MULTIGRID APPLICATIONS TO THREE-DIMENSIONAL ELLIPTIC COORDINATE GENERATION

Frank C. Thames

NASA Langley Research Center
Hampton, Virginia

Presented at the International Multigrid Conference

Copper Mountain, Colorado
April 6-8, 1983

MULTIGRID APPLICATIONS TO THREE-DIMENSIONAL ELLIPTIC COORDINATE GENERATION

Frank C. Thames

NASA Langley Research Center
Hampton, Virginia

ABSTRACT

The multigrid algorithm was applied to solve the coupled set of elliptic quasilinear partial-differential equations associated with three-dimensional coordinate generation. The results indicate that the multigrid scheme is more than twice as fast as conventional relaxation schemes on moderate sized grids. Convergence rates of order 0.90 were achieved on 36,000-point grids. The paper covers the form of transformation, develops the set of generation equations, and gives details on the multigrid approach used. Included are a development of the full-approximation storage scheme, details of the smoothing rate analysis, and a section devoted to rational programing techniques applicable to multigrid.

1.0 INTRODUCTION

Although the multigrid has been widely used to obtain efficient solutions of two-dimensional problems (e.g., ref. 1), its use on three-dimensional problems has been fairly limited. Indeed, most of the three-dimensional applications have been focused on solving the scalar potential equation for transonic flows about aircraft components. McCarthy and Reyhner (ref. 2) applied the multigrid algorithm to solve the potential flow about engine inlets. Brown (ref. 3) improved these results by adding a mesh embedding feature. Multigrid was used to accelerate existing wing-fuselage flow solvers by Shmilovich and Caughey (ref. 4). All of these results were generated using the standard tridiagonal-line SOR (LSOR) method as the smoothing algorithm. Recently, Caughey extended the approach of reference 4 by increasing the bandwidth of the LSOR smoother to allow for pentadiagonal inversions along each line (ref. 5). He was able to obtain an effective spectral radius of 0.93 which is very good. Finally, Raj added multigrid to the FLO22 code to obtain both wing analysis and design solutions (ref. 6).

The current work concerns the application of the multigrid method to solve the coupled set of three quasilinear elliptic partial-differential equations used to generate three-dimensional curvilinear coordinate systems. The multigrid procedure replaced a conventional LSOR solver in an existing code used to generate coordinates about wing/wing-tip configurations (ref. 7). A triple alternating-line SOR smoother is used to drive the multigrid solution.

The remainder of this paper is divided into four sections. In the next section we briefly describe the form of the transformation and introduce the governing equations. This is followed by a section devoted to the solution of grid equations which focuses on a description of the multigrid algorithm, smoothing rate analyses, and multigrid programing techniques. Computed results and conclusions are presented in the last two sections.

2.0 COORDINATE TRANSFORMATIONS FROM ELLIPTIC EQUATIONS

2.1 Form of the Transformation

There are many ways to transform wing/wing-tip geometries. The form chosen here is illustrated in figure 1. In this form the wing is "sliced" along the leading- and trailing-edge lines and then "unfolded" so that the wing and wing-tip surfaces are mapped onto a portion of the $\xi_3 = (\xi_3)_{min}$ plane in the computational domain. A more detailed description of this transformation is given in figure 2. Parts (a) and (d) of this figure indicate that the transformation has the form of a "sideways" C-grid in the physical domain. Note also that the transformation has an axis singularity similar to conventional cylindrical coordinates since the lines $a_3 - c_3$ and $e_3 - g_3$ in the physical domain (fig. 2(a)) map onto the cross-hatched planes fore and aft of the wing tip, respectively, in the $\xi_3 = (\xi_3)_{min}$ computational plane (fig. 2(b)). Finally, note that the planes on either side of the singular planes are periodic with each other. That is, planes $\Lambda_{3,1.1}$ and $\Lambda_{3,1.3}$ are the same physical surface as are planes $\Lambda_{3,1.7}$ and $\Lambda_{3,1.9}$ (fig. 2(c)). The computational domain, D^*, is required to be regularly-spaced. This requirement allows us to assume a uniform spatial step size (usually unity on the fine grid) for the finite-difference approximations made in the computational domain.

2.2 Governing Equations and Boundary Conditions

In reference 8, Mastin and Thompson set forth the mathematical foundation for the extension of the two-dimensional elliptic solver transformation techniques (ref. 9) to three dimensions. Their results showed that if the curvilinear coordinates (ξ_1, ξ_2, ξ_3) are required to satisfy

$$\nabla^2 \xi_i = b_i, \qquad X \epsilon D, \qquad i = 1,2,3 \tag{1a}$$

with boundary conditions

$$\xi_i = p_i(x_1, x_2, x_3), \qquad X \epsilon \partial D, \qquad i = 1,2,3 \tag{1b}$$

then the Cartesian coordinates, (x_1,x_2,x_3), must satisfy the coupled quasi-linear set

$$\sum_{j=1}^{3} \sum_{k=1}^{3} \alpha_{jk} \frac{\partial^2 x_i}{\partial \xi_j \partial \xi_k} + \sum_{k=1}^{3} \alpha_{kk} \phi_k \frac{\partial x_i}{\partial \xi_k} = 0, \quad \Xi \in D^*, \quad i = 1,2,3 \tag{2a}$$

with boundary conditions

$$x_i = q_i \, (\xi_1,\xi_2,\xi_3), \qquad \Xi \in \partial D^*, \qquad i = 1,2,3 \tag{2b}$$

where:

$$\alpha_{jk} \equiv \sum_{m=1}^{3} \beta_{mj} \beta_{mk}, \qquad j,k = 1,2,3 \tag{2c}$$

β_{jk} is the cofactor of the (j,k)-element of the Jacobian matrix

$$M = \frac{\partial(x_1,x_2,x_3)}{\partial(\xi_1,\xi_2,\xi_3)} \tag{2d}$$

and J is the determinant of M. We also have made the substitution

$$b_k \equiv \alpha_{kk} \phi_k / J^2, \qquad k = 1,2,3 \tag{3}$$

in developing equations (2a). The reasoning behind this substitution and the details involved in computing the ϕ_k coordinate control functions are covered in reference 7 and will not be repeated here. Equations (1b) and (2b) tend to imply that all boundary conditions are of the Dirichlet type. Actually, mixed Dirichlet and Neumann conditions can be used. However, both types of conditions cannot be given everywhere on ∂D or ∂D^* as this would overspecify the differential equations. In the current work, only Dirichlet conditions are used.

3.0 SOLUTION OF THE TRANSFORMATION EQUATIONS

Our problem is to solve the coupled, quasilinear set of three partial-differential equations given by equations (2a) subject to the boundary conditions specified by equations (2b). This is not an easy task. The α_{jk} coefficients present in these equations correspond to squares of the rates of change of arc length in the various computational coordinate directions. If the physical plane coordinates are highly stretched, the magnitudes of the α_{jk} terms can range over a wide spectrum; and equations (2a) take on a singular perturbation character. Furthermore, the presence of second cross derivatives (elliptic solvers rarely produce orthogonal coordinates) and first partial derivatives (provided $\phi_k \neq 0$) tends to destabilize most algorithms, as the central-difference approximations to these terms are not diagonally dominant. Thus, to solve these equations efficiently, a powerful algorithm is required. Since the set given by equation (2a) is elliptic, the multigrid algorithm is the obvious choice.

3.1 Basic Multigrid and the Full-Approximation Storage (FAS) Scheme

The basic idea of the multigrid algorithm is to solve a fine-grid problem by computing corrections to the fine-grid solution on coarser grids. To see this, consider the problem

$$L^h U^h = f^h \tag{4}$$

where L is a linear operator and h is the grid spacing. If u^h is an approximation to U^h, equation (4) can be written as

$$L^h(u^h + V^h) = f^h \tag{5a}$$

or

$$L^h u^h + L^h V^h = f^h \tag{5b}$$

where $V^h = U^h - u^h$ is the error. Now, if V^h is smooth, we can interpolate it to a coarser grid and solve the coarse grid analog of equation (5b); namely,

$$L^{2h} (I_h^{2h} V^h) = I_h^{2h} (f^h - L^h u^h)$$

or

$$L^{2h} V^{2h} = f^{2h} \tag{6}$$

where I_h^{2h} is an interpolation operator from the grid with spacing h to the coarser grid with spacing $2h$. Equation (6) can be solved (the solution is cheaper on the coarser grid) and V^{2h} can be used to correct the fine-grid solution using

$$(u^h)_{new} = (u^h)_{old} + I_{2h}^h V^{2h} \tag{7}$$

Obviously, more than one coarse grid can be used. The key feature in the argument given above (other than the linearity of L) is that V^h must be smooth--that is, V^h must have no high-frequency content. Otherwise, the interpolation produces spurious results and the whole process falls apart since the smooth, low-frequency errors are not accurately represented on the coarse grid. One way to insure that V^h is smooth is to apply an algorithm to equation (5a) which damps (smooths) high-frequency error components. This is the usual practice and this algorithm is normally called the "smoother."

