JOURNAL OF COMPUTATIONAL PHYSICS 104, 118–128 (1993)

A Fast Algorithm for Grid Generation

STEPHEN A. JORDAN

Naval Undersea Warfare Center, Code 8322, Bldg. 1246, Newport, Rhode Island 02841

AND

MALCOLM L. SPAULDING

Department of Ocean Engineering, University of Rhode Island, Kingston, Rhode Island 02881

Received May 10, 1990; revised November 4, 1991

The need for fast grid generation has become an important part of solving large complex computational fluid dynamic problems. A new algorithm is developed for this purpose. The two-dimensional system of equations for coordinate generation includes a procedure to force the line spacing in the interior field to reflect the adjacent boundary point distribution. The coordinate generation equations, in Newton's iteration form, are solved with a modified strongly implicit (MSI) procedure. The algorithm is applied to typical H-, O-, and C-type grids and a practical problem configured as a forward facing cavity. The grid generation equations were initialized with their Laplacian counterpart. All computations were performed on a MicroVAX 3400 computer. Based on the numerical studies, the proposed solution scheme is faster than the more common point SOR and ADI methods for large grid generation problems of practical interest. The degree of computational savings however is highly problem dependent. Extensions of the solution scheme to problems involving three-dimensions, including surfaces in three-dimensional space, are described.

INTRODUCTION

The numerical generation of grids is now an integral part of solving complex computational fluid dynamic (CFD) problems. The grid serves as the basic foundation for constructing the CFD solution. The coherence between the grid's physical characteristics and the accuracy of the CFD predictions must always be carefully considered. In grid generation systems involving curvilinear coordinates, the truncation error is dependent on the local grid spacing, rate-of-change of that spacing, and the degree of nonorthogonality [1]. A high degree of nonorthogonality is tolerable within the field when the rate-of-change of grid spacing is small. Along the physical boundaries, however, the departure from orthogonality should be kept to a minimum. Also, misalignment of the grid lines with the flow direction can introduce false diffusion where first-order accurate upwind difference schemes are used [2]. Thus, establishing a well-defined grid system which accurately resolves the flow variables is a critical step to attaining the most accurate solution at reasonable computational costs.

An excellent example of the need for careful grid design is the comparative test case described by Napolitano and Orlandi [3]. Sixteen participants used finite difference, finite volume, or finite element techniques to predict laminar flow in a complex geometry test case. In their summary, the authors note large discrepancies among the various solutions presented. Napolitano and Orlandi partially attribute the disparities to the grid systems used. They suggest that nearly identical solutions for the test case were possible if grid dependence was known a priori.

The role of numerical grid generation in CFD has received much attention since the pioneering work of Thompson *et al.* [4] appeared nearly two decades ago. In the class of structured grids, elliptic, parabolic, or hyperbolic partial differential systems are numerically solved, usually by finite differences, to produce a discretization which is specifically adapted to the particular physical domain. Schemes have been devised for grid overlapping [5], time-dependency [6], solution-adaption [7], automatic resolution control [8], sub-domains [9], multi-block [10], orthogonality [11], and many others, including surface generation [12] and variational formulations [13, 14]. Some general purpose numerical grid generators are also available and include EAGLE [15], 3DGRAPE [16], and INGRID [17]. Each possess their own distinct characteristics, but the underlying fundamentals are equivalent. A set of coupled quasi-linear elliptic equations, each representing the solution of a spatial coordinate, are solved to generate a boundary-fitted curvilinear grid in the physical domain. The primary difference between these

0021-9991/93 $5.00

three grid generators is the inclusion of some of the special features previously mentioned. The user must decide which code best suits his particular needs.

The majority of previous work in grid generation focused on exploiting the technique's flexibility for discretizing geometrically complicated domains. Most employ either an accelerated point SOR, line SOR, or ADI method for solving the coordinate system. In practical problems, having highly complex geometric boundaries and a large number of computational points, the CPU time necessary to generate an acceptable grid using these solvers can be excessively high. Also, if an adaptive grid scheme is included the percent of CPU time spent generating the grid could equal, or even exceed, that for the flow solution. In this paper, the development of a robust algorithm, hereinafter called FLAGG, is presented for the fast generation of two- and, by extension, three-dimensional grids including arbitrary surfaces in three-dimensional space.

