j-1)$, intersects the j line. To evaluate the direction cosines of the $\hat{b}$ vector, we need to find which segment $(l \rightarrow l+1)$ along the j line will contain a normal vector acting from $(i, j-1)$. In both cases, we are dealing with the adaption at a nodal point (i, j) and the technique for finding l is the same.

Figure 8 shows a given vector $\vec{v}_i$ acting from node $(i, j-1)$ and intersecting $\vec{s}$ in the segment $(l, j) \rightarrow (l+1, j)$. D is at the point $(i, j-1)$ and $\vec{DA'} = |DA'|\hat{v}_i$. $\vec{AC}$ acts from $(l, j) \rightarrow (l+1, j)$ and $\vec{AC} = |AC|\hat{s}_l$. Since $\hat{v}_i$ intersects more than one $\vec{s}$ (as seen in the figure), we need to compute $|AA'|$ for each $\vec{s}_{l=1,2,3...}$ until $|AA'| < \Delta s_l$. When this condition is satisfied, l is the index of the correct streamwise segment, and $|AA'|$ and $|DA'|$ are also available from the same computational process.

To find $|AA'|$, we have (using vector addition)

$$|AA'|\hat{s}_l = \vec{AD} + |DA'|\hat{v}_i \tag{21}$$

From its physical location, we have

$$\begin{aligned}\vec{AD} &= (x_{i,j-1} - x_{l,j})\vec{\imath} + (y_{i,j-1} - y_{l,j})\vec{\jmath} \\ &= a_{x_l}\vec{\imath} + a_{y_l}\vec{\jmath}\end{aligned}$$

We can, therefore, rewrite equation (21) as

$$|AA'|(s_{x_l}\vec{\imath} + s_{y_l}\vec{\jmath}) = a_{x_l}\vec{\imath} + a_{y_l}\vec{\jmath} + |DA'|(v_{x_i}\vec{\imath} + v_{y_i}\vec{\jmath})$$

and, by equating coefficients of $\vec{\imath}$ and $\vec{\jmath}$, and solving the set of simultaneous equations, we obtain

$$|AA'| = \frac{a_{x_l}v_{y_i} - a_{y_l}v_{x_i}}{s_{x_l}v_{y_i} - s_{y_l}v_{x_i}} \tag{22}$$

$|AA'|$ is computed for $l = 1, 2, 3, \ldots$ until $|AA'| < \Delta s_l$.

At this point, $|AA'|$ and l are available. $|DA'|$ is now found from one of the simultaneous equations, i.e.,

$$|DA'| = \frac{|AA'|s_{x_l} - a_{x_l}}{v_{x_i}} \tag{23}$$

Note: if v_{x_i} is very small, use

$$|DA'| = \frac{|AA'|s_{y_l} - a_{y_l}}{v_{y_i}}$$

1.6.3 Appendix III: Definition of normal vectors. The handedness of a coordinate system is essential to define a unique normal direction and can be found by taking the cross-product of any two vectors acting at a point in the mesh. Choose any point in the center of the mesh (i, j) and define two vectors, $\vec{v}_1$, from $(i, j) \longrightarrow (i + 1, j)$ and $\vec{v}_2$, from $(i, j) \longrightarrow (i, j + 1)$. We know that

$$\vec{v}_1 \times \vec{v}_2 = (a_1\vec{\imath} + b_1\vec{\jmath}) \times (a_2\vec{\imath} + b_2\vec{\jmath}) = (a_1b_2 - b_1a_2)\vec{k} = c\vec{k}$$

If c is positive, we have a right-handed system, and if negative, a left-handed system. If c is found to be very small, another internal mesh point is chosen to ensure that numerical error is not a factor.

The normal, $\vec{N}$ to a vector $\vec{V} = a\vec{\imath} + b\vec{\jmath}$, is defined as $\vec{N} = \pm(-b\vec{\imath} + a\vec{\jmath})$, where the sign is chosen positive for a left-handed coordinate system and negative for a right-handed system.

2. SAGE USER GUIDE

2.1 Overview

The SAGE code is based on the self-adaptive grid method developed by Nakahashi and Deiwert (1985). This guide contains information that will enable the user to run the code with little knowledge of the mathematical concepts employed in the development of the adaption technique. Included is a detailed description of the input-control parameters, along with routine descriptions, nomenclature, etc. The code stands alone and has been run on many computer systems. Users wishing to understand and/or amend the code can find the details of the mathematical background in the first section of this document.

Briefly, the code reads two data files: one that contains the coordinates of the grid (x, y) and another that contains corresponding flow-field variables q. (It is assumed that these datasets are formatted as defined in the plotting software package, PLOT3D.) The grid is then adapted with respect to the flow-field variables as described by a control-file input by the user. The code returns the redistributed grid points and the corresponding interpolated flow variables in the same PLOT3D format.

The 2-D adaption takes place as a sequence of 1-directional adaptions. Along each grid line, (x, y) is transformed to a 1-D arc-length variable, s. An input-control variable determines the direction of the adaption. The redistribution of points is controlled by two parameters related to tension and torsion springs, and by the minimum and maximum allowable grid spacings. Figures 1 and 2 show examples of the grid adaption process. Assuming that adaption steps in the j direction, the tension-spring constants, ω, are evaluated at each point (i) along the current line (j), i.e., $\omega_i = f(s_{i,j})$, and are a function of the gradient of the user-chosen flow-field variable. The torsion parameter $\tau_i = f(s_{i,j}, s_{i,j-1}, s_{i,j-2})$ is the link between the current line and the previously adapted lines, thus providing control of orthogonality and smoothness constraints. A system of $(n_i - 2)$ equations is then developed for the constant j line and solved as a tridiagonal system. Once the adaption is completed for the current j line, the code steps to the next j line (either forward or backward) and repeats the same procedure. If the user requires adaption stepping in the i direction, the same procedure is carried out along lines of constant i.

It should be noted that by decoupling the 2-D system, the solutions are no longer unique. Backward or forward stepping will produce quite different grid solutions.

There are six major steps taken in the code:

1. Input of three data files: initial grid, flow-field solution, and user control parameters.
2. Initialization and reorganization of data for computational purposes.
3. Adaption along start line using 1-D technique.
4. For each subsequent j line:
   (a) computation of variables that define torsion and tension constants, and hence coefficients of the tridiagonal matrix equation (as defined in Section 1);
   (b) iterations to obtain new values of s, and hence (x, y).

5. Step 4 repeated until all requested lines are completed.
6. Output of new grid and interpolated flow-field files in original format.

Following this introduction are sections describing the user input parameters, execution commands, explanation of output messages, contents of each subroutine, a nomenclature set that relates code names to analysis variables, and a list of the major variables. Finally, many examples are given, along with input parameter lists, and grid and flow-field plots.

2.2 Execution of the SAGE Code

The following is an example of the command file required for the VAX/VMS system to assign the input and output files of the code. For other computer systems, users must appropriately assign the six files.

```
$ASSIGN SAGE.INP FOR005
$ASSIGN SAGE.OUT FOR006
$ASSIGN XY.GRD FOR007
$ASSIGN Q.FUN FOR008
$ASSIGN XY.OUT FOR010
$ASSIGN Q.OUT FOR011
```

where SAGE.INP contains the user-supplied adaption control variables, and SAGE.OUT is a message file output by the code.

The remaining files are in PLOT3D binary:

XY.GRD contains the initial grid points
Q.FUN contains the flow-field variables to which the grid is to be adapted
XY.OUT contains the adapted grid points
Q.OUT contains the flow-field variables interpolated on the adapted grid

<u>PLOT3D unformatted files</u>

The following are the read statements used for the binary files:

```
XY.GRD:
   READ (7) IMAX,JMAX
   READ (7) ((X(I,J),I=1,IMAX),J=1,JMAX),((Y(I,J),I=1,IMAX),J=1,JMAX)
Q.FUN:
   READ (8) IMAX,JMAX
   READ (8) FSMACH,ALP,RE,TIME
   READ (8) ((Q(I,J,N),I=1,IMAX),J=1,JMAX),N=1,NDIM)
```

where

IMAX = the number of points in the i direction in the grid file
JMAX = number of points in the j direction
NDIM = number of Q variables (default = 4)

As seen in the above format, four flow-field variables are expected in the Q file. PLOT3D preassigns $\rho, \rho u, \rho v$, and e, but since the SAGE code requests only the index of the function, any variables may be stored. Note that it is possible to handle more than four flow variables by changing NDIM in the parameter statement at the beginning of each subroutine.

Currently, the maximum dimension of IMAX and JMAX is 232. This value is set in the parameter statement at the beginning of each subroutine, and hence can be changed by a single edit command. Note that, since data may be switched internally, all 2-D arrays must be square and be greater than or equal to the largest of IMAX and JMAX.

FSMACH, ALP, RE, and TIME are not used in the code and may contain dummy values. They are part of the PLOT3D package and are displayed on the output plots.

XY.OUT and Q.OUT are written in the same format as the input GRD and FUN files.

2.3 User Input Parameters

SAGE.INP is the input-parameter control file. The grid adaption is based on the user's choice of these parameters, which are listed and briefly described below. A more complete explanation is given in the subsequent section. Although 23 control parameters are described, generally only a few need to be changed from the default value. Values in parentheses indicate default values.

In this version of the code, SAGE.INP is in namelist format $NAMEL. Namelist format implies that the variable need be input only if it is different from the default value. Since most control parameters maintain their code-generated values, namelist is a convenient method for the user-input control file. Although namelist is now a standard Fortran feature, users without this capability will need to change the input-read statement in the routine "*INITIAL*" and must ensure that all 23 parameters are given input values.

If more than one adaption pass is to be made (for example, 2-directional adaption), subsequent adaptions can be made on the "adapted" grid by linking together up to 10 sets of $NAMEL within the same SAGE.INP file.

Before describing the input parameters, some terminology needs to be clarified.

The term physical domain is used to reference the complete grid defined by the input grid file. Adaption domain is the part of the grid, as defined by the control file, that is to be adapted. These two domains can be equivalent. (See fig. 5, where the shaded area represents an adaption domain within the physical domain.)

The term adaption direction is used to define the 1-D line along which adaption takes place. The marching (or stepping) direction is the direction in which the adaption steps after each 1-D line adaption has taken place. In the case illustrated in figure 5, the stepping direction is j. The code makes no a priori assumptions of these directions until the control file has been read, since a user parameter defines whether i or j is the stepping direction.

Frequent reference is made to edge boundaries. It is assumed that two marching boundaries and two side edge boundaries exist along the outside limits of the adaption domain. Their definition depends on the stepping direction of the adaption: the marching boundaries are the first and last lines to be adapted; the side-edge boundaries occur at the first and last point of each adapted line. For example, in figure 2 with adaption in the j direction, the start marching boundary occurs at $j = j_{st}$; however, if adaption is in the i direction, this boundary will be at $i = i_{st}$.

2.3.1 Parameter control file, SAGE.INP.

$NAMEL

IST:	(1)	first adaption line in i direction
IEND:	(IMAX)	last adaption line in i direction
JST:	(1)	first adaption line in j direction
JEND:	(JMAX)	last adaption line in j direction
JSTEP:	(.TRUE.)	=.TRUE. for adaption to step in the j direction =.FALSE. for adaption to step in the i direction
INDQ:	(1)	indicates adaption flow-field variable q, default is first index, density
IQ(6):	(0)	enables a combination of q variables to drive the adaption
RDSMAX:	(2.0)	proportioned maximum allowed Δs, $\Delta s_{MAX} (\geq 1.0)$
RDSMIN:	(.5)	proportioned minimum allowed Δs, $\Delta s_{MIN} (\leq 1.0)$
CLAM:	(.01)	coefficient of torsion parameter constant, λ $(.05 - .000001)$
CT:	(.5)	proportion of "straightness" to "normal" for torsion parameter
NFILT:	(2)	number of passes to smooth q and ω
INTER:	(2)	2- or 3-pt interpolation
MGSTEPS:	(0)	number of lines to full adaption in the stepping direction
NEDGE:	(0)	controls side-edge adaption

		1=both edges, 2=start edge, 3=end edge
MARCH:	(.FALSE.)	=.TRUE. to extrapolate last adapted line throughout the remaining flow
ADD:	(0)	=integer to increase number of mesh points in requested range
LSTADD,LENDADD		requested range, used only if ADD$\neq 0$. If omitted, and ADD is set, the complete adaption range is assumed
NOUP:	(.FALSE.)	.TRUE. suppresses adaption, useful for adding points with no adaption
SAVE:	(.TRUE.)	=.TRUE. to store data for PLOT3D at completed adaption =.FALSE. to suppress the O/P of the adapted files
ORTHS	(.TRUE.)	setting this to .FALSE. will suppress orthogonality at the start marching boundary
ORTHE	(.TRUE.)	similarly, .FALSE. suppresses orthogonality at the end marching boundary

$

For the first adaptive effort, try using the default values and, if necessary, change the parameters that physically describe the system: for example, the boundary lines (IST, IEND, JST, JEND) and the adaptive flow-field variable, INDQ.

RDSMAX, RDSMIN, and CLAM are the first variables to investigate if this first pass is not satisfactory. If a catastrophic error occurs, significantly change CLAM. Note that CLAM has no effect on the initial line.