To develop a multigrid algorithm suitable for nonlinear operators, we return to equation (5a) and subtract $L^h u^h$ from both sides. This yields

$$L^h(u^h + V^h) - L^h u^h = f^h - L^h u^h \tag{8}$$

The coarse-grid equation which approximates equation (8) is

$$L^{2h}(I_h^{2h}u^h + V^{2h}) - L^{2h}(I^{2h}u^h) = I_h^{2h}(f^h - L^h u^h)$$

or

$$L^{2h}u^{2h} = f^{2h} \tag{9}$$

where

$$f^{2h} \equiv I_h^{2h}(f^h - L^h u^h) + L^{2h}(I_h^{2h}u^h) \tag{10}$$

The corresponding fine-grid correction step is

$$(u^h)_{new} = (u^h)_{old} + I_{2h}^{h}\left[u^{2h} - I_h^{2h}(u^h)_{old}\right] \tag{11}$$

Note that the term in brackets is indeed a correction. Again, the extension to multiple coarse grids is obvious. Equations (9)-(11) define the FAS multigrid scheme which is applicable to both linear and nonlinear operators. One of the nice features of the FAS approach is that the problem solved on all grids has the same form. This feature has significant programing implications and will be discussed later.

3.2 Multigrid Smoothing Factor

We now return to the problem of smoothing the error in the solution on a given grid. As mentioned earlier, one way to smooth is to apply an algorithm which damps high-frequency wave numbers. To quantify this notion, define the smoothing factor (ref. 10) as

$$\mu \equiv \max \left\{g(\theta_1,\theta_2,\theta_3)\right\}, \ \frac{\pi}{2} \leqslant \left| \theta_1,\theta_2,\theta_3 \right| \leqslant \pi \tag{12}$$

where g is the von Neumann damping ratio for the smoothing algorithm and θ_i is the phase angle associated with the ξ_i-coordinate $(i = 1,2,3)$. Note that μ measures the performance of the smoother only at the high wave numbers. Acceptable values of the smoothing factor lie in the range $0 < \mu \leqslant 0.6$. Such values have been obtained for two-dimensional applications (e.g., ref. 1), but are difficult to obtain in complex three-dimensional applications.

3.3 Smoothing Algorithms for the Coordinate Generation Equations

In the current work, alternating line SOR (LSOR) methods were used as smoothers. Line SOR methods work well on elliptic equations. Their principal drawback is that their damping rates (and, hence, the smoothing factors) deteriorate rapidly if a grid stretch occurs in the implicit direction. This deterioration can be alleviated in many cases by alternating LSOR sweeps in several directions. For example, if g_1 is the damping rate for the ξ_1-LSOR (i.e., the method is implicit in the ξ_1-direction) sweep and g_2 is the rate for the ξ_2-LSOR sweep, then the total damping rate for the ξ_1:ξ_2-LSOR alternating method is

$$g = g_1 g_2$$

Since $g_1, g_2 < 1.0$, the product, $g_1 g_2$ is less than either g_1 or g_2. Even if one of the factors is near unity (due, say, to a grid stretch in that direction), the total damping rate will be acceptable as long as the other factor is small. Grids with severe stretching in only one direction are common in many applications. The alternating implicit sweeps also promote boundary data communication, so both sweeps should be kept even though one of them may have a poor damping rate.

For the coordinate generation application, a triple alternating line SOR smoother is used. That is, the smoothing algorithm is $\xi_1 : \xi_2 : \xi_3$-LSOR. Thus, the total damping rate is

$$g = g_1 g_2 g_3 \tag{13}$$

Consider the application of this method to the linear, scalar model problem

$$\sum \sum a_{jk} u_{\xi_j \xi_k} = \sum c_k u_{\xi_k} = 0; \qquad j,k = 1,2,3 \tag{14}$$

The damping rate expression is quite complex, so we will analyze its behavior numerically. Figure 3 presents contours of the damping rate, g, for a low value of θ_3 and for typical values of the derivative coefficients obtained in practice. Figure 3(a) indicates that the triple-line algorithm does not damp very well ($g_{max} = 0.95$); in particular, it does a poor job on the lower frequencies. Note, however, in figure 3(b) the lower frequencies have been eliminated, so the contours shown represent the multigrid smoothing factor. (See eq. 12.) In this instance,

$g_{max} = 0.50$ or, by definition, $\mu = 0.50$. Thus, although the alternating-line method is not a particularly good approach for solving equation (14), it is an excellent smoother for the multigrid algorithm.

To finish the smoothing analysis, values for the free parameters inherent in the $\xi_1 : \xi_2 : \xi_3$-LSOR algorithm--namely, the various LSOR acceleration parameters--need to be established. Figure 4 presents plots of μ versus ω_1 for various values of ω_2 and for $\omega_3 = 1$. ($\theta_3 = 0$ is the worse case.) The derivative coefficient values are the same as used to develop figure 3. The data given in figure 4 indicate that the optimum μ is obtained for the set

$$\{\omega_1, \omega_2, \omega_3\} = \{1.8,\ 1.0,\ 1.0\}.$$

The optimum value of $\omega_3 = 1$ was determined from other results similar to those given in figure 4. The set of ω_i's obtained above apply to the finest grid. The optimum value of ω_1 decreases toward unity on the coarser grids.

3.4 FAS Multigrid Programing Techniques

There is a general belief that multigrid is difficult to code. This is not true. The keys to successful multigrid programing are: (1) organized data bases and (2) the development of grid-independent code. Each of these key elements can be accomplished by utilizing the modularization and dynamic addressing capabilities available in FORTRAN.

Modularization should be used in all programs. Basically, it implies that code which performs different functions should be placed in different routines. The suggested modularization for multigrid programs is shown in figure 5. The routines enclosed by the dashed line must be made independent of grid size as they must compute on all grids.

Grid-size independent code can be realized by using subroutine argument lists rather than common to implement data base flow between routines. To understand why argument lists are useful, one must understand how they "transfer" data. Recalling that all subroutines are compiled independently, consider the following code:

```
SUBROUTINE INJ(NDF,NDC,IM,JM,UF,UC)
DIMENSION UF(NDF,1),UC(NDC,1)
                   .
                   .
                   .
DO 10 J=1, JM
DO 10 J=1, IM
                   .
                   .
                   .
RETURN
END
```

The compiler assumes that the calling routine will provide SUBROUTINE INJ with the <u>core address</u> of each argument in the list (not the values associated with each argument). Since UF and UC are two-dimensional arrays, the compiler assumes that INJ will be provided with the addresses of UF(1,1) and UC(1,1), respectively. Moreover, the DIMENSION statement tells the compiler to use the two-dimensional array address calculation algorithm to compute addresses for UF and UC. To do this, the compiler must know the "correct" first dimension (first two dimensions for three-dimensional arrays); hence, these dimensions are "sent-in" as variables--NDF and NDC. This subroutine is thus completely independent of the size and dimensions of the arrays UF and UC. This is exactly the situation we wished to create. The example given above was not chosen at random. The call list is identical with the author's two-dimensional fine-to-coarse grid injection routine. (See fig. 5.)