FORMULATION

The coordinate generating system of equations employed here was originally presented by Shanks and Thompson [18] and later reformulated into generic form by Thomas [8]. Here, only the mathematical principles are presented. Additional details and applications are found in the original works [8, 18] as well as survey articles and paperbacks by Thompson *et al.* [19, 1] and others [20].

A boundary-fitted curvilinear grid in the physical domain is generated by numerically solving a set of Poisson equations which have been mathematically transformed to curvilinear space. In the two-dimensional physical domain, the curvilinear coordinates (ξ, η) are defined by the Poisson system of equations as

$$\nabla^2 \xi = g^{11} P \tag{1a}$$

$$\nabla^2 \eta = g^{22} Q. \tag{1b}$$

The source terms in Eq. (1) are used to manipulate the field line spacing and are scalar products of the contravariant metric coefficients (g^{11}, g^{22}) and the control functions (P, Q). Transformation of Eq. (1) to the curvilinear space produces a set of coupled quasi-linear elliptic equations defined as

$$g_{22}(\mathbf{r}_{\xi\xi} + P\mathbf{r}_{\xi}) - 2g_{12}\mathbf{r}_{\xi\eta} + g_{11}(\mathbf{r}_{\eta\eta} + Q\mathbf{r}_{\eta}) = 0 \tag{2a}$$

$$\mathbf{r} = \langle x, y \rangle. \tag{2b}$$

The covariant metric coefficients are

$$g_{11} = x_{\xi}^2 + y_{\xi}^2 \tag{3a}$$

$$g_{12} = x_{\xi} x_{\eta} + y_{\xi} y_{\eta} \tag{3b}$$

$$g_{22} = x_{\eta}^2 + y_{\eta}^2. \tag{3c}$$

A two-dimensional grid is generated by solving Eq. (2) with the Cartesian coordinate values of x and y on the physical boundaries assigned as the boundary conditions along the coincident curvilinear lines in the transformed space.

The elliptic system in Eq. (2) mathematically maps the physical domain onto the curvilinear (or computational) space which is orthogonal, rectangular and uniform. The mapping is one-to-one, meaning that each Cartesian point (x, y) corresponds directly to the integer indices of a single set of curvilinear coordinates (ξ, η). One-to-one mapping is guaranteed only if the Jacobian of the transformation is non-vanishing at each grid point. In the physical domain, the Cartesian coordinates are the independent variables and the curvilinear coordinates are the dependent variables. In transformed space the role of each is interchanged. Thus, by prescribing curvilinear lines along all boundaries in the physical domain, the coincident values of the Cartesian coordinates serve as Dirichlet boundary conditions for generating the grid using Eq. (2). Inasmuch as the grid is fit to the physical boundaries, geometric interpolation, as required by using rectangular grids in the physical domain, is eliminated.

In Eq. (2a), the control functions P and Q are evaluated by a procedure described by Thomas [8]. The strategy involves numerically determining the appropriate control function (either P or Q) at the two boundary end points of each curvilinear line and then linearly interpolating these values along the curvilinear line to determine the magnitudes in the field. At convergence, the line spacing in the field reflects the adjacent boundary point distribution. This is a convenient way of automatically controlling the resolution of the grid. Expressions for evaluating the control functions at the boundaries are obtained by projecting Eq. (2a) along each curvilinear line family (ξ and η) and enforcing the orthogonality condition ($g_{12} = 0$). Accordingly, setting $\mathbf{r}_{\xi} \cdot \mathbf{r}_{\eta} = 0$ and dotting $\mathbf{r}_{\xi}$ and $\mathbf{r}_{\eta}$ with Eq. (2a) gives expressions for $P(\xi)$ and $Q(\eta)$ as

$$P(\xi) = -\mathbf{r}_{\xi} \cdot \mathbf{r}_{\xi\xi}/g_{11} - \mathbf{r}_{\xi} \cdot \mathbf{r}_{\eta\eta}/g_{22} \tag{4a}$$

$$Q(\eta) = -\mathbf{r}_{\eta} \cdot \mathbf{r}_{\eta\eta}/g_{22} - \mathbf{r}_{\eta} \cdot \mathbf{r}_{\xi\xi}/g_{11}. \tag{4b}$$