2.3.2 Explanation of user-controlled parameters.

ADD. When this is set, the number of points within the requested range (see LSTADD,LENDADD) is increased. For example, if ADD=2 , two points will be added between each consecutive grid point in the requested interval. Adding occurs only in the adapting direction and not the stepping direction. Be sure that the defined dimensions are not exceeded as a result of adding points.

CLAM, (λ), controls the magnitude of the torsion parameter constant, τ, which in turn controls the relative influence of the torsion term with respect to the tension term. This is the most difficult parameter to ascertain. Its order of magnitude can lie anywhere between 10^{-5} and 5×10^{-1}. A value of zero will turn off torsion, and the system of equations for the adaption line will be independent of its neighbor. A larger value of CLAM will decrease the influence of the flow-field gradients, and will therefore generate a smoother but less adapted grid. The magnitude of λ is related to how close the initial grid follows the high gradient regions.

CT, (C_t), is a variable that represents the direction of the torsion term, whereas λ is the magnitude in this direction. $0 \leq C_t \leq 1$, where $C_t = 0$ emphasizes orthogonality and $C_t = 1$ emphasizes "straightness." The default of 0.5 places the torsion vector halfway between. This value is suitable in most cases, but may cause problems when side boundaries are concave. (See the example section.)

INDQ indicates which of six possible flow-field variable(s) will drive the redistribution of grid points. Given the standard PLOT3D format

1 $\rightarrow$ density, ρ
2 $\rightarrow$ x-momentum, ρu
3 $\rightarrow$ y-momentum, ρv
4 $\rightarrow$ stagnation energy, e

In addition, two more options are computed by the code (assuming ideal gas and the above order of variables)

5 $\rightarrow$ pressure, p
6 $\rightarrow$ Mach number, M

The user will normally choose to adapt to the variable that is considered most representative of the flow-field features. However, if each flow-field variable demonstrates different features, it may be advantageous to combine them to bring out these same features on the adapted grid. In this case, set INDQ=0 and input IQ.

(The user may increase the number of variables by changing the value of NDIM from its default value of 4. In this case, the code assumes that indices 5 and/or 6 will contain user data, and that pressure and Mach number are indexed by NDIM+1 and NDIM+2, respectively.)

IQ is an array of six (NDIM+2) variables which are only used when INDQ=0. They allow the user to adapt to more complex functions. The order of the six variables corresponds to the flow-field variables (i.e., IQ(2) represents ρu). The value of the index represents the contribution of the variable to the adaption variable. For example, IQ(1)=1, IQ(5)=1 [i.e., (1,0,0,0,1,0)] will produce an adaptive function of $\frac{1}{2}(\rho + p)$, and IQ(1)=2, IQ(5)=3, IQ(6)=1 will produce $\frac{2}{6}\rho + \frac{3}{6}p + \frac{1}{6}$M. As another example, (1,0,0,0,0,0) is the same as INDQ=1.

INTER indicates whether to use a linear or quadratic Lagrange polynomial for interpolations. Interpolation is frequently required throughout the code: for example, each time a new s_i is computed, the corresponding flow-field values are interpolated.

IST,IEND contain the indices defining the first and last boundary lines of the adaptive domain in the i direction, and, similarly, **JST,JEND** define the j direction. These variables define the limits of the adaption domain, and they must lie within the input grid boundaries of the physical domain, i.e., $1 \leq$IST,IEND$\leq$IMAX and $1 \leq$JST,JEND$\leq$JMAX. Remember that the i and j directions always represent the first and second indices on the input XY.GRD file.

Reversing the adaption direction. Setting IST>IEND and/or JST>JEND will reverse the direction of the adaption. The reversing of the data is handled internally and is imperceptible to the user. Reversing the stepping direction will completely change the resulting grid, since the redistribution along the initial line, as well as the connecting torsion spring angles, will be quite different. Reversing the order in the nonstepping direction will have no effect on the solution, since the solution along a line is independent of the order of points. However, for meshes that contain very large and very small Δs_i, numerical accuracy may be influenced by the adaption direction. There is a more detailed description of data structure at the end of this section.

Figure 5 shows an example of these initial boundary lines where the input physical mesh has a dimension of (15,12) and contains an internal adaption domain. If the i direction is adapted from left to right, then IST=3 and IEND=12; however, if adapting from right to left, then IST=12 and IEND=3. Similarly for the j direction: JST=4,JEND=12 when adapting from bottom to top, and JST=12,JEND=4 when adapting from top to bottom. Note that these values are independent of JSTEP.

JSTEP controls the direction of stepping. JSTEP=.true. implies that stepping will occur in the j direction, i.e., on each j line, all i points will be redistributed before stepping to the next j line. Conversely, JSTEP=.false. will produce stepping in the i direction, redistributing all j points before stepping to the next i line.

LSTADD, LENDADD. These are input only if ADD$\neq$ 0 and if only a portion of the adaption region is to be expanded. If they are not input and ADD$\neq$ 0, the default is the entire adaption region. If ADD=0, their value is ignored.

MARCH. If the last line to be adapted (j_{end}) is within the physical grid boundary, a sharp discontinuity may show between this adapted line and the juxtaposed nonadapted line: setting MARCH=.true. requests realigning the remaining nonadapted lines (i.e., lines from $j_{end} + 1$ to j_{max}). This is not an adaption: the grid points are redistributed as proportional to the final adapted line.

MGSTEPS provides continuity when adaption starts at a line internal to the physical boundary. It enables the code to start with no adaption on the first line (i.e., the original distribution is retained) and linearly increases the adaption effect until, after MGSTEPS lines, full adaption occurs. At full adaption, the values of C_t and λ, etc. will coincide with the input values. If MGSTEPS =1, no adaption will be performed on the first line, but full adaption occurs along the second line. As an example, see figure 5. If JST=3 and MGSTEPS=4, no adaption will be performed on line 3, and full adaption will occur on line 7. See the boundary enforcement section in the analysis for details.

NEDGE is a flag that introduces internal control of side-edge boundaries. Side-edge conditions frequently need special handling. For example, when the side edge of the adaption domain lies internal to the physical grid boundary, the side-edge spacing should be continuous with the spacing in the external region. Even when the two boundaries coincide, the default value of zero may produce unsatisfactory results. In either case, NEDGE can be set and the code will try to improve the side spacing. Depending on the case, both or only one of the side spacings may need improving. NEDGE=1 requests both edges, NEDGE=2 requests start edge only, and NEDGE=3 requests end edge only.

NFILT is a "filtering" variable that smooths the gradient of the input q data. The default will generally suffice; however, if the flow-field variables are very discontinuous, it is necessary to increase the value of NFILT. For adaption to raw-input flow-field data, set=0. The tension parameters are also smoothed under the control of this parameter.

NOUP. This variable is convenient if the user would like to increase points within the grid, using the interpolation provided by ADD, without performing any adaption.

ORTHS,ORTHE. The code assumes that orthogonality to the marching boundaries is a desirable feature. However, for situations when this assumption is invalid, ORTHS and/or ORTHE can be changed to false, and adaption will be fully determined by the flow.

RDSMAX,RDSMIN control the density of the redistributed points and are the maximum and minimum allowable grid spacings. They are input as proportioned values and are changed to physical variables internally, i.e., $\Delta s_{MAX} \times s_{max}/(n_i - 1)$ and $\Delta s_{MIN} \times s_{max}/(n_i - 1)$, where n_i is a constant equal to the total number of points along the adapted line, and s_{max} is the length of the current adaption line. This implies that RDSMAX $\geq$ 1.0 and RDSMIN $\leq$ 1.0. (If both are set to 1.0, a totally uniform grid will result.) As an example, RDSMIN=0.5 will prevent a converged Δs from being less than half the average step size. (Since many factors influence the distribution of grid points, this control is not absolute.) Note that, since the adaption along the first line is not influenced by torsion (CLAM), this initial line will present a clearer picture of the effect of RDSMAX and RDSMIN.

SAVE. This is useful when more than one adaption pass is made in the same program run. The output grid and function files are large, and setting SAVE=.false. on any set of $NAMEL will suppress the output for that particular adaption. If SAVE=.true., each subsequent output set of XY.OUT and Q.OUT files will be assigned to different unit numbers. As stated in the execution section, the first output set is assigned to units 10 and 11. The second output set will therefore be units 12 and 13, etc.

Comment on internal reordering of data. The choice of stepping and adaption directions is a powerful feature in this code. The order of the (x, y) variables is determined by the input grid file, i.e., the first index represents the i direction and the second, the j direction. The user chooses the values of IST, JST, IEND, JEND, and JSTEP, based on the physical grid; however, in the code itself it is always assumed that the stepping direction is j, and that both i and j increase monotonically. Since the user can request adaption in any order, it may be necessary to internally rearrange the data in the physical domain (both grid and flow-field input files) to correspond with the assumptions. This rearranging is imperceptible to the user.

Rearrangement takes place for each element (i, j) of x, y, and q if

1. $i_{st} > i_{end}$, then each index (i, j) is replaced by $(imax - i, j)$
2. $j_{st} > j_{end}$, then each (i, j) is replaced by $(i, jmax - j)$
3. adaption is requested in the i direction, then $x_{i,j}$ and $y_{i,j}$ are interchanged, and $q_{i,j} \rightleftharpoons q_{j,i}$

At this point, the code also organizes the data such that the index parameter i along the adaption line always runs from 1 to n_i (even if $i_{st} \neq 1$), where $n_i = i_{end} - i_{st} + 1$. However, j is not changed and runs from j_{st} to j_{end}.

2.4 Output Message File

The SAGE.OUT file contains messages that may help to explain what has happened. At the end of each adaption, "ADAPTION n COMPLETE" indicates that the program was able to run to completion, and that XY.OUT and Q.OUT files have been created.

NUMBER OF POINTS INCREASED FROM n_1 TO n_2. (Informational)

If the ADD option has been input, this message gives the new dimension of the mesh.

ADD OPTION EXCEEDS DIMENSION. (Critical)

Self-explanatory. Increase array dimension by substituting for 232.

UNABLE TO DETERMINE LHS OR RHS. (Critical)

To compute the vectors required in the calculations, the mesh is tested to see if a left-handed or right-handed system is required for the stepping direction. If this error occurs, check the initial input grid.

NO CONVERGENCE ALONG INITIAL LINE, ERR=a_1. (Warning)

The initial line is a 1-D adaption only. This may not be a catastrophic error, especially if a_1 is small; however, the adaption may not be completely satisfactory. The only control parameters that affect the initial line are RDSMAX, RDSMIN, and NEDGE.

NO CONVERGENCE ON LINE j, ERR=a_2. (Warning)

This message is only a warning and adaption continues. Even several of these messages may of be no concern as long as a_2 is small. If adaption is successfully completed, then check new mesh to see if it is acceptable.

NO CONVERGENCE OF B ON LINE j, B SET TO a_3. (Warning)

This message is only a warning and should have very little effect on the final grid. When B does not converge, it is set equal to the value of B at line $j - 1$.

S IS NON-MONOTONIC ON LINE j. (Critical)

This message will terminate the program. It indicates that the values of s_i at the completion of the iteration on line j are not monotonically increasing, thus implying cross-over of points. Since this is unacceptable, the program outputs the data. It is then possible to see the plots and reevaluate the control parameters.

2.5 Outline of Each Subroutine

The SAGE code consists of the following subroutines, listed here in alphabetical order. Arguments shown in boldface are computed within the subroutine. Some of the routine descriptions refer to the analysis section of this document.

ADDPTS(ADD,LSTADD,LENDADD,IINVERSE)

This routine is called by *MAIN* if ADD$\neq$ 0 on the input parameter file. Extra points (depending on ADD) are inserted between each of the input nodal points within the range LSTADD to LENDADD, by a linear interpolation.

ADDV(COSX1,COSY1,A1,COSX2,COSY2,A2,**COSX,COSY**)

This is a utility routine. The direction cosines of two unit vectors (COSX1,COSY1) and (COSX2,COSY2) are input parameters. The routine computes the direction cosines (COSX,COSY) of the unit vector representing the sum of the input vectors, proportioned by the coefficients A1 and A2.

CROSSV(XT,YT,XT1,YT1,DST,COSV,**DAP,AAP,ICROSS**)

Appendix III of section 1 contains the analysis used to develop this routine. COSV is the array containing the direction cosines of the vector $\vec{v}_i$ defined in that appendix. (XT,YT) are the coordinates of a j line, (XT1,YT1) are the coordinates of its $j-1$ line, and DST is Δs along j. The routine computes $s - s'$ (i.e., AA$'$) and DA$'$, the distance between s' and the corresponding node at $(i, j-1)$, as shown in figure 3. ICROSS is the array containing l, indicating the intersecting segment for each COSV.

DLENG(JL)

When NEDGE is set, the tension parameter ω is amended at the edges (i.e., at IST and IEND) to improve edge-boundary spacing. This routine computes DLENGS and DLENGE, which are used in the edge ω calculation (by routine WTEDGE). The values of DLENGS and DLENGE depend on whether the grid is defined outside of the adaption domain. JL indicates which j line is needed.

EDGEMG(VAR,L1,L2)

This is a utility routine. For various reasons, the value of some variables at the IST and/or IEND edges are overridden. In order to blend this different value into the calculations, this routine will merge the new value into the next three grid locations. VAR is the variable, L1 is the location to start the first-edge blending, and L2 is the end edge.