To take advantage of the grid-size independent code and link it to the data base organization to be discussed later, it is necessary to know what happens in the calling routine. Suppose we have the following

```
      DIMENSION U(34)
      DATA N1/5/,N2/3/,I1/S/,J1/5/
                          .
                          .
                          .
      CALL INJ(N1,N2,I1,J1,U(1),U(26))
                          .
                          .
                          .
```

The calling routine sends the INJ routine the core addresses of each of the arguments in the call. Thus, the call has set up the following storage-location equivalence between the calling routine and INJ: (N1,NDF), (N2,NDC), (I1,IM), (J1,JM), (U(1),UF(1,1)), and (U(26),UC(1,1)).

Note that all the quantities are actually stored in the calling routine. The subroutine argument list merely allows us to change both the name of a variable and the manner in which it is addressed.

The above example leads naturally to the discussion of data base management in multigrid codes. Consider the following code for a three-grid problem in which the grids are 9 by 9, 5 by 5, and 3 by 3:

```
      SUBROUTINE CONTROL
      COMMON/DA/U(115),N(3),IADR(3)
      DATA N/9,5,3/,IADR/1,82,107/
                 .
                 .
                 .
      NADR = IADR(NG)
      CALL RELAX(N(NG),U(NADR))
                 .
                 .
                 .
      SUBROUTINE RELAX(N,U)
      DIMENSION U(N,1)
                 .
                 .
                 .
```

In this example U is used to store the unknowns for all grids, N is used to store the first dimension of each grid, IADR stores the start addresses for the unknowns on each grid, and NG is the grid number. (NG=1 is the finest grid.) Note that RELAX never knows which of the three grids it is performing relaxation sweeps on. Finally, note that the programing techniques described above also make it fairly easy to retrofit multigrid to an existing code.

3.5 Multigrid Cycling Algorithms

Implicit in the multigrid method are the specification of grid switching or cycling algorithms. Brandt (ref. 10) prefers adaptive grid switching algorithms. These can be complex to code and are prone to traps (e.g., limit cycles in which the solution procedure "bounces" back and forth between two or more of the coarse grids). The development of complex grid cycling algorithms also tends to keep the code generator's mind off what he should be doing--namely, developing a good smoother. In the current work a fixed cycle is used. That is, a specified number of smoother sweeps is carried out on each grid during every multigrid cycle.

4.0 DISCUSSION OF RESULTS

A number of different cases were computed to establish the performance of the multigrid solution procedure. Before discussing these, we present a sample coordinate system generated by one of the solutions. Figure 6 shows a segment of a 33^3 grid illustrating portions of the wing upper surface boundary data ($\xi_3 = 1$) and three computed coordinate planes ($\xi_1 = 2, 17$ and $\xi_2 = 22$). The coordinates are smooth and well distributed, reflecting the influence of the ϕ_2 and ϕ_3 coordinate control functions.

To illustrate that the multigrid algorithm does dramatically reduce the computational effort required to solve equations (2), figure 7 presents a comparison of the iteration histories of three methods: (1) Multigrid with a $\xi_1{:}\xi_2{:}\xi_3$-LSOR smoother, (2) the $\xi_1{:}\xi_2{:}\xi_3$-LSOR algorithm without multigrid, and (3) the ξ_3-LSOR method without multigrid which was used in reference 7. Here, R is the residual (maximum of the three residuals resulting from equations (2a)), R_I is the initial residual, ρ_{eff} is the effective spectral radius (defined below) and λ is the average number of work units required to reduce the residual one order of magnitude. The effective spectral radius is defined as

$$\rho_{eff} \equiv (R_F/R_I)^{\frac{1}{w}}$$

Where R_F is the final residual, and w is the number of work units required to converge the solution. In this paper, a work unit is defined as one LSOR sweep on the finest grid. Figure 7 indicates that the multigrid algorithm is almost twice as fast as the $\xi_1:\xi_2:\xi_3$-LSOR method alone and nearly three times faster than the ξ_3-LSOR approach used in reference 7. Note that since the multigrid residual history line is curved, the asymptotic spectral radius is somewhat higher than ρ_{eff}. This somewhat undesirable behavior results from the smoothing algorithm's inability to smooth effectively on the finest grid.

Figure 8 documents the effect of initiating the multigrid algorithm on the coarsest rather than the finest grid. Note that, although the effective spectral radius is lowered somewhat by starting on the coarsest grid, the asymptotic convergence rate of both approaches is the same, as should be expected. Starting on the coarsest grid does not affect the smoothing factor which determines the overall convergence rate. It was mentioned in section 3.0 that the presence of centrally-differenced first partial derivatives has an adverse effect on the smoothing performance of most algorithms. Fortunately, this is not the case with the current application as evidenced by the convergence rate data given in figure 9 which documents the effects of the coordinate control functions, ϕ_i, i=1,2,3. (Note from eq. (2a) that the presence of non-zero ϕ_i's introduces first derivatives into the equation.)

The computer code which generated the results given in this paper is capable of implementing seven different smoothing algorithms--each of the uni-directional LSOR solvers and all possible combinations of the three. To implement this capability and to save storage, each of the LSOR algorithms computes its own residual. Thus, for example, the $\xi_1:\xi_2:\xi_3$-LSOR smoother uses three work units per iteration on the fine grid. Since only one residual calculation is really needed to implement any of the mutiple-line algorithms, the computational cost of the triple-line smoother could be at least halved with no significant loss in smoothing efficiency if only one residual per iteration were computed. Under these circumstances, ρ_{eff} would decrease to approximately 0.84 and λ to about 15.

5.0 CONCLUSIONS

The multigrid algorithm utilizing a triple alternating-line SOR smoother has been successfully applied to accelerate the convergence of the system of three-dimensional, elliptic, coordinate-system generation equations. Effective spectral radii of order 0.90 were obtained on 33^3 grids. The method is robust and, if programed properly, easy to apply to any set of equations. However, the results given in this paper do indicate the need for further development of more effective smoothing algorithms.

6.0 REFERENCES

1. Jameson, A.: Acceleration of Transonic Potential Flow Calculations on Arbitrary Meshes by the Multiple Grid Method. AIAA Paper 79-1458, July 1979.

2. McCarthy, D. R.; and Reyhner, T. A.: Multigrid Code for Three-Dimensional Transonic Potential Flow About Inlets. AIAA J., vol. 29, no. 1, Jan. 1982, pp. 45-50.

3. Brown, J. J.: A Multigrid Mesh-Embedding Tecnique for Three-Dimensional Transonic Potential Flow Analysis. NASA CP-2002, Oct. 1981, pp. 131-149.

4. Shmilovich, A.; and Caughey, D. A.: Application of the Multi-Grid Method to Calculations of Transonic Potential Flow About Wing-Fuselage Combinations. NASA CP-2002, Oct. 1981, pp. 101-130.

5. Caughey, D. A.: Multi-Grid Calculation of Three-Dimensional Transonic Potential Flows. AIAA Paper 83-0374, Jan. 1983.

6. Raj, P.: A Multigrid Method for Transonic Wing Analysis and Design. AIAA Paper 83-0262, Jan. 1983.

7. Thames, Frank C.: Generation of Three-Dimensional Boundary-Fitted Curvilinear Coordinate Systems for Wing/Wing-Tip Geometries Using the Elliptic Solver Method. Numerical Grid Generation, J. F. Thompson, Ed., North Holland, 1982, pp. 695-716.

8. Mastin, C. W.; and Thompson, J. F.: Transformation of Three-Dimensional Regions onto Rectangular Regions by Elliptic Systems. Numerische Mathematik, vol. 29, 1978, pp. 397-407.

9. Thompson, J. F.; Thames, F. C.; and Mastin, C. W.: Automatic Numerical Generation of Body-Fitted Curvilinear Coordinate System for Field Containing Any Number of Arbitrary Two-Dimensional Bodies. J. of Comp. Phys., vol. 15, no. 3, July 1974, pp. 299-319.

10. Brandt, A.: Multi-Level Adaptive Solutions to Boundary-Value Problems. Mathematics of Computation, vol. 31, no. 138, April 1977, pp. 333-390.