By assuming that the curvature of the local transverse line emanating from the boundary is negligible ($\mathbf{r}_{\eta\eta} \simeq 0$ in Eq. (4a) and $\mathbf{r}_{\xi\xi} \simeq 0$ in Eq. (4b)), the control functions can be defined as

$$P(\xi) = -(x_{\xi} x_{\xi\xi} + y_{\xi} y_{\xi\xi})/g_{11} \tag{5a}$$

$$Q(\eta) = -(x_{\eta} x_{\eta\eta} + y_{\eta} y_{\eta\eta})/g_{22}. \tag{5b}$$

These expressions are analogous to those used in Middlecoff and Thomas [21]. Values of $P(\xi)$ are computed along boundary lines of constant η and interpolated along field lines of constant ξ. The control function $Q(\eta)$ is determined

in a similar manner. With $P(\xi)$ and $Q(\eta)$ specified by this procedure, the line spacing throughout the field is easily and automatically changed by redistributing the adjacent boundary points. All of the grids presented in this paper were generated by solving the coordinate system in Eq. (2) with the field control functions determined through linear interpolation of their boundary values as defined by Eq. (5).

SOLUTION METHODOLOGY

Newton's method, coupled with the modified strongly implicit (MSI) procedure of Schneider and Zedan [22], was used to solve the grid generation equations. In Schneider and Zedan, the MSI procedure was presented as a general purpose solver for a linear set of equations defined by a nine-point numerical molecule. Their development is an extension of Stone's five-point scheme [23]. Using sample heat conduction problems, they demonstrated significant improvements over Stone's method. The method proved to be computationally faster than point SOR, line SOR, and ADI methods. Here, the MSI procedure is implemented as part of a new solution scheme for numerical grid generation because it is especially well-suited to handle non-orthogonal terms implicitly.

The quasi-linear set of equations were first cast into Newton's iteration form and then expanded according to the nine-point MSI procedure. Although the MSI procedure is more difficult to use than simpler methods, such as point SOR, line SOR, and ADI, its computational saving is the compensating factor. By coupling the MSI procedure with Newton's form of the coordinate equation system, the programming effort is significantly reduced.

Newton's vector form [24] of the coordinate generating system in Eq. (2) is

$$\mathbf{r}^{n+1} = \mathbf{r}^n - J^{-1}(\mathbf{r})^n \, R(\mathbf{r})^n, \tag{6a}$$

where

$$R(\mathbf{r})^n = g_{22}(\mathbf{r}_{\xi\xi} + P\mathbf{r}_\xi)^n - 2g_{12}\mathbf{r}^n_{\xi\eta} + g_{11}(\mathbf{r}_{\eta\eta} + Q\mathbf{r}_\eta)^n \tag{6b}$$

and the vector $J(\mathbf{r})^n$ is the Jacobian of the residual function $R(\mathbf{r})^n$. The superscript n is an iteration parameter which represents successive updates to the coordinate vector $\mathbf{r}$. By defining a corrective quantity $\mathbf{z}^{n+1} = \mathbf{r}^{n+1} - \mathbf{r}^n$, Eq. (6a) becomes

$$J(\mathbf{r})^n \, \mathbf{z}^{n+1} = -R(\mathbf{r})^n. \tag{7}$$

The components of $J(\mathbf{r})^n$ are the coefficients of the coordinate generating system in difference form, where the metrics are assumed to be constant. Using second-order central differencing for all the spatial derivatives in Eq. (2), a nine-point numerical molecule is produced. The components of the Jacobian vector at a field point are

$$J(\mathbf{r}) = [1/2g_{12},\ ag_{11},\ -1/2g_{12},\ bg_{22},\ -2(g_{11}+g_{22}),\ cg_{22},\ -1/2g_{12},\ dg_{11},\ 1/2g_{12}] \tag{8}$$

where

$$a = 1 - Q/2$$
$$b = 1 - P/2$$
$$c = 1 + P/2$$
$$d = 1 + Q/2.$$

The metric coefficients in $J(\mathbf{r})$ are determined by Eq. (3) and the control function field magnitudes are obtained through linear interpolation of the boundary values defined by Eq. (5). For generating two-dimensional grids, the iterative solution of each Cartesian coordinate, in Newton's form, is

$$J(\mathbf{r})^n \begin{bmatrix} z_x \\ z_y \end{bmatrix}^{n+1} = -\begin{bmatrix} R(x) \\ R(y) \end{bmatrix}^n \tag{9a}$$

with

$$x^{n+1} = x^n + z_x^{n+1} \tag{9b}$$

$$y^{n+1} = y^n + z_y^{n+1}. \tag{9c}$$

The residuals $R(x)$ and $R(y)$ are defined by Eq. (6b) with (2b). Numerical generation of a grid is achieved by successively solving Eq. (9a) for both z_x and z_y, followed by updates of the field point values x and y as defined in Eqs. (9b) and (9c), respectively. Grid convergence is achieved when the root-mean-squares (RMS) of both $R(x)$ and $R(y)$ are below an error tolerance specified by the user.