FBAR(J)

This routine is called once for every j line. The Δs and the gradients $\partial q/\partial s$ are computed at the input grid points and stored in F. The routine *GETB* is called to find the value of B for this j line, based on the input grid. For lines other than the first, initial guesses for the s distribution are extrapolated from the converged s_i along $j-1$ line. Since these do not correspond to the input points, new local values of x and

y (XJ,YJ) are interpolated. Routine INTF is called to interpolate the value of F at these new grid points and to compute the normalized form of F, i.e., FB.

FILTER(VAR,NIPTS,NFILT)

This routine is called to smooth $\partial q/\partial s$ and ω. NFILT is the number of smoothing passes (default=2).

GETB(J)

This routine computes the value of B used to evaluate ω. B is found by an iterative process and is said to converge when the minimum requested Δs equals the computed minimum Δs. The analysis for this routine is given in appendix I of section 1.

GETDSM

This routine computes ω_t, the modifier of ω that is set when any computed Δs lies outside the requested range.

INITIAL(**NOMORE**)

This routine sets all the default values and reads the input parameter file. When necessary, the grid points and corresponding flow-field data are rearranged to correspond to the data organization assumed by the analysis. In addition, it is determined if the coordinate system is left-handed or right-handed. If no more adaptions are requested, NOMORE is set and returned to the main routine.

INTF(F,**FB**,SN,SMS,NPTS,IFL)

This routine interpolates to find $\bar{f}$ (FB) at the new s_i (SN), based on the input values of f (F) and the midpoints (SMS) of the input s array. This means that throughout the code, $\bar{f}$ is evaluated at the current values of s, and f is evaluated at the input grid points. IFL is a flag that indicates that no interpolation is required.

LAGCOF(SNEW,SARR,NPTS,**M**,**P1**,**P2**,**P3**)

This routine computes the Lagrange coefficients P1, P2, and P3 for a point SNEW with respect to the input s array, SARR. These are used by the calling routine to interpolate for the variable at SNEW. First or second order is available, based on INTER. M is the associated index for P1, and will reflect a forward or backward interpolation, depending on the location of SNEW within the interval.

LINE1

This routine solves for the adapted values of s_i along the initial line $j = j_{st}$. Since the torsion term is zero on this line, the Δs_i are computed from the 1-D approach.

MARCHES

If the final adapted line is internal to the physical boundary, this routine will redistribute the points on the remaining j lines, based on the distribution along the j_{end} line. This is performed only on request by the user and is not an adaption to the flow field. However, it will prevent the discontuity between the last adapted line and the outer boundary.

OUTPUT(NADS,OK)

OUTPUT is called after all lines have been adapted. Data is reorganized to conform with the input mesh structure and then written into the grid and flow-field files. NADS indicates which adaption number has been completed (maximum of 10 sets) and OK indicates that a normal completion has been reached. This routine is also called if s becomes nonmonotonic (OK=false). In this case, the new mesh points that have been computed are output to help the user determine the problem.

SETUP(J)

SETUP computes the direction cosines of the vectors $\vec{u}$ and $\vec{e}$ used to evaluate the torsion vector $\vec{t}$. Also computed are the modified values of torsion parameters C_t,λ, and t_n.

SOLUT(J,**OK**)

When SOLUT is called, all variables have been computed that are needed to obtain the new distribution of s_i. The coefficients of s_i (see eq. (8) in the first section of this report) are set up in a tri-diagonal matrix and solved for s_i. Interpolated values of ω_i are found at these new s_i and an iteration is performed until $\sum|(s_i^{(n)} - s_i^{(n-1)}|$ is small or too many iterations have been performed. In both cases, a check is made to see if all the s_i are monotonic. If so, the program continues; if not, the flag OK is set to false, causing the program to output the current grid and terminate.

SWAP

This routine is called if $j_{st} > j_{end}$ or $i_{st} > i_{end}$. The order of (i,j) in the x and y input matrices are reorganized to ensure that internal computations have monotonically increasing indices.

SWAPXY

Since the internal computation assumes that j is the stepping direction, this routine is called to interchange x, y, and q when stepping is requested in the i direction.

TORSION(J)

This routine computes the direction cosines of the normal vector $\hat{n} = f(\hat{b},\hat{u})$ and subsequently the torsion vector $\hat{t} = f(\hat{n},\hat{e})$. The routine *CROSSV* is called to find the intersection of $\hat{t}$ and the j line from which s_i' can now be evaluated. Finally, a check is made to ensure that s' monotonically increases. If it does not, the code attempts to correct this, but any severe problems will cause the code to terminate in the *SOLUT* routine.

UNITV(X1,Y1,X2,Y2,**DIRCX,DIRCY**)

This is a utility routine that finds the unit vector from (X1,Y1) to (X2,Y2). DIRCX and DIRCY are the direction cosines of this vector.

UPDATE(J)

UPDATE is the last routine called in the iteration loop for the current j. Newly adapted values of s_i have been found. The values of x, y, and q at this new distribution are interpolated and replaced into the matrices containing the physical mesh. In addition, these values of x and y are stored locally in XJ1 and YJ1 for the next line adaption.

VMERGE(DIRV,LST,LEND)

This routine performs the same function as *EDGEMG*, but with a vector quantity (DIRV) in place of a scalar value. LST and LEND indicate which value of DIRV needs to be merged into the next three points.

WTEDGE(J)

This routine is called by *MAIN* when NEDGE modification is requested. The edge values of the tension parameter at the next j line are a function of the average $\omega\Delta s$ along the just completed adapted line. This routine calls *DLENG* to obtain the appropriate value of edge Δs on the next line, computes the average $\omega\Delta s$ on this line, and evaluates WDS and WDE to be used in routines *LINE1* and *SOLUT*.

2.6 Nomenclature

The following is a list of variables used in the formulation of the SAGE methodology. The boldface characters represent the corresponding variable name used in the Fortran code.

A, B	constants used to compute ω, **(A,B)**
C_t	input proportion coefficient for torsion, **CT**
C_{t_m}	modified value of C_t, **CTM**
$\vec{b}$; (b_{x_i}, b_{y_i})	normal vector to j line; direction cosines of $\vec{b}$, **COSB**
$\vec{d}$; (d_{x_i}, d_{y_i})	straightness vector, **COSD**
$\vec{e}$; (e_{x_i}, e_{y_i})	average straightness vector, **COSE**
f	gradient of q, **F**
f_{min}, f_{max}	minimum and maximum f used to normalize f, **FMIN,FMAX**
$\bar{f}$	normalized function of f, **FB**
i	subscript indicating the current node in adaption direction, **I**
$imax, jmax$	total number of points in i and j directions of input mesh file, **IMAX,JMAX**
i_{st}, j_{st}	indices indicating start of adaption domain in i and j directions, **IST,JST**
i_{end}, j_{end}	indices indicating end of adaption domain, **IEND,JEND**
j	subscript of the current stepping line, **J**
l	a local subscript relating to node i, **L**
m_g	input value indicating number of stepping lines before full adaption, **MGSTEPS**
n_i	number of points in the adaption line, **NIPTS**
n_j	number of lines in the stepping direction, **NJPTS**
n_m	value of j at full adaption line $f(m_g)$, **CNM**
$\vec{n}$; (n_{x_i}, n_{y_i})	orthogonality vector, **COSN**
q	input flow-field variable $(\rho, \rho u, \rho v, e, P, M)$, **Q**
$\vec{r}$; (r_{x_i}, r_{y_i})	average of $\vec{s}_i$ and $\vec{s}_{i-1}$, **COSR**
s	streamwise length, **SS or SN**
$\vec{s}$; (s_{x_i}, s_{y_i})	vector representing s, **COSS**
s_{max}	maximum value of s on line j, **SMAX**
s'	value of streamwise length used in torsion computation, **SP**
Δs_i	$s_{i+1} - s_i$, **DS**
$\Delta s_{MIN}, \Delta s_{MAX}$	requested minimum and maximum grid spacings, **DSMIN,DSMAX**
$\Delta s_{min}, \Delta s_{max}$	computed minimum and maximum grid spacings
T	torsion force
$\vec{t}$; (t_{x_i}, t_{y_i})	total torsion vector, **COST**
t_n	proportion of $\vec{b}$ and $\vec{u}$ used to compute $\vec{n}$, **TN**
$\vec{u}$; (u_{x_i}, u_{y_i})	vector normal to $j-1$ line, **COSU**
x, y	input grid mesh, **(X,Y)** globally and **(XJ,YJ)** locally
λ	input magnitude of torsion control parameter, **CLAM**
ω	tension (weighting) parameter, **WEIGHT**
ω_t	computed weighting on ω, **WT**
τ	torsion related parameter, **TAU**

2.7 List of Major Variables

This section contains the list of variable names (in alphabetic order) used in the SAGE code. Local variables that contain only intermediary values are not listed. The format is given as:

Variable name(dimension) $/r_1/r_2/$ brief description

r_1 describes what type of variable—input, local, parameter, or name of common block.

r_2 lists routine(s) where variable is initialized.

A	/COM2/INITIAL/ A used to compute ω
AA(IJD)	/local/SOLUT/ coeff. of s_{i-1} in solution matrix
AAP(IJD)	/COM9/CROSSV/ $s - s_i'$
ACT	/local/TORSION/ final modified C_t
ADD	/COM13/input/ if set, add grid points
ALPHA	/COM12/input/ information only for PLOT3D
AMACH(IJD)	/local/FBAR/Mach number
B	/COM2/GETB/ B used to compute ω
BB(IJD)	/local/TORSION/ coeff. of s_i in solution matrix
BCONV	/local/GETB/ convergence criteria for B iteration
BJ1	/local/GETB/ value of B along $j - 1$ line
CLAM	/COM10/input/ λ, magnitude of torsion
CLAMW	/COM10/SETUP/ modified CLAM
CNM	/local/SETUP/ λ modifier when MGSTEPS$\neq$ 0
CONV	/COM15/INITIAL/ general convergence criteria
COSB(IJD,2)	/local/TORSION/ direction cosines of $\vec{b}$, normal to j line
COSD(IJD,2)	/local/SETUP/ temporary vector, $\vec{d}$
COSE(IJD,2)	/COM7/SETUP/ straightness vector, $\vec{e}$
COSS(IJD,2)	/local/SETUP/ streamwise vector, $\vec{s}$
COST(IJD,2)	/local/TORSION/ torsion vector, $\vec{t}$
COSU(IJD,2)	/COM7/SETUP/ $\vec{u}$, normal to $j - 1$ line
CRX,CRY	/local/SETUP/ coeffs. of $\vec{r}$, used to compute $\vec{u}$
CT	/COM10/input/ C_t, direction of torsion vector
CTM	/COM10/SETUP/ modified C_t
DAP	/argument/CROSSV/length DA' for s' calculation
DLENGE	/COM6/DLENG/ Δs computed for end-edge modification
DLENGS	/COM6/DLENG/ Δs computed for start-edge modification
DMINSDB	/local/GETB/ $\partial\, min(\Delta s)/\partial B$
DS(IJD)	/COM3/FBAR,LINE1,SOLUT/ $s_i - s_{i-1}$
DSM(IJD)	/COM6/GETDSM/ ω_t, modification to ω to maintain mesh spacing
DSMAX	/COM6/FBAR/ Δs_{max} from RDSMAX
DSMIN	/COM6/FBAR/ Δs_{min} from RDSMAX

DSP(IJD)	/local/TORSION/ $s'_i - s'_{i-1}$
F(IJD)	/COM2/FBAR/ flow gradient, $f = \partial q / \partial s$, at input s
FB(IJD)	/COM2/INTF/ $\bar{f}$, normalized f at current s
FF(IJD)	/local/SOLUT/ coeff. of RHS of solution matrix
FMAX,FMIN	/local/INTF/max. and min. f
FSMACH	/COM12/input/ information only, for PLOT3D
ICROSS	/argument/CROSSV/ index for interval location of s'
IEND	/COM4/input/ last node along i in adaption domain
IINVERSE	/COM13/INITIAL/ implies that adaption along decreasing values of i
IJD	/dimension/parameter statement/ currently set at 232
IMAX	/COM4/input/ no. of points in physical domain, i direction
INDQ	/COM5/input/ index for q for adaption variable
INTER	/COM11/input/ order of interpolation, 2 or 3
IQ(6)	/COM5/input/ related to INDQ, proportions q adaption function
IST	/COM4/input/ first node along i in adaption domain
JEND	/COM4/input/ last node in j adaption domain
JINVERSE	/COM13/INITIAL/ implies that adaption along decreasing values of j
JMAX	/COM4/input/ no. of points in physical domain, j direction
JST	/COM4/input/ first node in j adaption domain
JSTEP	/COM13/input/ indicates whether adaption steps in i or j direction
LENDADD	/argument/input/start of ADD range
LSTADD	/argument/input/end of ADD range
MARCH	/COM13/input/ if set, interpolation continues up to physical boundary
MAXITS	/COM15/INITIAL/ max. no. of iterations for convergence
MG1	/COM9/INITIAL/ flag for NEDGE at i_{st}
MG2	/COM9/INITIAL/ flag for NEDGE at i_{end}
MGSTEPS	/COM11/input/ no. of steps before full adaption occurs
NADS	/argument/MAIN/index on number of adaptions
NDIM	/dimension/parameter statement/ used for input q, currently set to 4
NEDGE	/COM9/input/ initiates side-edge boundary overide
NFILT	/COM11/input/ no. of passes for smoothing data
NFLAG	/COM11/INITIAL/ indicates left- or right-handed coordinate system
NITG	/local/OUTPUT/ unit numbers for output grid file
NIPTS	/COM4/INITIAL/ total no. of points in computation domain, i direction
NITQ	/local/OUTPUT/ unit numbers for output q file
NJPTS	/COM4/INITIAL/ total no. of points in computation domain, j direction
NOUP	/COM15/input/ prevents adaption
NOMORE	/argument/INITIAL/ when no more input datasets, ends program
OK	/argument/SOLUT/ indicates if solution available
ORTHE	/COM16/input/ remove orthogonality at outer boundary
ORTHS	/COM16/input/ remove orthogonality at initial boundary
P1,P2,P3	/argument/LAGCOF/ coeff. of Lagrange polynomials
PRES(IJD)	/local/FBAR/pressure
Q(IJD,IJD,NDIM)	/COM1/input/flow-field variables
RDSMAX	/COM5/input/ relative value of Δs_{max}