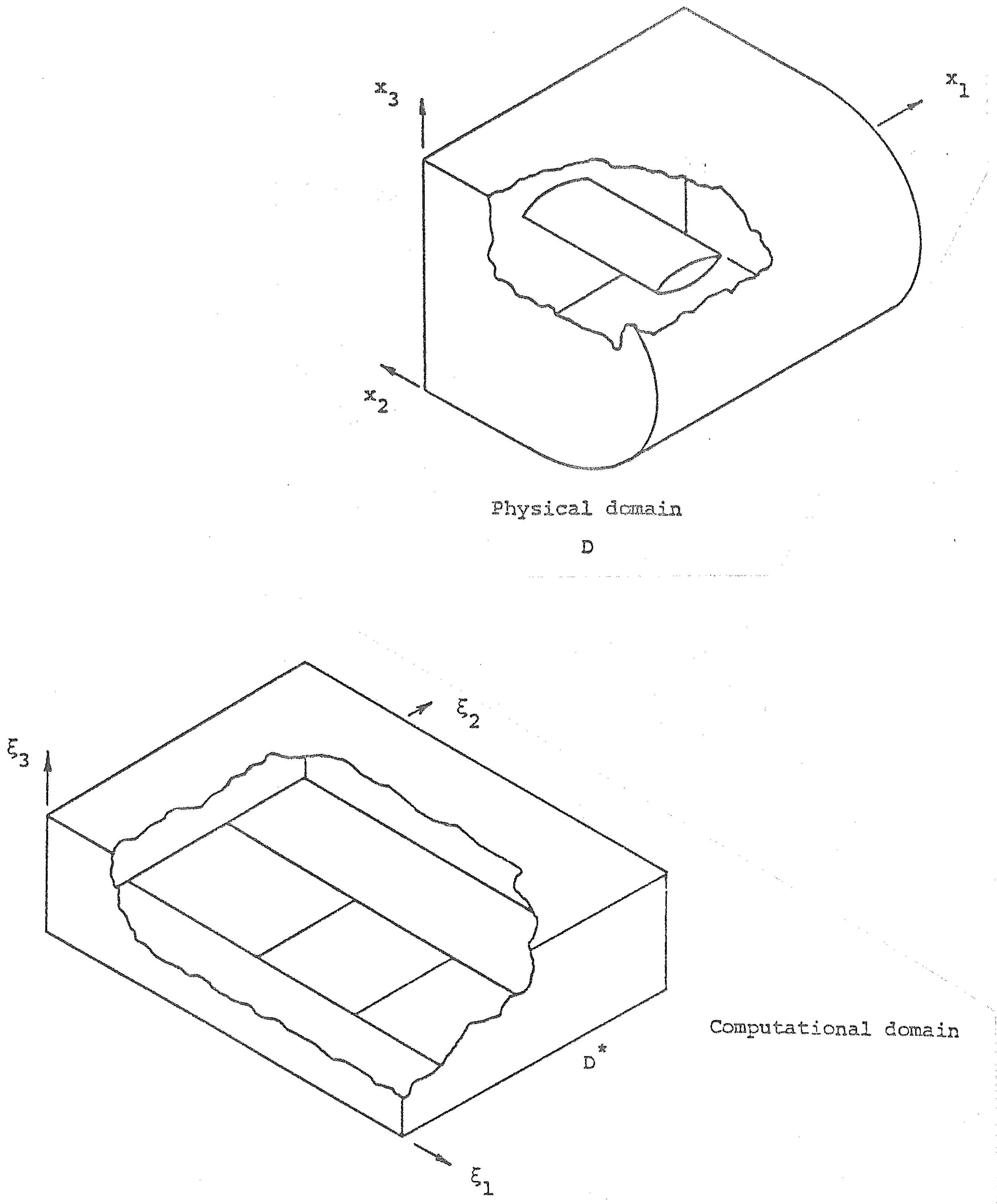
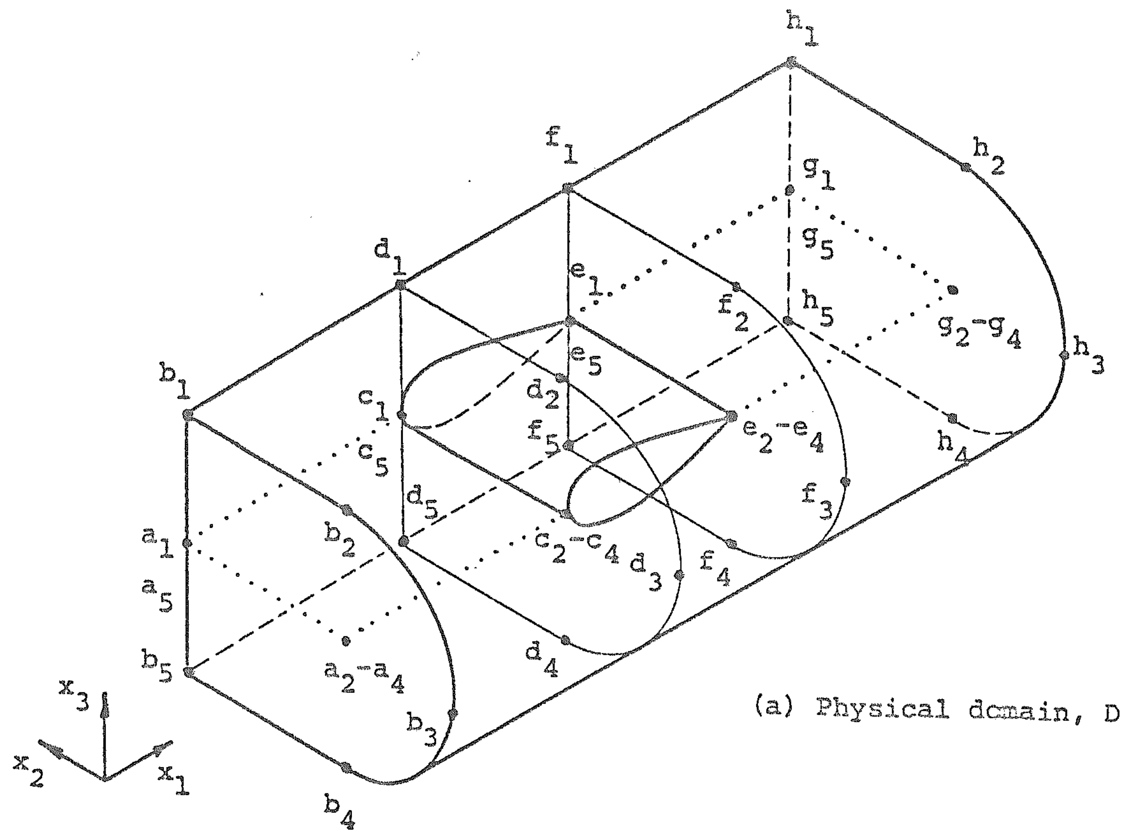
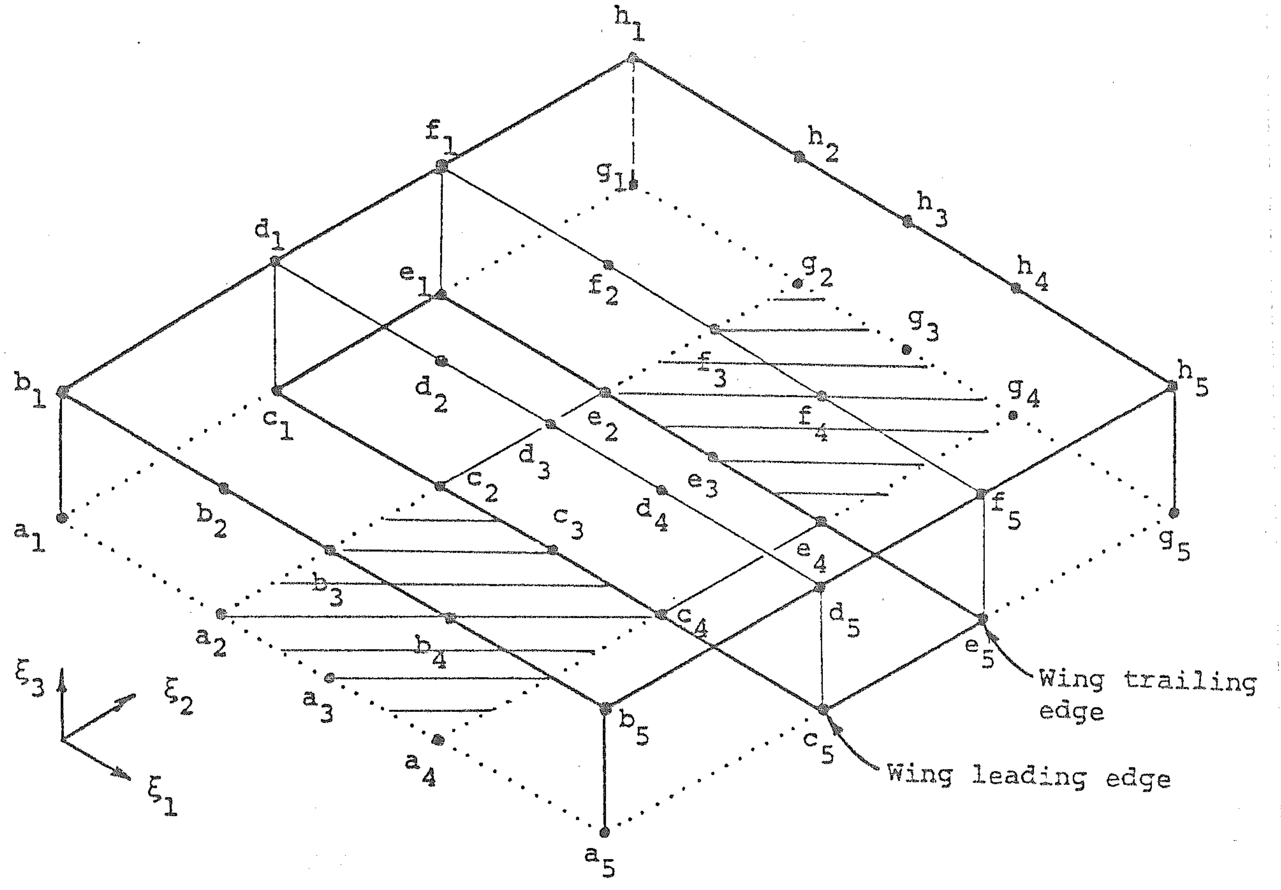