Solution of the coordinate equation system in Eq. (9a) is accomplished through implementation of the nine-point MSI procedure by Schneider and Zedan [22]. Accordingly, the coefficient matrix $J(\mathbf{r})$ is modified to minimize the computational effort required for its decomposition. This is done by introducing four new entries into the numerical molecule, as illustrated in Fig. 1, that expand into four additional diagonals in the coefficient matrix $J(\mathbf{r})$.

These new diagonals have an asymmetric influence in the modified coefficient matrix that is partially cancelled by performing a Taylor series expansion of the original nine quantities to determine the value of the four new entries. An iterative parameter α is also introduced as part of the partial

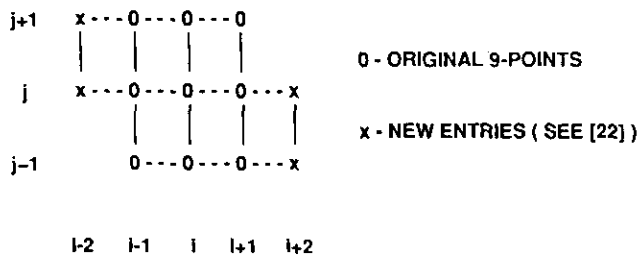


FIG. 1. Computational molecule of the nine-point MSI procedure.

cancellation strategy. Schneider and Zedan tested a wide range of values for α between 0.1 and 1.0 and determined an optimum value near 0.5. Since the complete development of the MSI procedure is rather long, only its main features, specific for implementation into Newton's form of the grid generation system, are presented below. The iteration sequence of the resultant solution scheme, however, is described in its entirety.

Assuming that the four additional diagonals are contained in a new matrix $G(\mathbf{r})$, Eq. (7) is redefined as

$$[J(\mathbf{r}) + G(\mathbf{r})]^n \, \mathbf{z}^{n+1} = -R(\mathbf{r})^n, \tag{10}$$

where the sum of $J(\mathbf{r})$ and $G(\mathbf{r})$ comprise the modified coefficient matrix. The construction of $G(\mathbf{r})$ is such that the decomposition of the modified coefficient matrix into an upper and lower triangular matrix product is more computationally efficient than decomposing the original matrix $J(\mathbf{r})$ by itself. The iterative form of Newton's method, expanded according to the MSI procedure, is

$$\begin{aligned}[J(\mathbf{r}) + G(\mathbf{r})]^n \, \mathbf{z}^{n+1,m+1} &= [J(\mathbf{r}) + G(\mathbf{r})]^n \, \mathbf{z}^{n+1,m} \\ &\quad - [R(\mathbf{r})^n + J(\mathbf{r})^n \, \mathbf{z}^{n+1,m}],\end{aligned} \tag{11}$$

where, again, the outer iteration n represents successive corrections to the field point values of x and y. The second superscript m signifies the inner iterations for computing the corrective quantity $\mathbf{z}$ by the MSI procedure. Letting $\boldsymbol{\delta}^{m+1} = \mathbf{z}^{n+1,m+1} - \mathbf{z}^{n+1,m}$, then Eq. (11) is rewritten such that

$$\begin{aligned}[J(\mathbf{r}) + G(\mathbf{r})]^n \, \boldsymbol{\delta}^{m+1} &= -[R(\mathbf{r})^n + J(\mathbf{r})^n \, \mathbf{z}^{n+1,m}].\end{aligned} \tag{12}$$