RDSMIN	/COM5/input/ relative value of Δs_{min}
RE	/COM12/input/ not used, PLOT3D variable
SAVE	/COM13/input/ flag to suppress output
SMAX	/COM6/FBAR/ max. value of s
SMS(IJD)	/COM3/FBAR/ midpoints of Δs
SN(IJD)	/COM3/FBAR/ current s array along j line
SNM(IJD)	/COM3/UPDATE/ converged s array along $j - 1$ line
SP(IJD)	/COM9/TORSION/ s', intersection of torsion vector and j line
SPC	/local/SOLUT/ $\lambda\omega_{max}$ used in torsion term
SS(IJD)	/COM3/FBAR/ initial value of s along j line
TAU	/local/SOLUT/ τ, torsion parameter
TIME	/COM12/input/ not used, PLOT3D variable
TN	/COM10/INITIAL/ proportion of $\vec{b}$ to $\vec{u}$
TNM	/COM10/SETUP/ TN modified for boundaries
TSPC	/local/SOLUT/ final torsion term in solution equation
WDE	/COM2/WTEDGE/ weighting factor used when NEDGE set at end boundary
WDS	/COM2/WTEDGE/ as WDE, but at initial boundary
WEIGHT(IJD)	/COM2/LINE1,SOLUT/ ω, tension parameter
WT(IJD)	/COM6/GETDSM/ ω_t, correction to ω
WTSUM	/local/GETB,LINE1,WTEDGE/ $\sum 1/\omega$
X(IJD,IJD)	/COM1/input,UPDATE/ input grid points, i direction
XJ(IJD)	/COM8/FBAR/ X at current SN
XJ1(IJD)	/COM8/UPDATE/ X at SNM
Y(IJD,IJD)	/COM1/input,UPDATE/ input grid point, j direction
YJ(IJD)	/COM8/FBAR/ Y at current SN
YJ1(IJD)	/COM8/UPDATE/ Y at SNM

3. EXAMPLES

This section contains several sets of examples to familiarize the user with the adaptive-grid process. Each example includes plots of the initial grid and flow-field contours, the input-control-parameter file used to adapt the grid, the resultant adapted grid and a discussion on the choice of control parameters. The first example in case 1 also gives the file assignments. In addition, two of the cases show the improved flow solution obtained from the flow solver using the adapted grid as input.

Case 1. Flow in a Supersonic Inlet

Figure 9(a) shows the 101 $\times$ 79 initial grid (assigned as input file INLET.GRD) for an inlet flow-field problem. Flow is from left to right and a shock emanates from the upper-wall corner, reflecting off the lower wall. In addition, an expansion fan originates from the downstream upper-wall corner and interacts with the reflected shock. The computed solution (INLET.FUN), generated by the flow solver using this initial grid, is shown as density contours in figure 9(b). The SAGE code is now run to create a new grid that is more adapted to this solution, i.e., the grid points are redistributed with respect to a chosen flow-field variable (in this case, density) and output to the file INLET.XY1 assigned to unit 10. SAGE also interpolates the complete flow-field solution onto these new grid points and outputs this file (INLET.Q1) to unit 11. The updated files will subsequently be input into the flow solver to produce more accurate solutions. Several examples of grid adaption are given for this inlet problem to demonstrate the effect of the control parameters.

Example 1: Single pass, stepping in j direction. The following are the job execution commands for this example on a DEC VAX computer:

```
$ASS CONTROL.INP FOR005
$ASS MESSAGE.OUT FOR006
$ASS INLET.GRD FOR007
$ASS INLET.FUN FOR008
$ASS INLET.XY1 FOR010
$ASS INLET.Q1 FOR011
```

CONTROL.INP contains the user-specified parameters. Until the user is familiar with the basic parameters, the first attempt may be an input-control file with no parameters; i.e., all default values are used. The choice of all default values for this example case is shown in figure 9(c). The grid has been more evenly spaced and there is a slight clusterering of points around the shock on the lower-wall boundary but the remaining grid lines are not adequately adapted to the flow features. This implies that the torsion parameter is too large and is overriding the local tension term, preventing the tension springs from pulling the points to the high-gradient regions. In addition, the minimum grid spacing is too large.

The input-control file was therefore chosen to contain:

$namel $rdsmin = .2 , clam = .001 , nedge = 1$$

Only three parameters are input: the minimum allowable grid spacing, the magnitude of the torsion term (an order of magnitude less than the default value), and a request for edge control (which frequently needs setting). The remaining parameters retain their default values, signifying that the full grid will be adapted, density is the adaption variable, and stepping is in the j direction. On completion of the execution, the MESSAGE.OUT file contains "ADAPTION 1 COMPLETE".

The result of this adaption is shown in figure 10(a). It can be seen that, since adaption began along the lower surface ($jst = 1$), the redistributed points cluster around the point of reflection on this line. However, points do not become sufficiently clustered at the corner shock and at the start of the expansion wave region on the upper surface. This is to be expected, since the adaption on the initial line is a 1-D solution and will pick up features quite clearly. However, as stepping continues, the features on subsequent lines get "dampened" by the torsion (i.e., smoothness) control.

Example 2: Adaption with backward j steps. This example maintains the same parameters as example 1, except that the upper surface is chosen as the initial adaption line, i.e., $jst = 79, jend = 1$. Figure 10(b) shows the result of this adaption. In this case, the upper surface is more clearly adapted, and the point of reflection on the lower surface is more spread out. Comparing these two examples indicates quite clearly that the starting line has a strong effect on the resultant adapted grid, and for some applications it may need to be chosen carefully.

Example 3: Adaption in i direction. This example shows the very different grid generated when the adaption is performed by stepping in the i direction ($jstep = .false.$). Based on the experience obtained from the first two examples, adapting from right to left will produce a better grid: there are no gradients on line $ist = 1$ to adapt to. Hence, the input control file was chosen to contain:

$namel $jstep = .f., ist = 101, iend = 1, rdsmin = .25, clam = .001, nedge = 1$$

Figure 10(c) shows the result of this adaption. Not surprisingly, the reflected shock region on the lower surface is not "captured" by this adaption, and thus this grid is not as suitable as those shown above. This demonstrates that the choice of stepping direction is important in producing a desirable grid.

Example 4: Effect of changing control parameters. This example is actually a collection of examples that show the effect of varying the control parameters $clam$, Ct, $rdsmax$, and $rdsmin$. Figure 11(a) shows an adapted grid using "baseline" values. These are the same as the default values in the code with the exceptions of $clam = .001$ and adaption proceeds from top to bottom (i.e., $jst = 79, jend = 1$). Figures 11(b) through 11(f) are grids adapted by changing just one of the baseline parameters. Figure 11(b) shows the result with $clam = .01$, and it is immediately obvious why this code default value was not chosen as the baseline value for these comparison cases: $clam$ is too large to allow any of the flow features to be "captured" by the adaption. Figure 11(c) shows the case for $clam = .0001$, and this smaller value produces a grid with points more clustered around the shocks. Figures 11(d) and 11(e) show the effect of the Ct parameter: $Ct = 1.0$ in figure 11(d), emphasizing straightness, and $Ct = .0$ in figure 11(e), emphasizing orthogonality. The latter shows how the orthogonality term dampens the adaption for this case: we are requesting orthogonality to the already parallel i lines. Finally, the effect of changing the minimum and maximum allowable grid spacing is shown in figure 11(f), where $rdsmax = 4.0$ and $rdsmin = .25$ are used. This mesh size control is only a factor of 2 different from that in the baseline example (fig. 11(a)), but significantly densifies the spacing in the shock regions.

Example 5: Two-directional adaption. Two-directional adaption is created by adapting in one direction (e.g., stepping in j) and then adapting in the other direction (i.e., i), using the first adapted grid as input. The resultant grid will depend on the order of the stepping direction. Two passes (i.e., two sets of input control parameters) were made to produce the adapted grid shown in figure 12(a). If both these passes are made during the same execution run (not to be recommended for a first attempt), then it is necessary to assign additional files for the output:

```
$ASS INLET.XY2 FOR012
$ASS INLET.Q2 FOR013
```

Data will therefore be output for both passes, unless the SAVE parameter is set to false.

The control file for this example contains:

$namel $clam = .001, rdsmin = .2, nedge = 1$ $
$namel $jstep = .f., ist = 101, iend = 1, clam = .001, rdsmin = .25, nedge = 1$ $

The first pass steps in the j direction and uses the same input control parameters as example 1, and thus produces the adapted grid shown in figure 10(a). The second pass steps in i, with the same parameters as example 3. Note that any parameters that remain unchanged for the two passes still need to be defined in the control file, since the code restores all default parameters at the conclusion of each pass.

The control file used to produce figure 12(b) contains the same parameters, but the adaption order is reversed. That is, the first pass steps in the i direction and the second pass in the j direction. Although the controls for the second pass are the same as example 1, the input grid is different, giving a quite different adapted grid. The difference between figures 12(a) and 12(b) is obvious. Although this example shows that both adaptive sequences do adapt the grid, it should be noted that it is also possible that one order of adaption will produce a completely unacceptable result, while the reverse is quite suitable.

Example 6: Flow solution using adapted grid. Figures 13(a) and 13(b) demonstrate the improved flow-field solution obtained when the flow solver is rerun using an adapted grid as input. The adapted grid shown in figure 13(a) was obtained by requesting a significant amount of clustering in the high gradient regions: $rdsmax = 4.0, rdsmin = .25$, and $clam = .0001$. In addition, adaption begins at the upper-wall surface. This grid and the interpolated flow-field variables were input to the flow solver, producing the flow solution (shown as density contours) seen in figure 13(b). The improvement in the resolution of the incident and reflected shock is obvious when compared to the initial solution shown in figure 9(b).

Case 2. Hypersonic Blunt-Body Flow

The second case contains two examples that demonstrate the effect of the CLAM and CT parameters. Figures 14(a) and 14(b) show an initial grid (32×32) and corresponding density flow-field contours for a hypersonic blunt body. The j direction marches out from the surface to the free-stream boundary, and the i direction marches along the body starting at the lowest point. This is a simple 1-directional problem with the shock shape aligned with the grid. The outer-side boundary is curved (when marching in the i direction) and the default values of $clam = .01$ and $Ct = .5$ will prevent the adapted grid lines from "turning" sufficiently at this edge, giving either a false densing of points at the outer side boundary or, possibly, a critical error message. In this situation, two remedies are available. CLAM can be decreased, hence de-emphasing the effect of torsion and thereby allowing the tension coefficient to "pull" the nodes toward the shock wave. Alternatively, the orthogonality restraint can be removed by setting $Ct = 1.0$. Figure 14(c) shows the result of setting $clam = .001$ and retaining all other parameters at their default value. The adapted grid obtained by setting $Ct = 1.0$ is very similar and is not shown here. The following are the two input-control files:

$namel $jstep = .f., clam = .001$ $ and $namel $jstep = .f., ct = 1.0$ $

Both the adapted grids show the required result, i.e., the clustering of grid points across the shock region.

Case 3. Blunt-Body Shock Impingement Problem

The third case introduces the use of IQ and ORTHE parameters. Figure 15(a) shows the input grid of a cowl/lip shock interaction problem. The j direction marches from the body surface to the outer free stream and the i direction marches from the lower point of the sphere. The flow-field contours of both density (fig. 15(b)) and Mach number (fig. 15(c)) are given, since the adaption flow-field parameter is a combination of the two. The blunt body shocks, impinging shock, and shear layers represent a complex flow that requires adaption in both directions to adequately capture all the flow features. Figure 15(d) shows the adapted grid produced by the following 2-pass control file:

$namel $jstep = .f., ist = 91, iend = 1, indq = 0, iq(1) = 2, iq(6) = 1,$
$rdsmin = .25, ct = .7, clam = .005, nedge = 2$ $
$namel $indq = 0, iq(1) = 2, iq(6) = 1, rdsmin = .3, orthe = .f.$ $

The first pass steps in the i direction, starting from the topmost boundary. The parameter $indq = 0$ indicates that IQ will be input, and in this case two-thirds of the density function and one-third of the Mach function will be combined to become the adaption variable. As explained in case 2, the curvature of the outer boundary requires increasing CT. Setting $nedge = 2$ requests that only the side-edge boundary at $j = 1$ be treated. The second pass introduces the use of the ORTHE variable: stepping occurs from the sphere wall to the outer curved boundary, where the default would turn the grid to be normal to this outer boundary. ORTHE overrides this and allows the adaption to occur naturally.