Figure 1.- Schematic of the transformation.

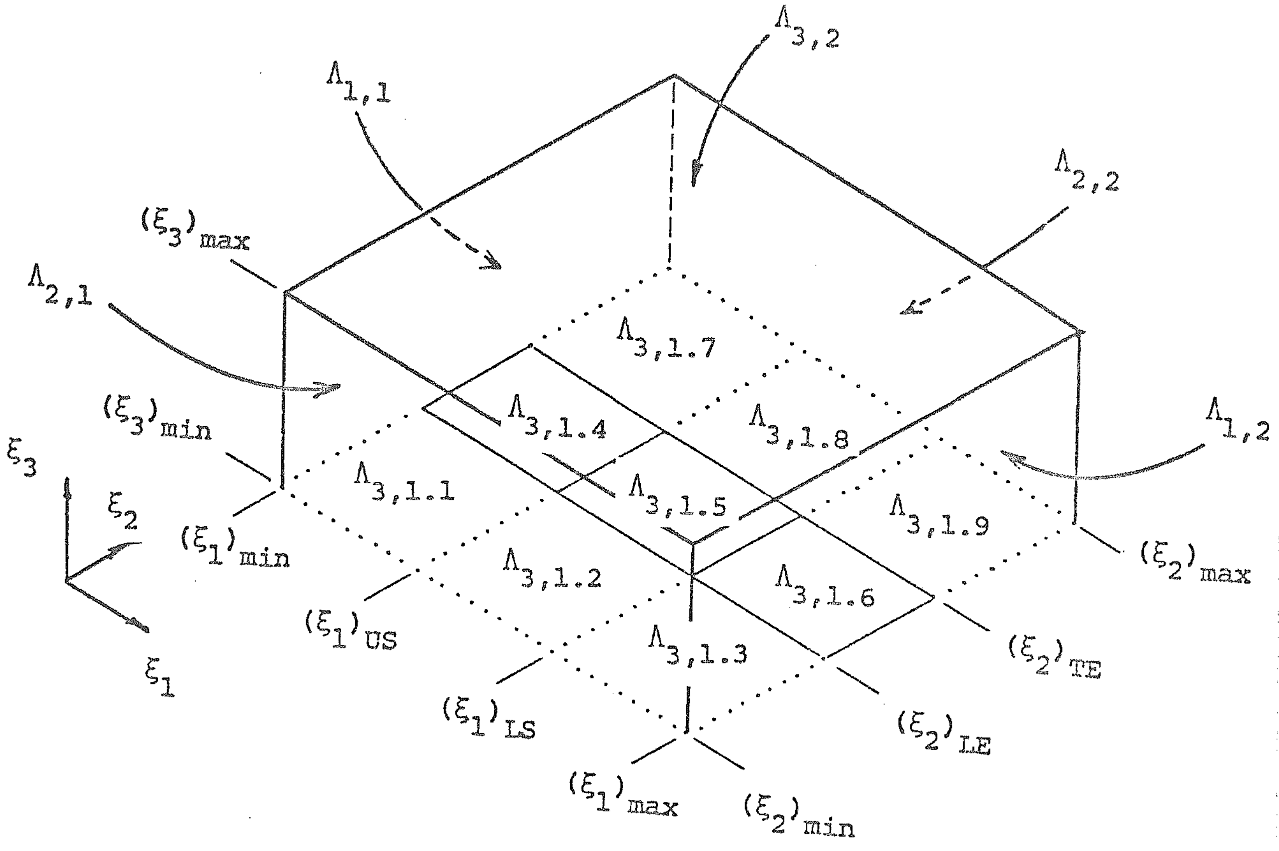


(a) Physical domain, D

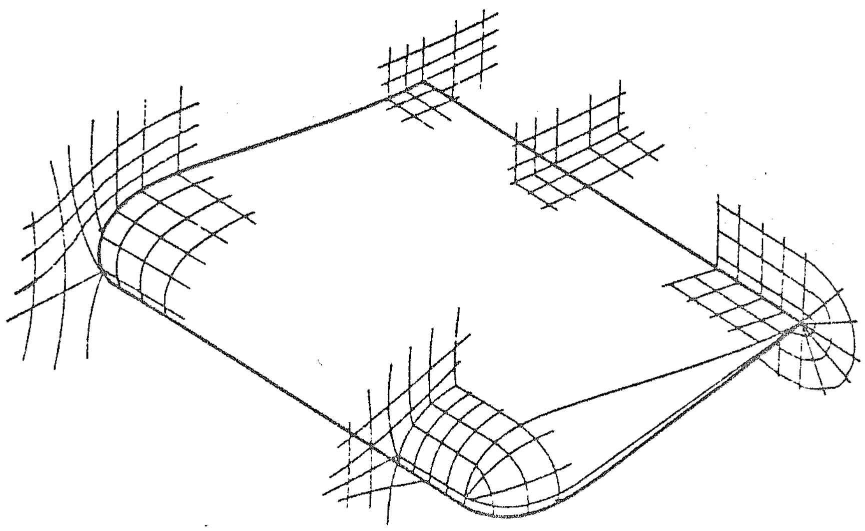


(b) Computational domain, D^*

Figure 2.- Details of the transformation.