Decomposition of the modified coefficient matrix into an upper and lower matrix product appears as

$$L(\mathbf{r})^n \, U(\mathbf{r})^n = [J(\mathbf{r}) + G(\mathbf{r})]^n \tag{13}$$

which introduces a two-step forward and backward substitution process for updating the corrective quantity $\mathbf{z}$. After substituting Eq. (13) into Eq. (12) and redefining the source term, the entire iteration sequence in Newton's form coupled with the MSI procedure is

$$F(\mathbf{r})^{n,m} = -[R(\mathbf{r})^n + J(\mathbf{r})^n \, \mathbf{z}^{n+1,m}]. \tag{14a}$$

$$L(\mathbf{r})^n \, \mathbf{v}^{m+1} = \omega F(\mathbf{r})^{n,m} \tag{14b}$$

$$U(\mathbf{r})^n \, \boldsymbol{\delta}^{m+1} = \mathbf{v}^{m+1} \tag{14c}$$

$$\mathbf{z}^{n+1,m+1} = \mathbf{z}^{n+1,m} + \boldsymbol{\delta}^{m+1} \tag{14d}$$

$$\mathbf{r}^{n+1} = \mathbf{r}^n + \mathbf{z}^{n+1}, \tag{14e}$$

where $R(\mathbf{r})$ and $J(\mathbf{r})$ are defined by Eqs. (6b) and (8), respectively. Note, that $F(\mathbf{r})^{n,m}$ is multiplied by a relaxation parameter ω in Eq. (14b) to permit some flexibility over the total number of outer iteration cycles. For brevity, complete definitions of the components in the lower $L(\mathbf{r})$ and upper $U(\mathbf{r})$ matrices are not shown, but can be found in [22].

The inner iteration m represents solution of the corrective quantity $\mathbf{z}$ by the MSI procedure. Obviously, tight convergence of this quantity is not necessary, and usually the inner iterations are terminated when $F(\mathbf{r})_{\mathrm{RMS}} \leqslant 0.1 \, R(\mathbf{r})_{\mathrm{RMS}}$. Updating the Cartesian coordinate values followed by updating the metric coefficients are part of the outer iteration loop n. Thus, by lagging the metric coefficients in the inner loop, the quasi-linear coordinate system is fully linearized which permits direct implementation of the MSI procedure. A single outer iteration involves the complete system in Eq. (14) of which only one Cartesian coordinate is operated on at a time.

The coordinate equation system could be solved using easier methods, but because the equations are recast into Newton's form, implementation of a more complicated procedure is greatly simplified. The simplification lies in the specification of Dirichlet boundary conditions. Since the residuals $R(x)$ and $R(y)$ are zero-valued everywhere on the boundaries, so are the corresponding corrective quantities. Consequently, at the computational points adjacent to the boundary, it is not necessary to modify the source term (or residual) to account for the product of the corrective term $\mathbf{z}$ with the boundary terms in the coefficient matrix $J(\mathbf{r})$. For instance, the computation at a point which is directly adjacent to a concave corner point, would not require the expansion of the source term (residual) to include the information along the two adjacent boundaries; i.e., five points. This condition greatly simplifies implementation of the MSI procedure since the program accounting necessary for tracking the location of the boundaries is eliminated. The result is a scheme which is easy to program and especially attractive for domains with internal boundaries.

NUMERICAL STABILITY

During iterative solution of the grid generation system of equations, one must be concerned about numerical stability.

In a Laplacian system with Dirichlet type boundary conditions, the maximum principle guarantees a smooth regular grid. However, the Poisson system weakens this principle [25]. Depending on the magnitude of the control functions its solution, using an iterative method, may become unstable. For this reason, the Poisson system in Eq. (1) was chosen to control the field line spacing instead of a system, where the source term is defined by the control functions alone [26]. In the latter, the likelihood of unstable behavior is increased because, by comparison, the control functions can be several orders of magnitude larger. Thus, besides allowing easy access to the field line spacing through the distribution of the adjacent boundary points, the system in Eq. (1) reduces the adverse effect that the control functions have on numerical stability.

Even using the Poisson system just described, small changes in the control functions can still lead to divergence. This condition was mathematically illustrated by Sonar [27] in an example problem, where a linear Gauss–Seidel method was used to solve a coordinate generation system of equations similar to the one presented here. An easy remedy suggested by Sonar was to first set the control functions equal to zero and generate a Laplacian grid from the initial grid. An initial grid in this case could be attained by a simple algebraic procedure. After converging to a Laplacian grid, the control functions are implemented and the iterative process is continued until convergence to the final Poisson grid. This approach appears reasonable for two reasons: (i), if the sign of a particular control function in the initial grid is opposite to that in the final grid, then the change in the control function could be minimized by starting with zero, (ii), by starting with a Laplacian grid, any divergent behavior detected during convergence of the Poisson grid could be easily controlled by damping the rate of change of the control functions. For these reasons, the approach recommeded by Sonar was tested for generating example H-, O-, and C-grids, as well as a practical problem.