Case 4. Hypersonic Inlet (Zonal Adaption)

Figures 16(a) and 16(b) show the initial grid and density contours for a hypersonic cowl/lip inlet problem. This is a more complex case and requires dividing the adaption domain into zones—the blunt body region and the rectangular inlet region. The upper wall is the $j = 1$ line and the outgoing channel region (on the right side of the diagram) is the $i = 1$ line. Five adaption passes are required to create the final adapted grid shown in figure 16(c). The input control file consists of

$namel $rdsmin = 1.0, rdsmax = 1.0$ $
$namel $jstep = .f., rdsmin = .2, nedge = 1$ $
$namel $ist = 70$ $
$namel $jst = 32, jend = 1, iend = 71, rdsmin = .25, clam = .02, nedge = 1$ $
$namel $iend = 85, jst = 19, jend = 1, clam = .002, nedge = 1, mgsteps = 5$ $

The first pass spreads out the i grid lines evenly; the original grid points were more densely distributed in the curved section, leaving fewer grid lines in inlet area. The second pass steps in the i direction and adapts easily throughout the entire grid. The third, fourth, and fifth passes perform the adaption in the j direction. The very different flow features in the blunt-body region (blunt-body shock) compared to the features in the inlet region (Mach stem and reflected shocks) indicate dividing the adaption domain into these two zones: $i < 70$ and $i > 70$. Only the blunt-body section ($i > 70$) is adapted in pass three, with all default-control parameters. The adaption in the inlet domain starts at $jst = 32$ in order to pick up the Mach stem along the lower wall. After stepping through the triple-point region (the intersection of the cowl shock and the reflected normal shock), the redistributed points do not spread out sufficiently, so the adaption was stopped at line 19 and a fifth pass is started at line 19 with a decreased value of CLAM. Since this pass starts internally to the original adaption, it is necessary to set the MGSTEPS flag in order to merge smoothly from the already adapted region.

Case 5. Axisymmetric Plume Flow

This final case is presented to indicate the powerful effect of the adaptive grid process. Figure 17(a) shows the initial grid and computed solution (in Mach contours) of an axisymmetric nozzle-plume flow. This initial solution has not developed sufficiently to capture the final flow features: the outer shear layer, barrel shock, Mach stem, reflected shock, and the triple-point shear layer. Three iterations (through both adaption and flow solver) were made to produce the final adapted grid and solution shown in Figure 17(b). The definition of the flow features is greatly improved. Figure 17(c) points out the accuracy of the final solution: the lower picture is a shadowgraph of the actual experiment and is almost mirrored by the computed solution.

REFERENCE

Nakahashi, K.; and Deiwert, G. S.: A Self-Adaptive-Grid Method with Application to Airfoil Flow. AIAA Paper 85-1525, July 1985.

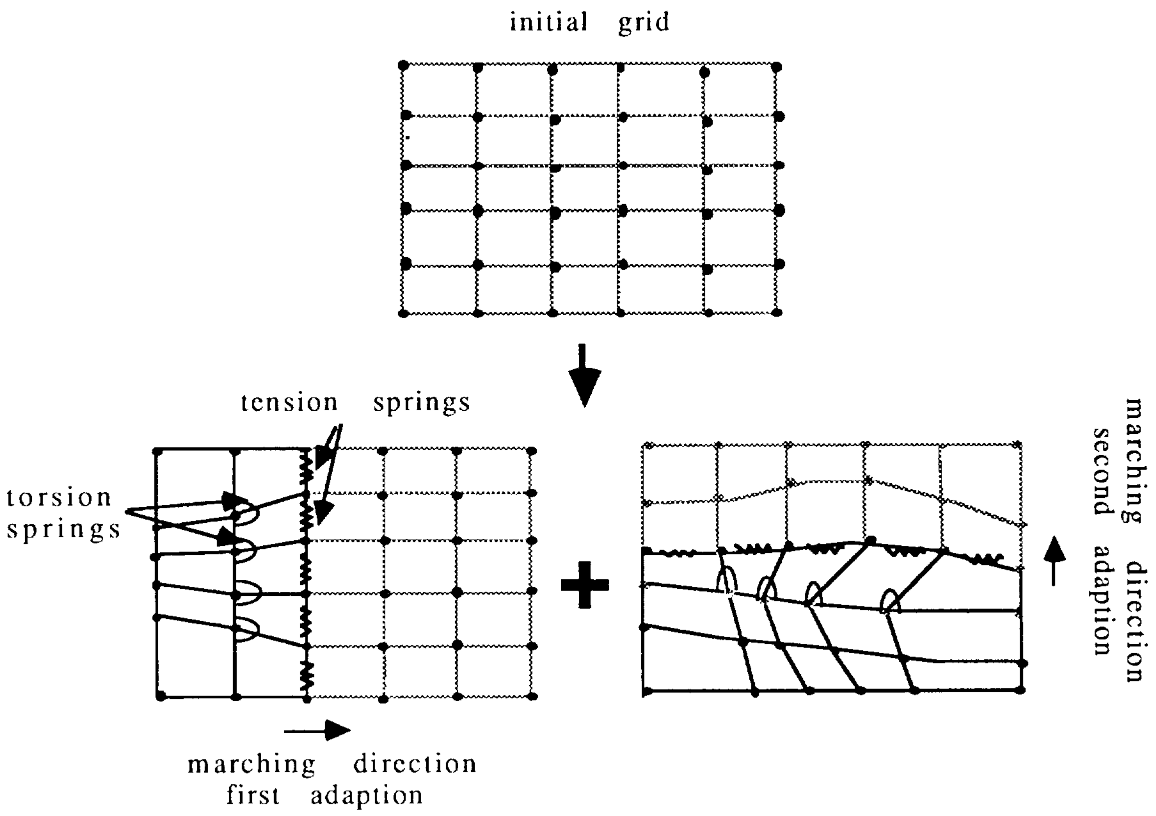


Figure 1.– Splitting of 2-D adaption into two 1-directional adaptions.

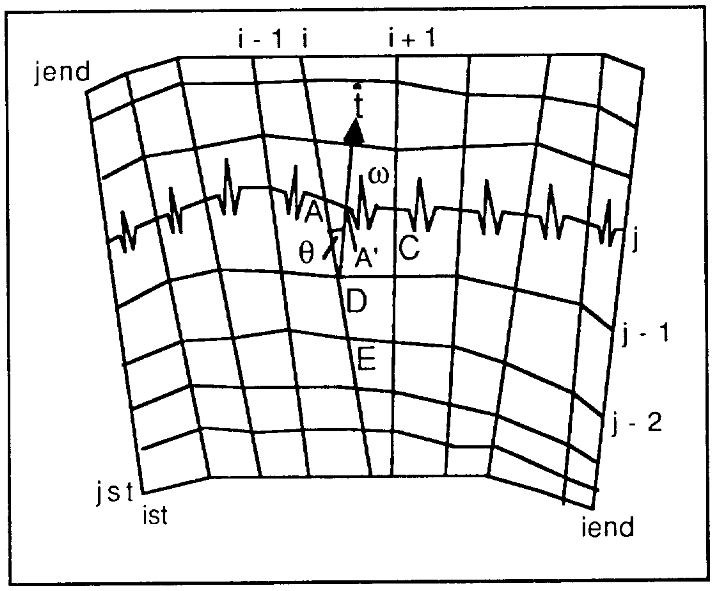


Figure 2.– Adaption line j, showing tension and torsion springs.

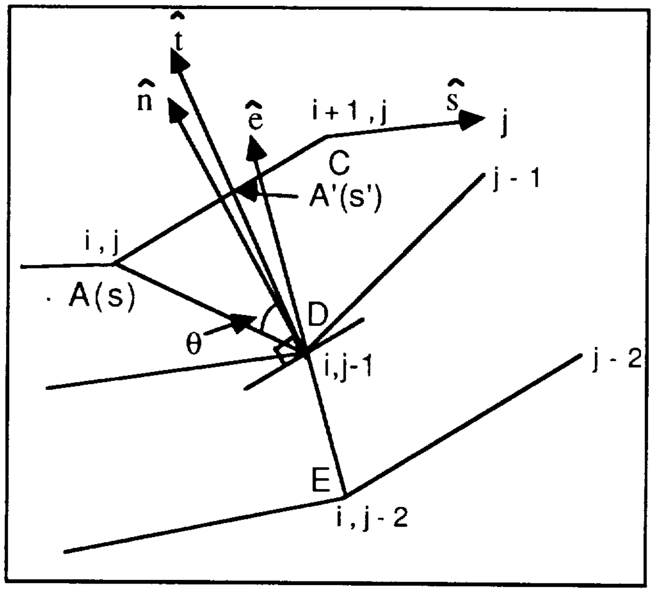


Figure 3.– Detail of point (i, j) showing torsion vector, $\hat{t}$.

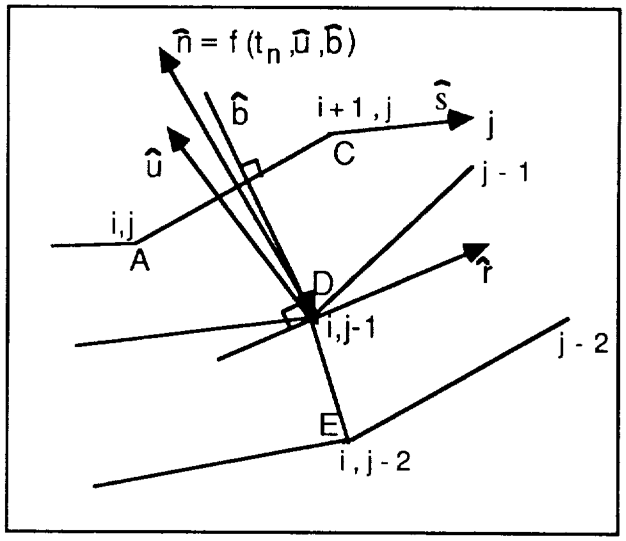


Figure 4.– Normal components of torsion vector.

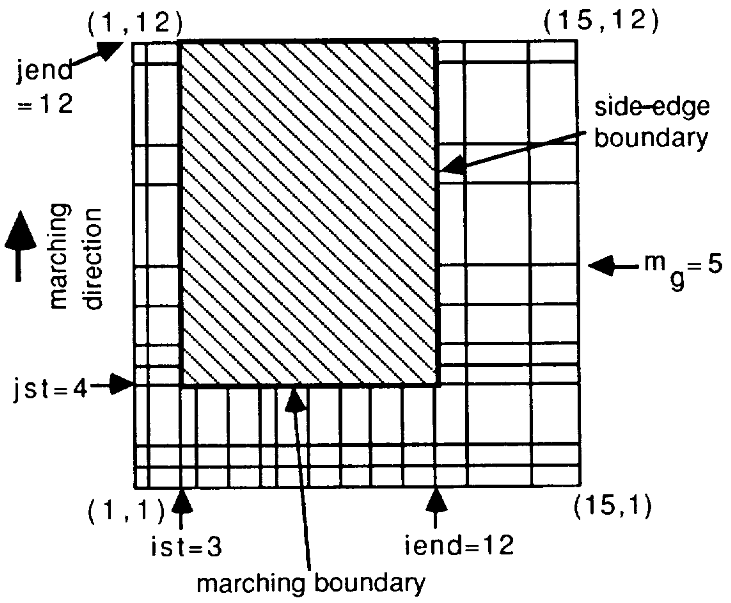


Figure 5.– Physical and adaption domains.

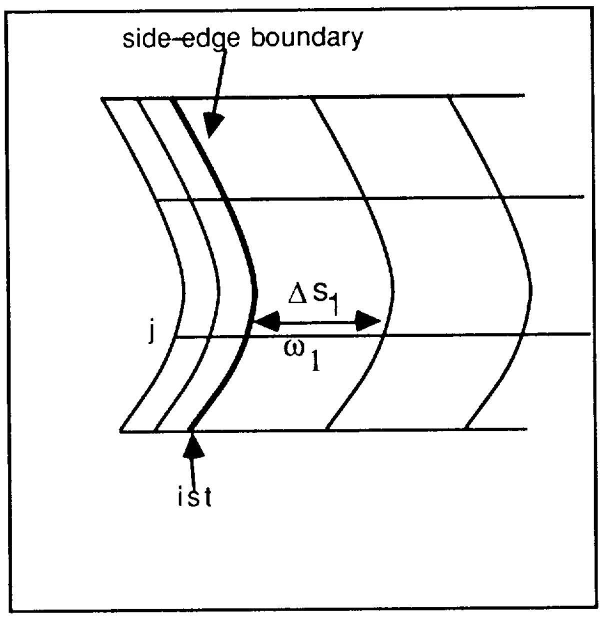


Figure 6.– Control of side-edge spacing.