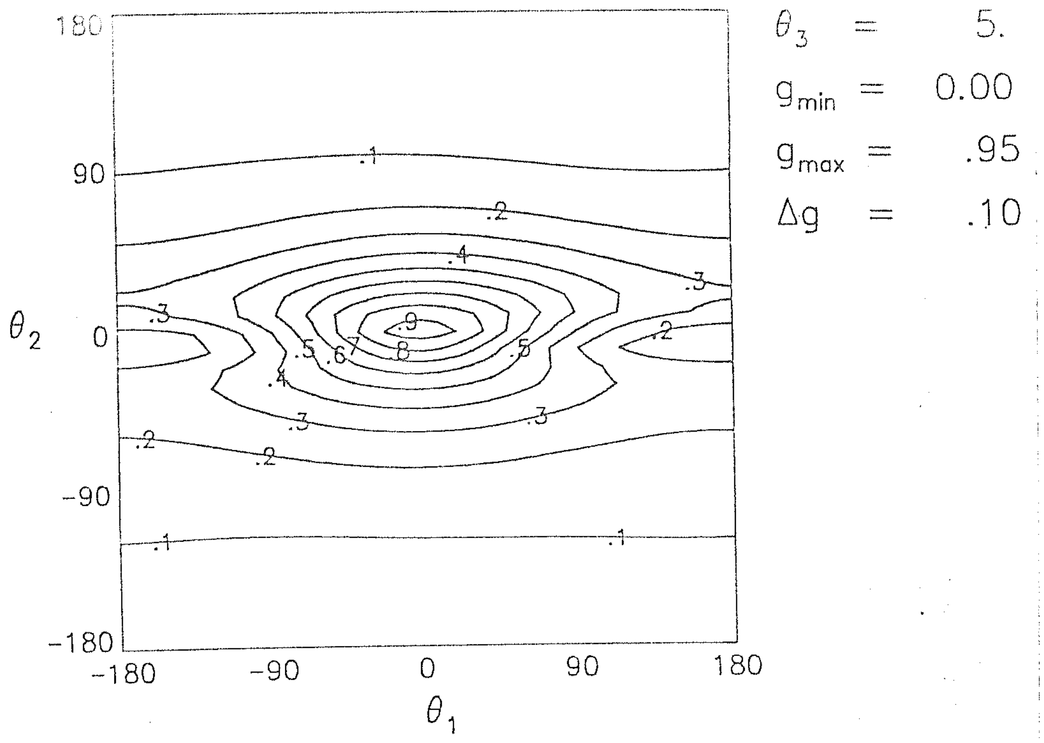


(c) Computational domain planes

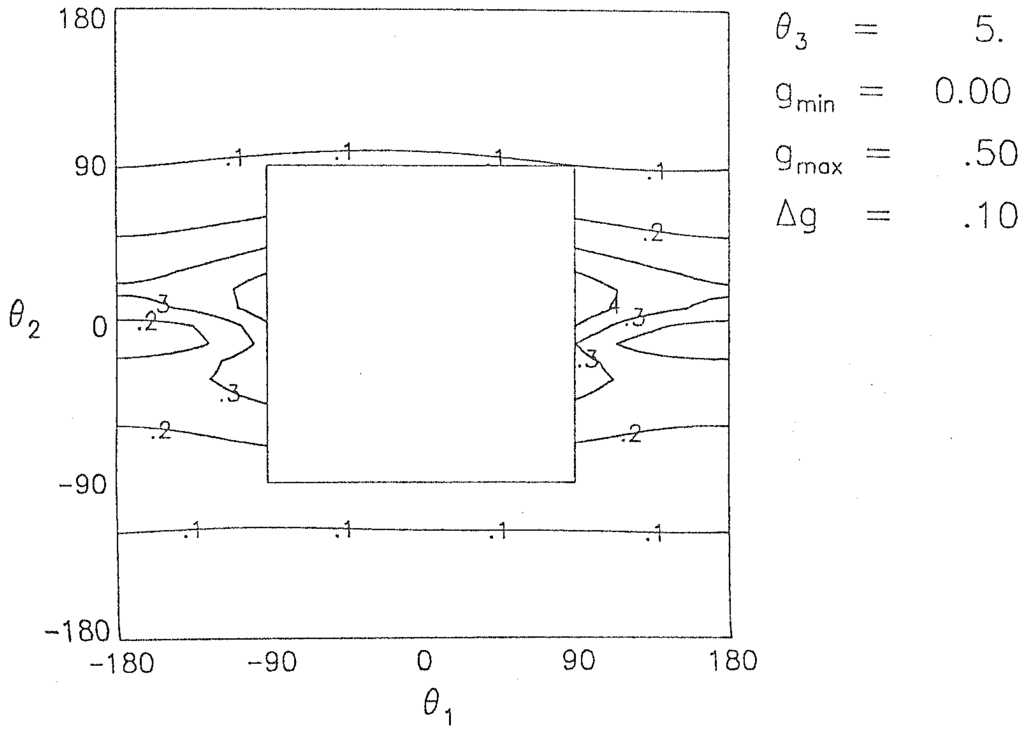


(d) Grid structure in the physical domain

Figure 2.- Concluded.



(a) von Neumann damping rate contours



(b) Multigrid smoothing factor contours

Figure 3.- von Neumann damping rate and multigrid smoothing factor contours for the $\xi_1:\xi_2:\xi_3$-LSOR algorithm appled to the model equation (ω_1=1.8, ω_2=1.0, ω_3=1.0).

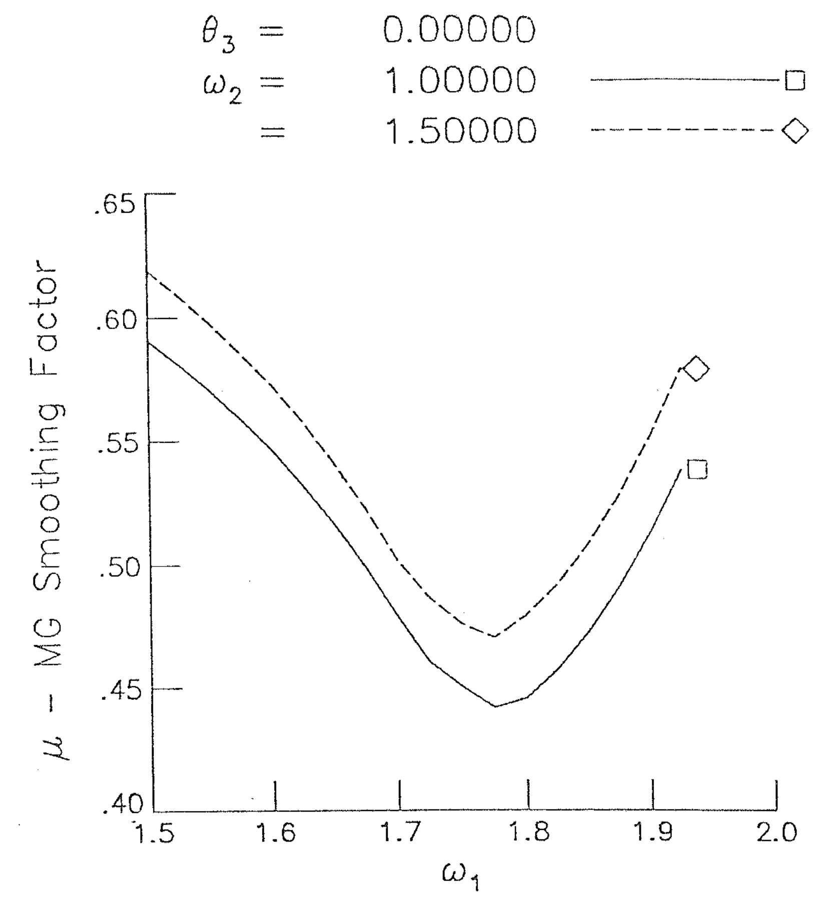
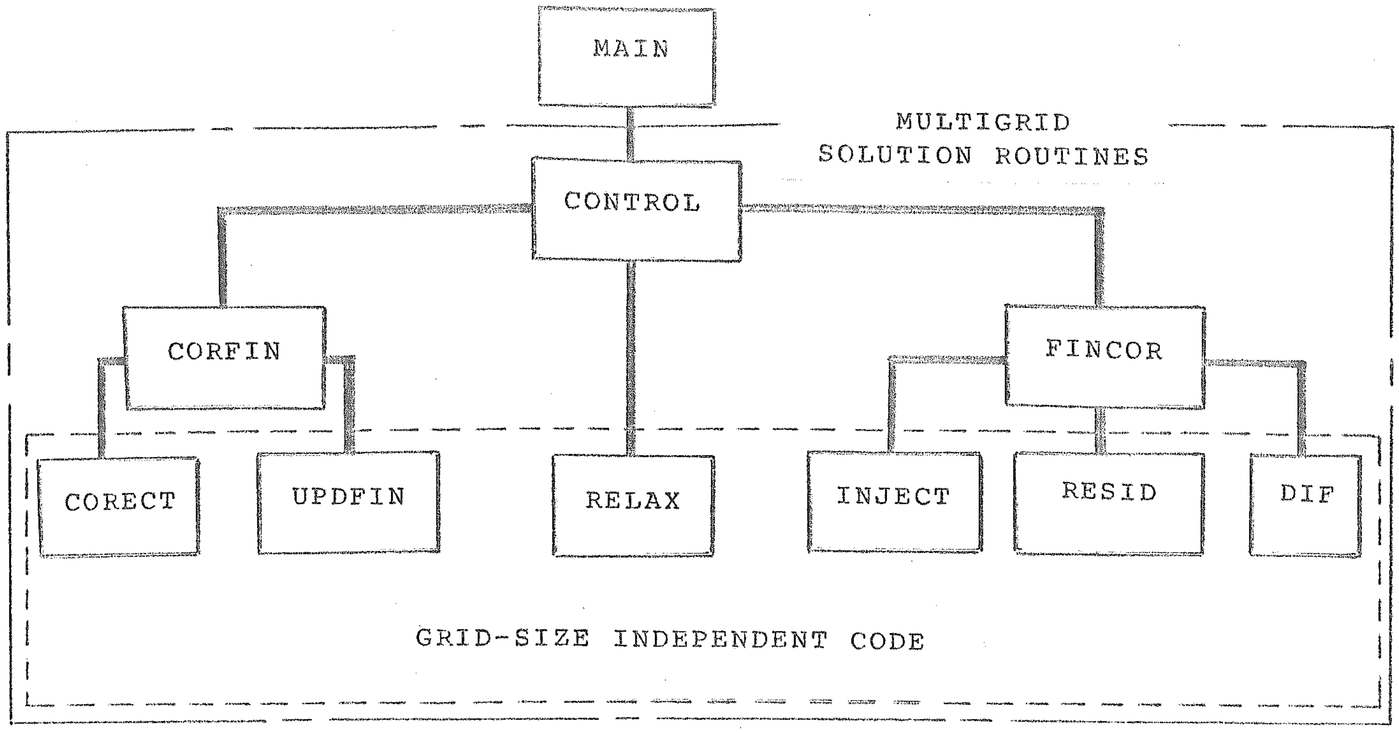