SAMPLE APPLICATIONS

A set of examples involving H-, O-, and C-type grids and a practical problem configured as a forward-facing cavity are used to benchmark the efficiency of the new solution scheme FLAGG. The efficiency is demonstrated by comparing the CPU times of FLAGG to those of the more commonly used point SOR (PSOR) and ADI methods. No special consideration was given to optimizing the acceleration parameter at each grid point in any of the solution methods; however, an optimum constant value was implemented for each method in the traditional way [1]. In these test cases, a Laplacian grid was generated from an initial algebraic grid and then the iteration process proceeded to convergence of the Poisson grid as recommended by Sonar [27]. The iterations were terminated when the residual RMS value fell below a prespecified error tolerance. The maximum RMS value was computed as

$$R(x, y)_{\mathrm{RMS}} = \mathrm{SQRT}\left(\frac{1}{M}\sum_{i=1}^{M} R(x_i, y_i)^2\right), \qquad (15)$$

where M is the total number of computational points. In this section, the convergence behavior for the Laplacian and Poisson grids are shown to examine the change in the RMS value when the control functions were activated.

Example cases for the H-, O-, and C-type topologies are shown in Figs. 2, 3, and 4, respectively. The initial (algebraic), Laplacian, and Poisson phases of each example case (Figs. 2a, 3a, and 4a) are shown for the coarsest grid studied (lowest number of computational points). Each initial grid was computed algebraically by linearly interpolating the coordinate boundary values in the η direction only. Convergence of the coarsest grid of each topology was achieved when the RMS value reached an error tolerance of 10^{-7} for the Laplacian grid and 10^{-8} for the Poisson grid. Since a tight convergence criteria on the Laplacian grid wastes computational resources, its error tolerance was defined to be an order of magnitude higher. Figures 2b, 3b, and 4b show the convergence behavior of the coarsest grid of each topology using the three iterative solvers. The influence of mesh refinement on the total computational time for each topology is shown in Figs. 2c, 3c, and 4c. These CPU times were normalized with respect to the time required by FLAGG, meaning that values greater than one for the PSOR and ADI methods signify longer times for convergence compared to the time required by the FLAGG routine.

During convergence of the coarsest H-shape grid (31×31 grid having 720 computational points) (Fig. 2b), an abrupt jump in the RMS value occurred when the control functions were introduced using the Laplacian grid as the initial condition. This jump shows that the RMS value to begin convergence of the Poisson grid was higher than that of the Laplacian grid and resulted in a large increase in the total number of outer interations. Figure 2b indicates that the PSOR method required more than four times as many iterations as FLAGG to converge the coarsest H-shape grid. This is reflected in the normalized CPU times shown in Fig. 2c, where the PSOR method required 273% more time compared to the FLAGG scheme. For the coarsest H-shape, the ADI method attained convergence faster than the PSOR method, but slower than the FLAGG scheme. The computational savings, using the FLAGG routine, gradually improved over the PSOR and ADI schemes as the grid was refined. For example, approximately 385% more CPU time was necessary to attain convergence using the PSOR method than by FLAGG when the grid spacing was