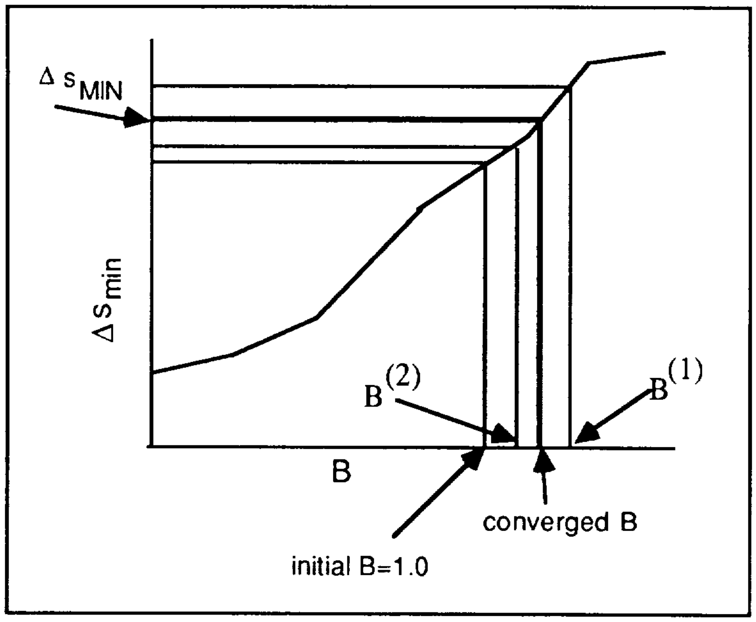


Figure 7.– Iterative calculation of B.

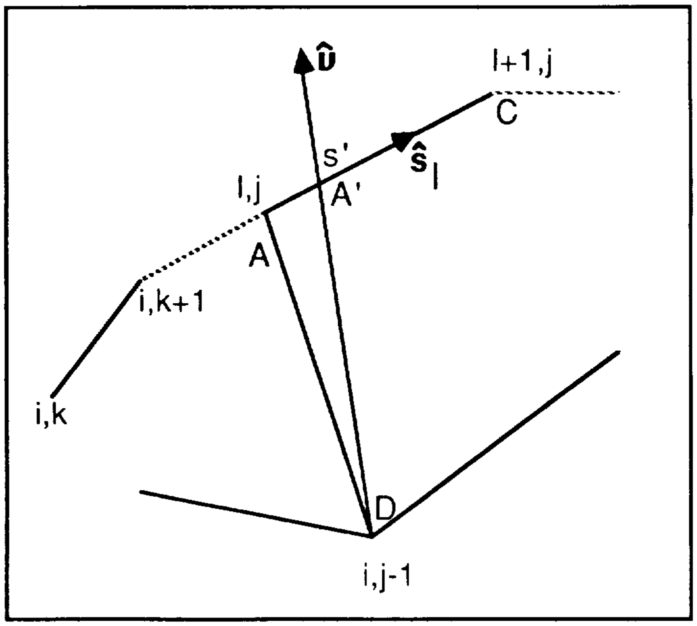
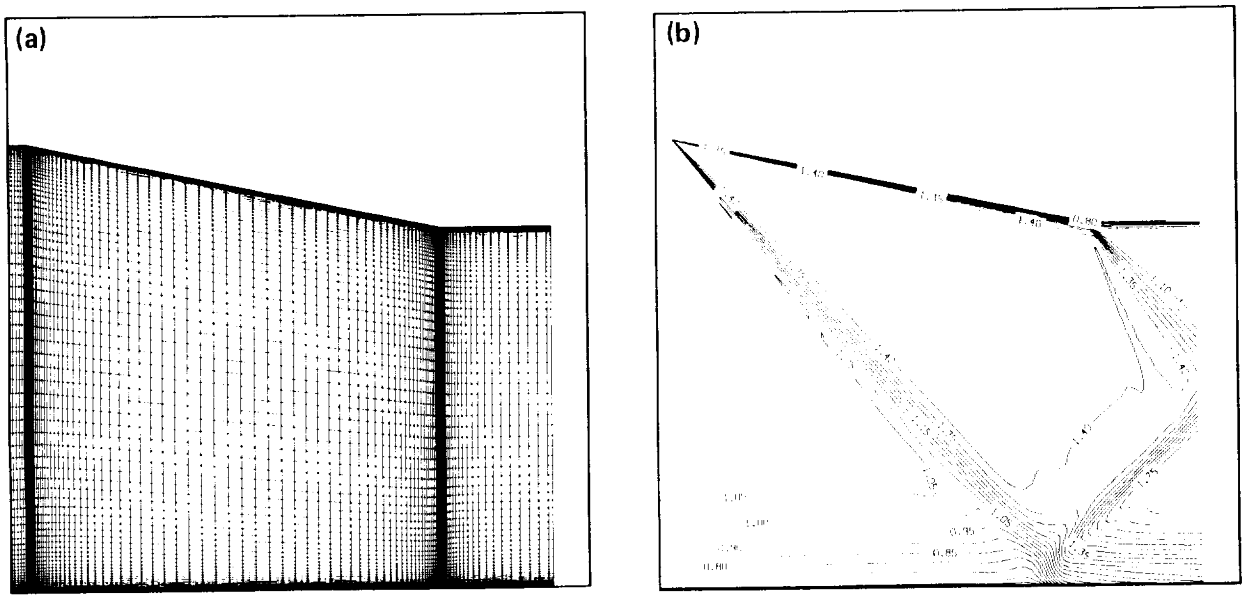


Figure 8.– Intersection of general vector $\hat{v}$ with streamwise vector.

ORIGINAL PAGE IS
OF POOR QUALITY



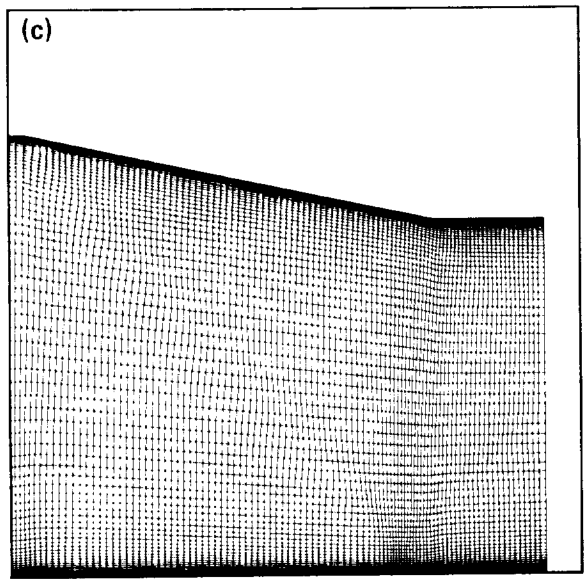


Figure 9.– Flow in a supersonic inlet. (a) Initial grid; (b) density contours; (c) adapted grid using all default parameters.

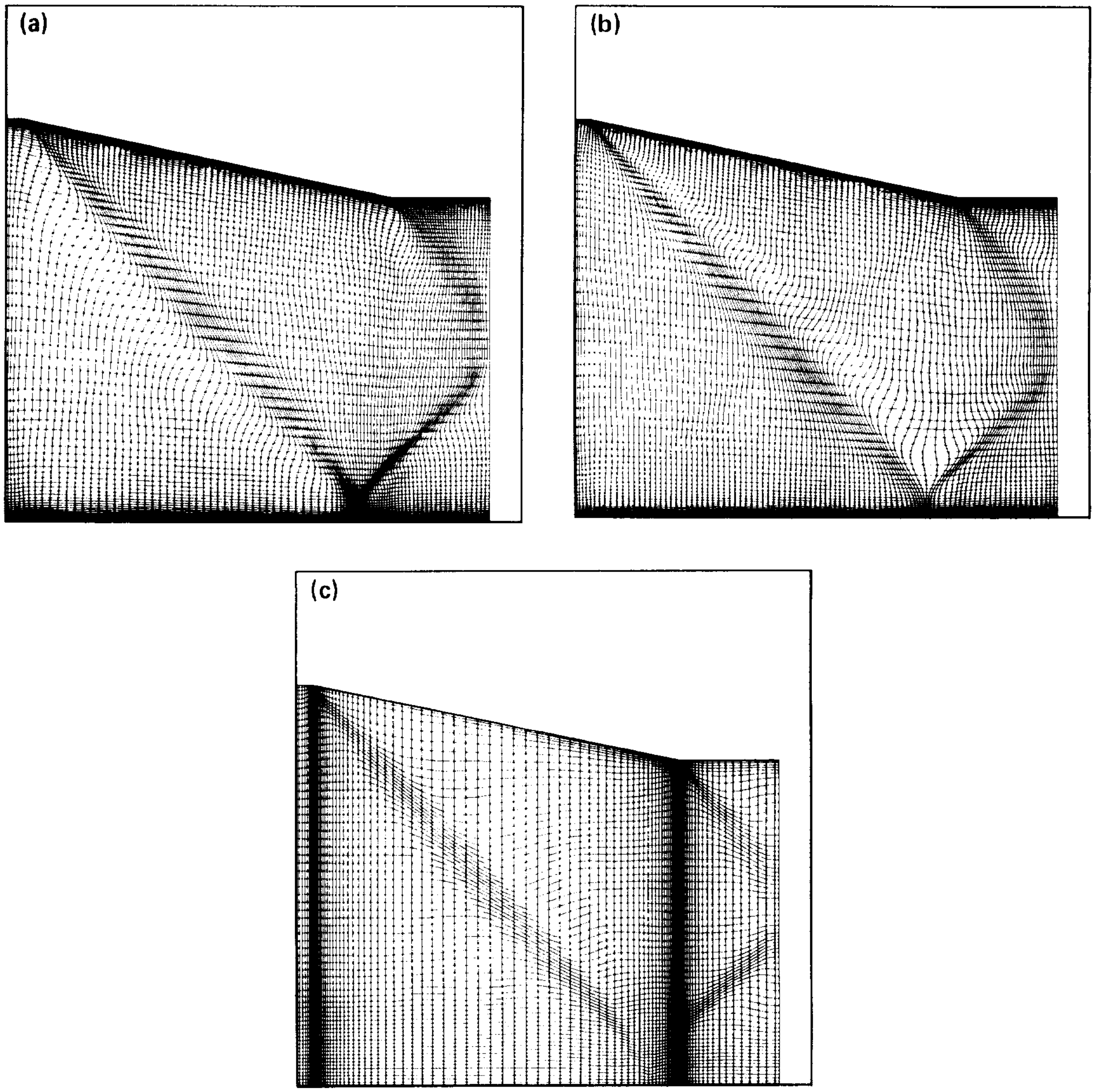


Figure 10.– Effect of adaption direction. (a) Adaption from bottom to top; (b) adaption from top to bottom, $jst = 79$, $jend = 1$; (c) marching in i, from right to left, $jstep = false$,$ist = 101$,$iend = 1$.

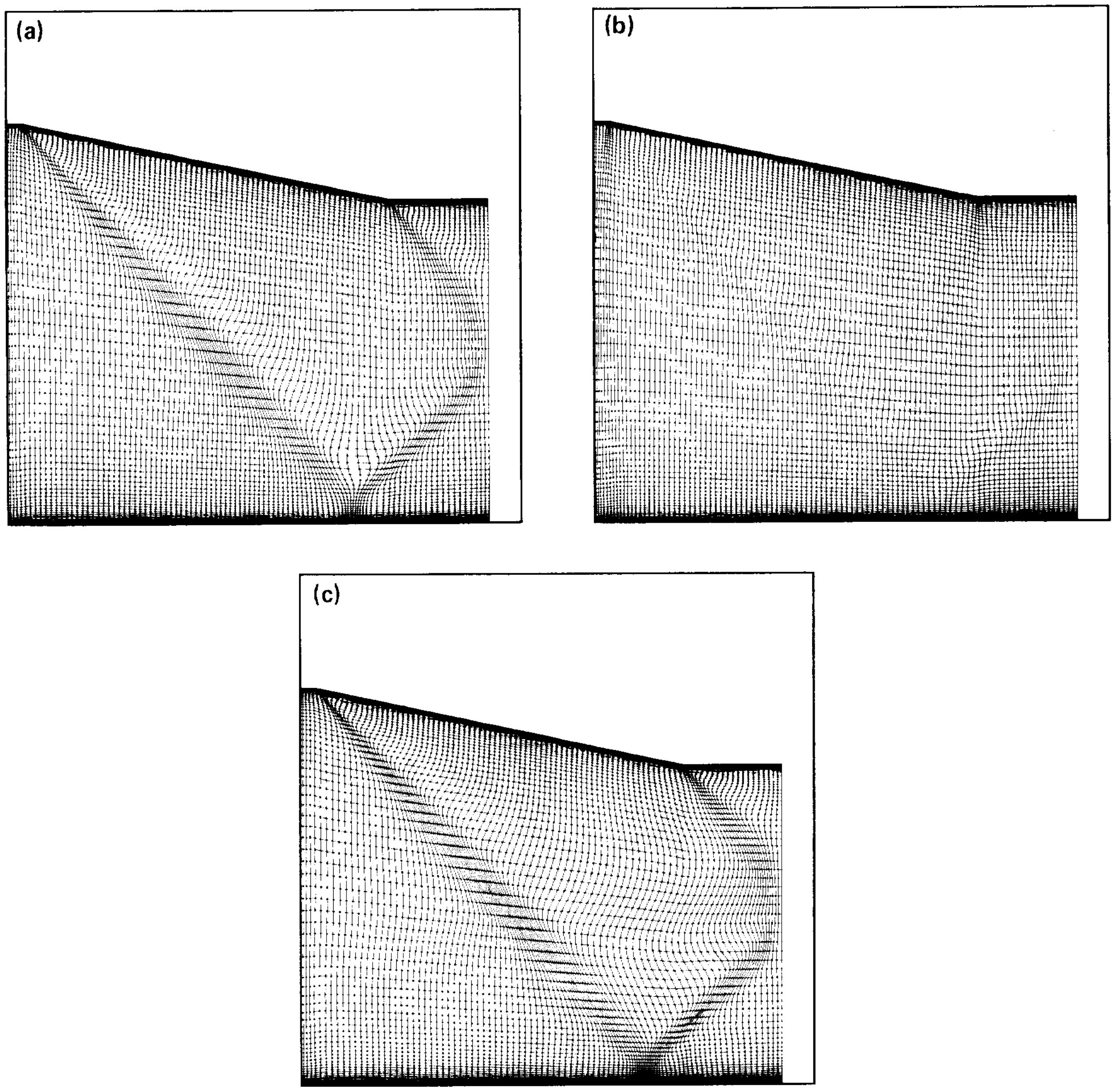


Figure 11.– Effect of control parameters. (a) "Baseline" adapted grid, λ = .001, jst = 79, $jend$ = 1; (b) λ = .01; (c) λ = .0001.