Figure 4.- Optimum multigrid smoothing factor determination for the $\xi_1:\xi_2:\xi_3$-LSOR algorithm applied to the model equation.



Routine	Purpose
CONTROL	Multigrid algorithm control (grid switching, etc.)
CORFIN	Implements coarse-to-fine grid transfer
CORECT	Computes: $C^{2H} = U^{2H} - I_H^{2H}(U^H)_{old}$
UPDFIN	Computes: $(U^H)_{new} = (U^H)_{old} + I_{2H}^{H}C^{2H}$
RELAX	Computes: $L^H U^H = F^H$
FINCOR	Implements fine-to-coarse grid transfer
INJECT	Computes: $U^{2H} = I_H^{2H}\ U^H$
RESID	Computes: $F_1 = L^{2H}(I_H^{2H}\ U^H)$ & $F_2 = I_H^{2H}(F^H - L^H U^H)$
DIF	Computes: $F^{2H} = F_1 - F_2$

Figure 5.- Modularization of the multigrid solution procedure.

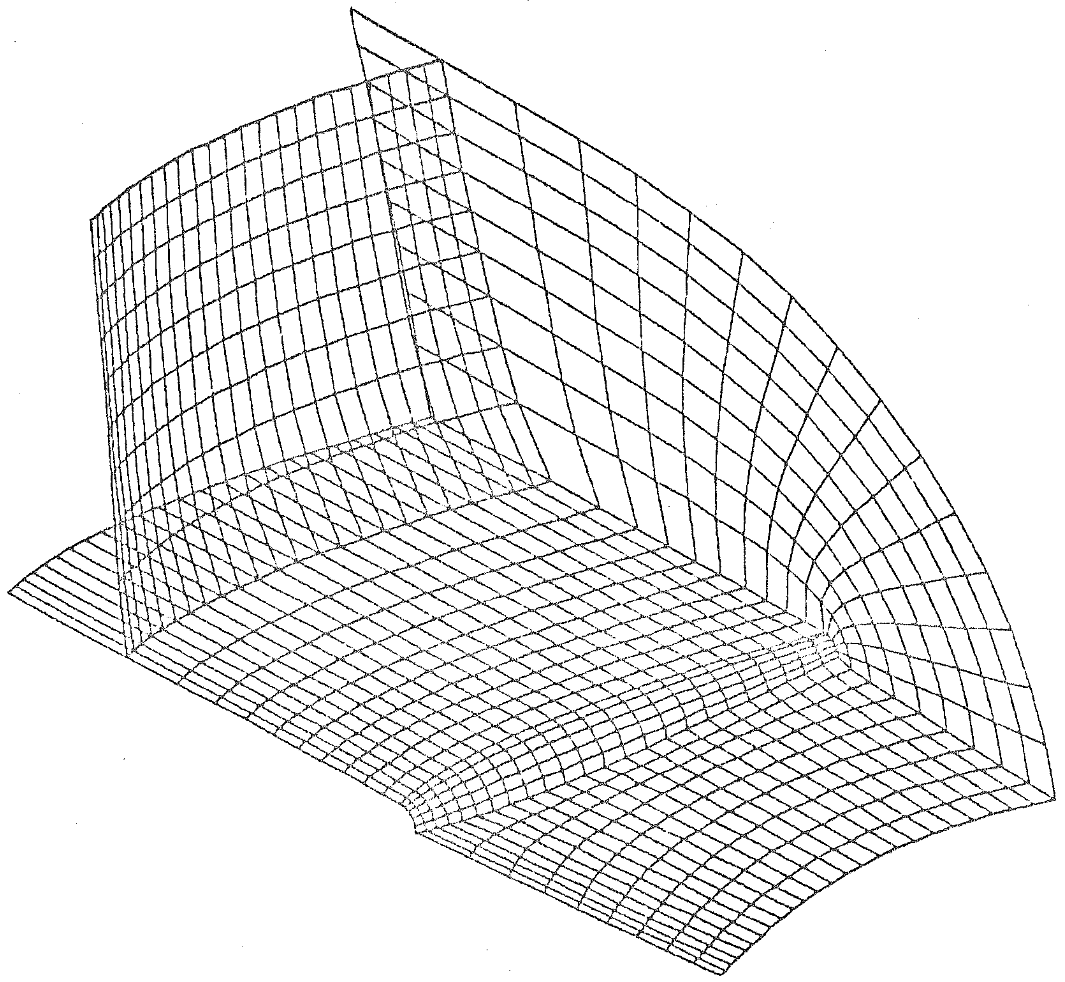


Figure 6.- Wing upper surface boundary data and computed ξ_1 and ξ_2 coordinate planes.