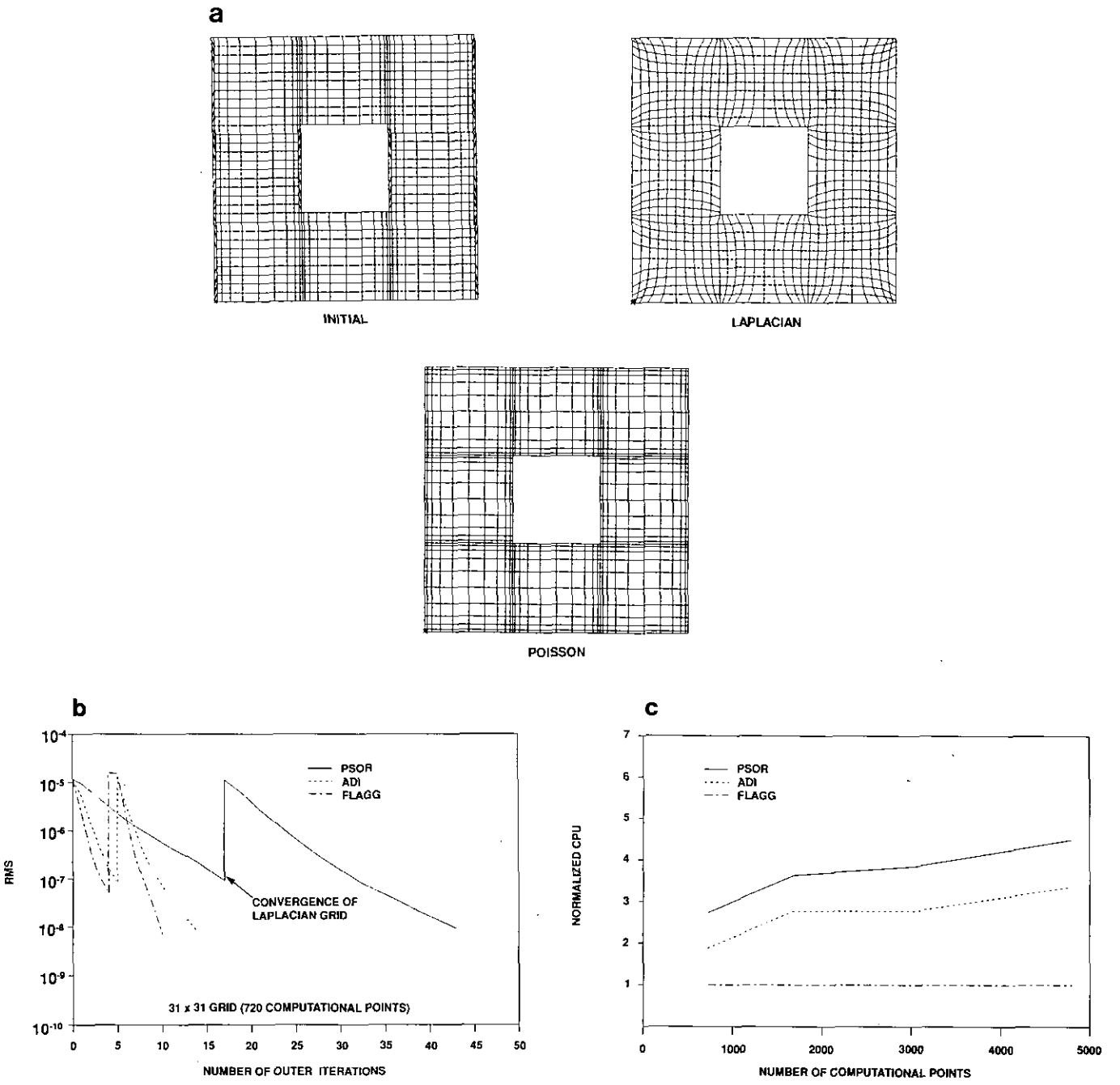


FIG. 2. H-shape example problem. (a) Initial, Laplacian, and Poisson phases of grid convergence for the coarsest grid (31 × 31 grid with 720 computational points). (b) Convergence behavior of the coarsest grid. (c) Normalized CPU times (with respect to the FLAGG time) using the PSOR, ADI, and FLAGG iterative solvers.

refined to one-half of its original size (61 × 61 grid with 3040 computational points). For this same grid, the ADI method required 278% additional time compared to FLAGG. Moreover, these improvements are conservative, since the smaller number of iterations inherent in FLAGG usually resulted in lower RMS values at convergence.

The initial, Laplacian, and Poisson phases of grid convergence for the O-type topology, shown in Fig. 3a, are each a 60 × 20 grid with 1160 computational points. Each represents the coarsest O-type grid tested. The initial grid was algebraically produced using stretching functions to cluster η lines near the inner and outer boundaries. The Laplacian grid was numerically generated with fixed points along the branch cut. Since generation of the Laplacian grid