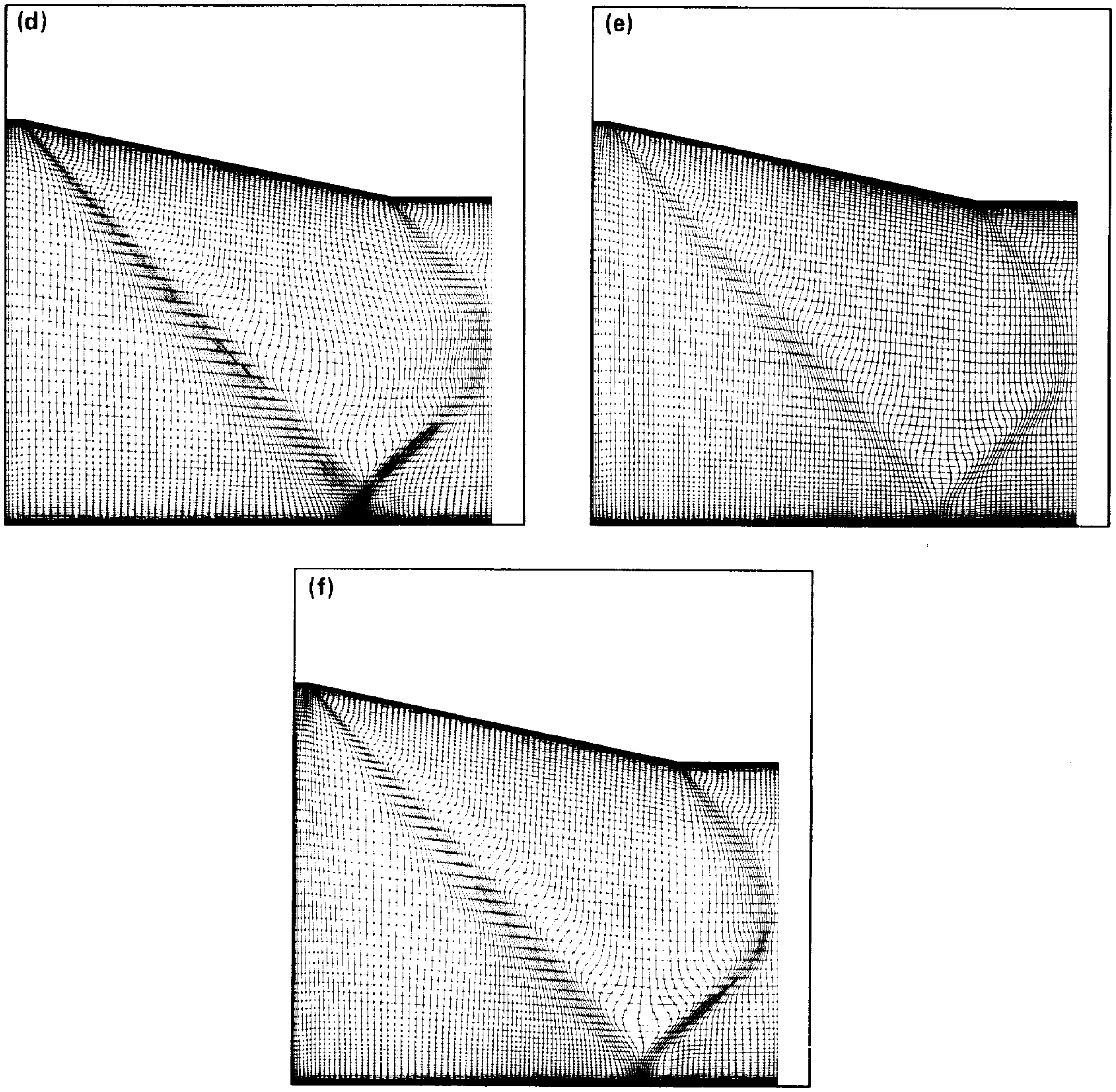
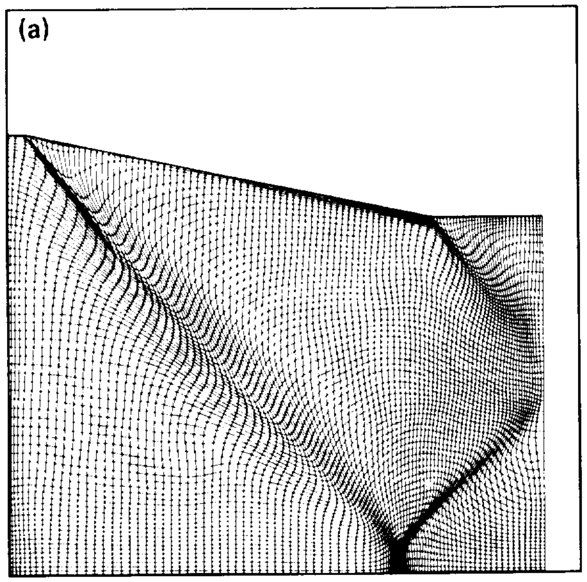


Figure 11.– Concluded. (d) C_t = 1.0; (e) C_t = .0; (f) Δs_{min} = .25, Δs_{max} = 4.0.



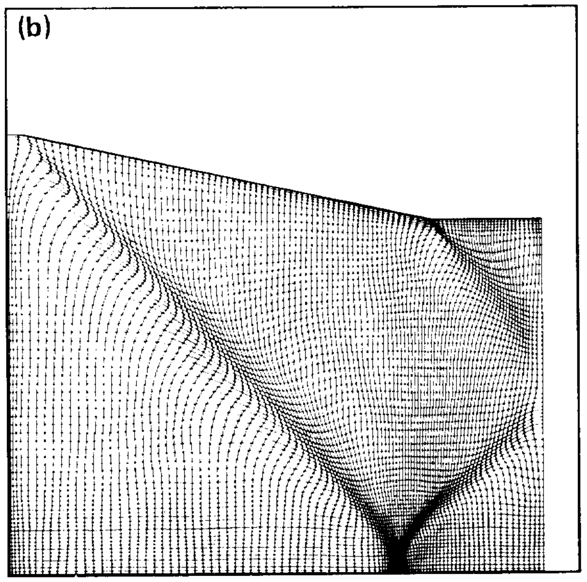
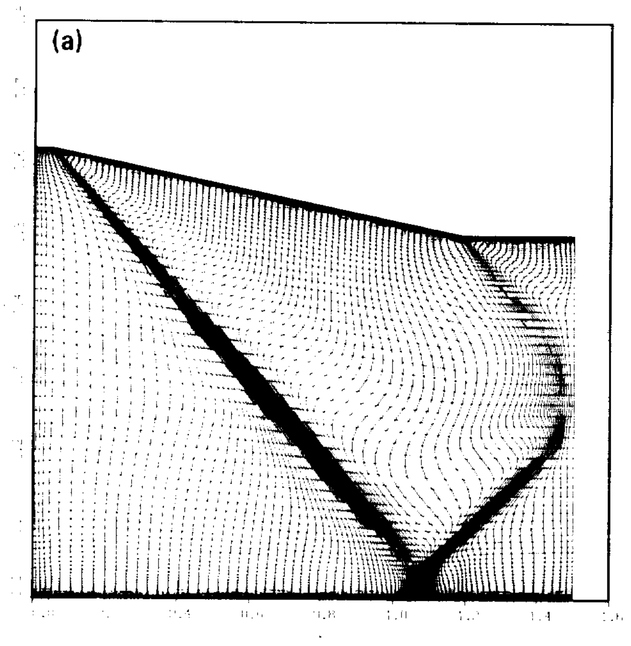


Figure 12.– 2-D adaption. (a) Marching in j followed by marching in i; (b) marching in i followed by j.



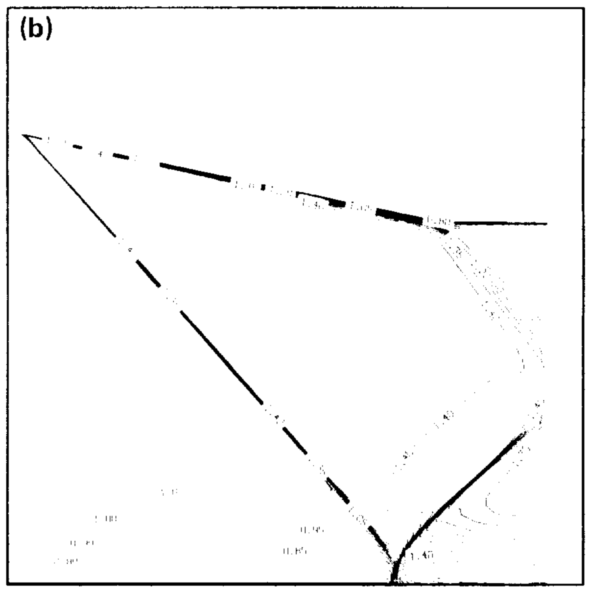


Figure 13.– Final solution using adapted grid as input. (a) Best adaption, $\Delta s_{min} = .25$, $\Delta s_{max} = 4.0$, $\lambda = .0001$; (b) solution density contours, using "best" adapted grid.

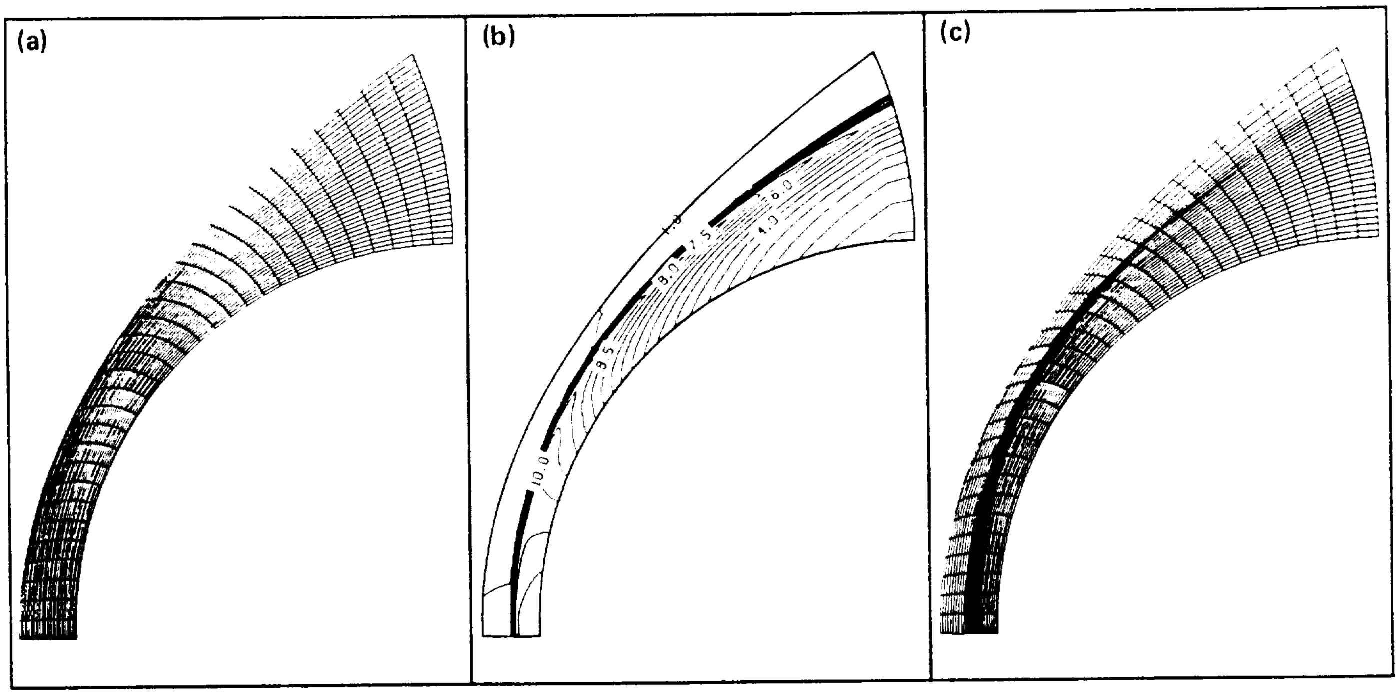


Figure 14.– Hypersonic blunt body. (a) Initial grid; (b) initial flow solution; (c) adapted grid, λ = .001.