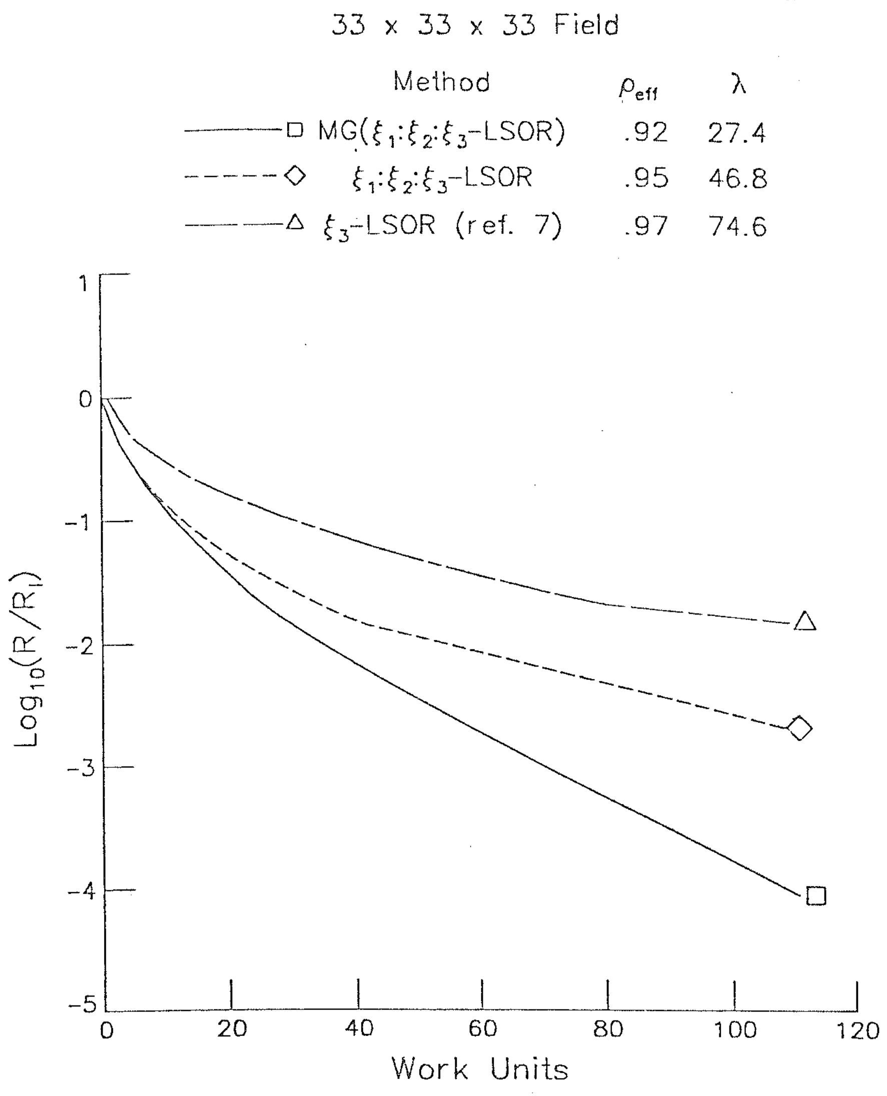


Figure 7.- Comparison of the convergence histories of three solution methods.

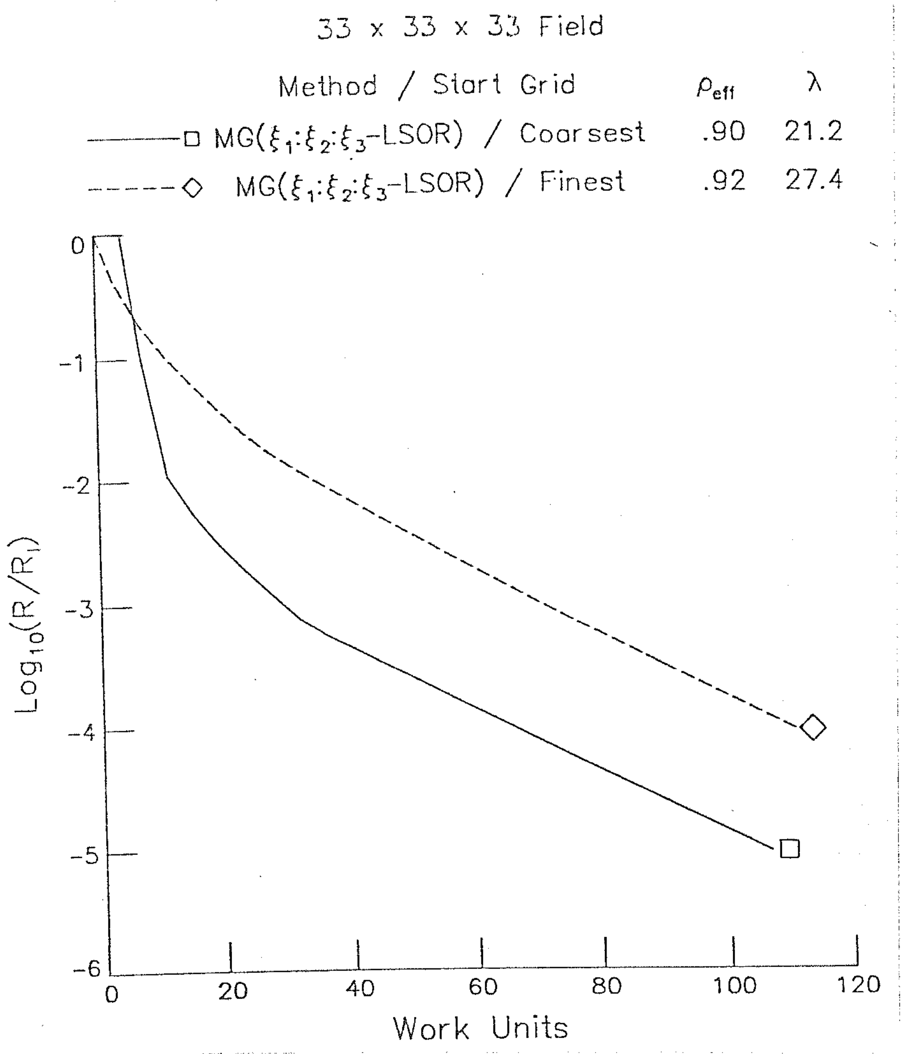


Figure 8.- Effect of starting grid on convergence rate.

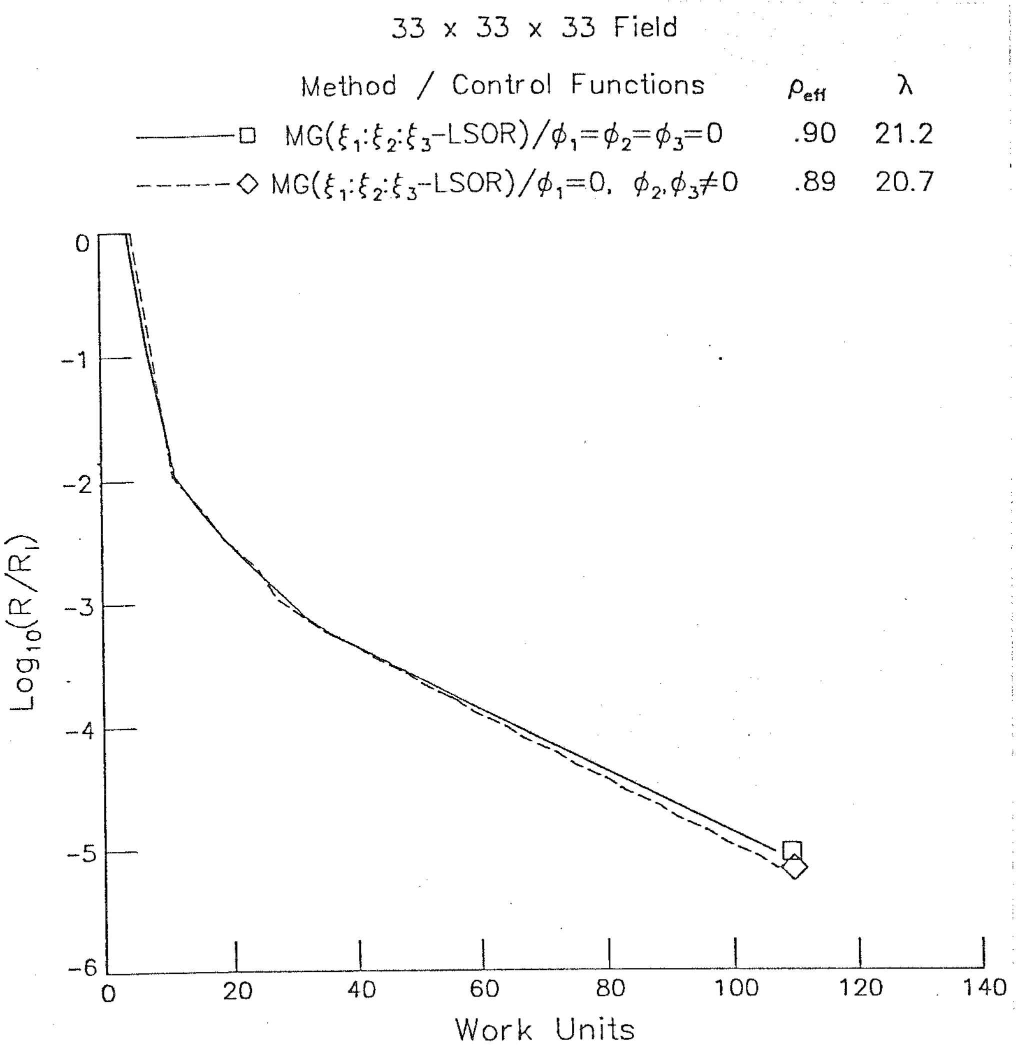


Figure 9.- Effect of coordinate control functions on convergence rate.