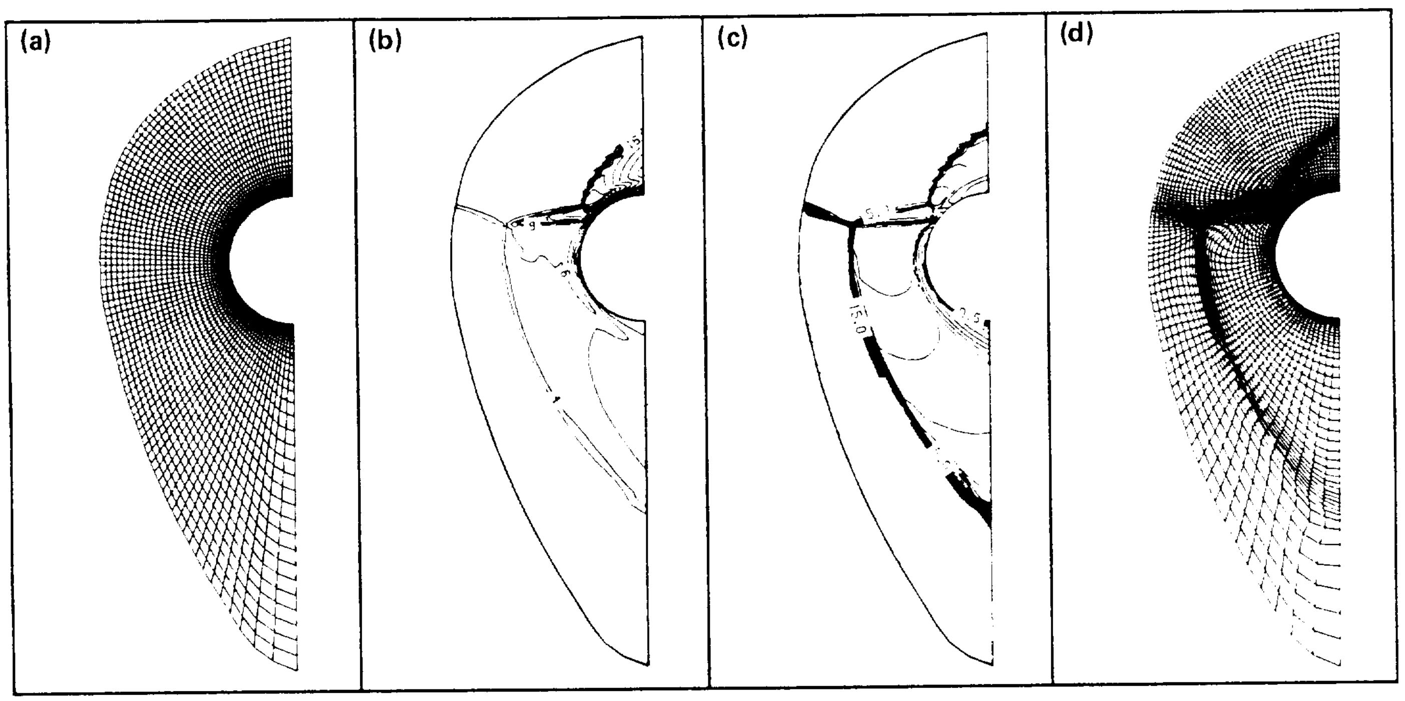


Figure 15.– Blunt-body shock impingement problem. (a) Initial grid; (b) initial density contours; (c) initial Mach contours; (d) adapted grid.

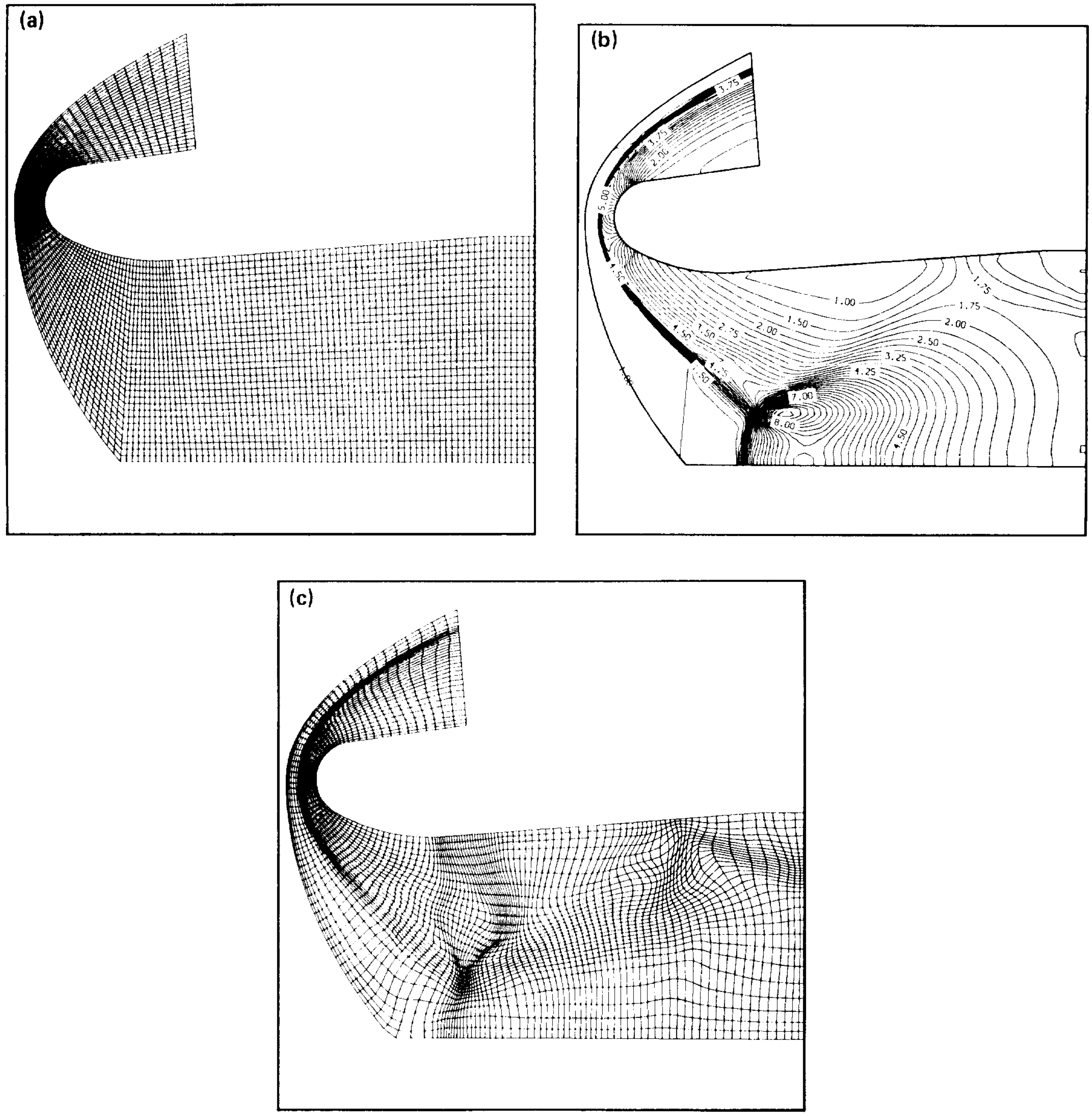
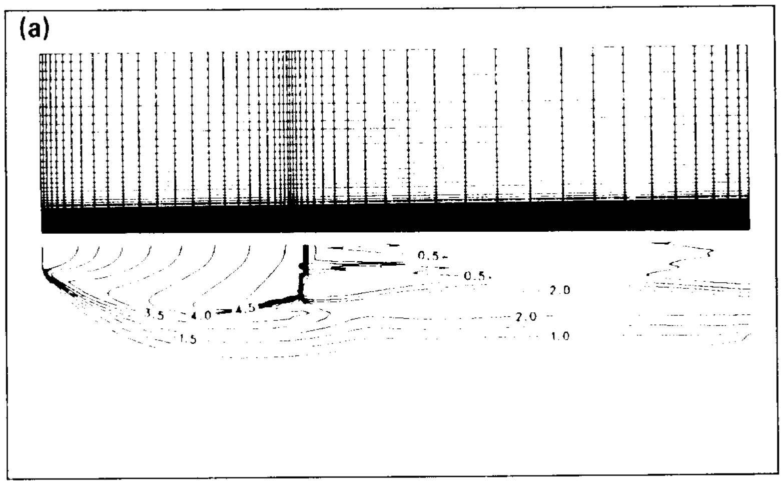
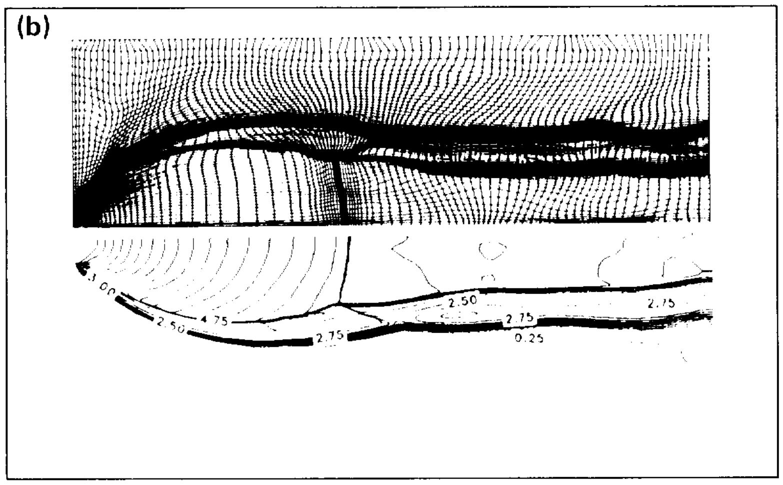


Figure 16.– Hypersonic inlet, zonal adaption. (a) Initial grid; (b) initial density flow contours; (c) adapted grid.

ORIGINAL PAGE IS
OF POOR QUALITY





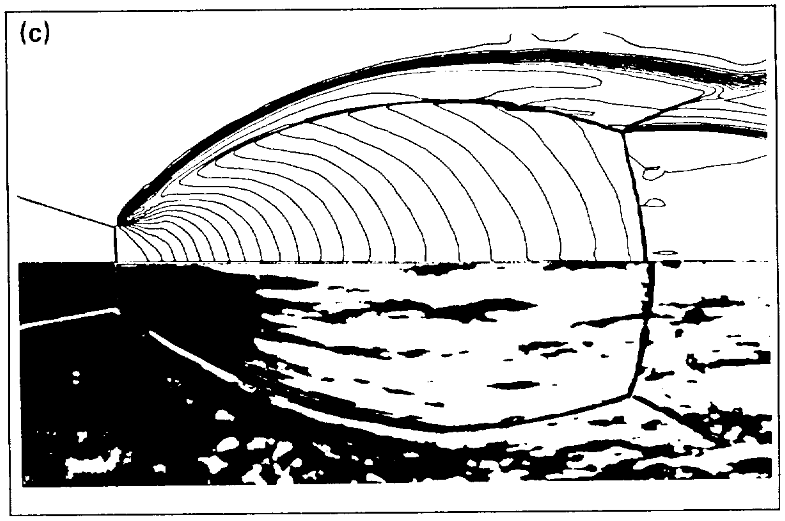


Figure 17.– Axisymmetric plume flow. (a) Initial grid and solution showing undeveloped flow features; (b) final adapted grid and flow solution obtained after several iterations; (c) comparison of computed solution with experimental shadowgraph results.

NASA
National Aeronautics and Space Administration

Report Documentation Page

1. Report No. NASA TM-102198	2. Government Accession No.	3. Recipient's Catalog No.
4. Title and Subtitle A Simplified Self-Adaptive Grid Method, SAGE		5. Report Date October 1989
		6. Performing Organization Code
7. Author(s) C. Davies* and E. Venkatapathy†		8. Performing Organization Report No. A-89156
		10. Work Unit No. 506-40-11
9. Performing Organization Name and Address Ames Research Center Moffett Field, CA 94035		11. Contract or Grant No.
		13. Type of Report and Period Covered Technical Memorandum
12. Sponsoring Agency Name and Address National Aeronautics and Space Administration Washington, DC 20546-0001		14. Sponsoring Agency Code

15. Supplementary Notes

Point of Contact: G. S. Deiwert, Ames Research Center, MS 230-2, Moffett Field, CA 94035
(415) 694-6198 or FTS 464-6198

*Sterling Software, Palo Alto, California

†Eloret Institute, Sunnyvale, California

16. Abstract

The formulation of the Self-Adaptive Grid Evolution (SAGE) code, based on the work of Nakahashi and Deiwert, is described in the first section of this document. The second section is presented in the form of a user guide which explains the input and execution of the code, and provides many examples. Application of the SAGE code, by Ames Research Center and by others, in the solution of various flow problems has been an indication of the code's general utility and success. Although the basic formulation follows the method of Nakahashi and Deiwert, many modifications have been made to facilitate the use of the self-adaptive grid method for single, zonal, and multiple grids. Modifications to the methodology and the simplified input options make this current version a flexible and user-friendly code.

17. Key Words (Suggested by Author(s)) Computational fluid dynamics Grid adaption		18. Distribution Statement Unclassified-Unlimited Subject Category - 61	
19. Security Classif. (of this report) Unclassified	20. Security Classif. (of this page) Unclassified	21. No. of Pages 49	22. Price A03

NASA FORM 1626 OCT 86

For sale by the National Technical Information Service, Springfield, Virginia 22161