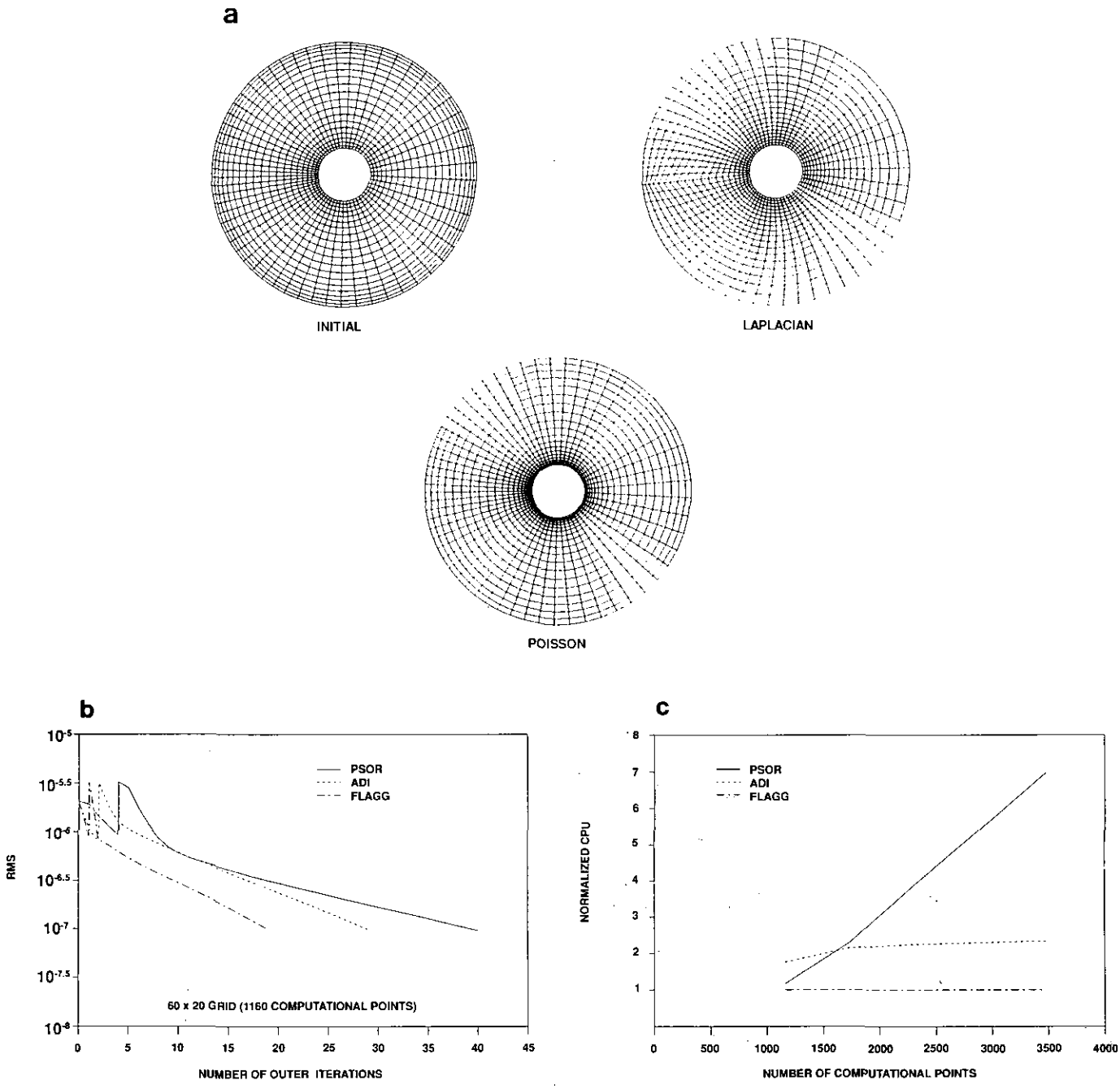


FIG. 3. O-type grid example problem. (a) Initial, Laplacian, and Poisson phases of grid convergence for the coarsest grid (60×20 grid with 1160 computational points). (b) Convergence behavior of the coarsest grid. (c) Normalized CPU times (with respect to the FLAGG time) using the PSOR, ADI, and FLAGG iterative solvers.

is only an intermediate step to facilitate numerical stability, increasing the CPU time by including the branch cut in the field computation is wasteful. However, as shown in Fig. 3a, fixing these points introduced some undesirable non-orthogonality in the form of distorted ξ lines. These distortions were later removed in the Poisson grid by including the branch cut within the computational field. Note that by allowing these points to float, the η line clustering shown in the initial grid was relaxed near the outer boundary while intensified at the inner boundary in the Poisson grid. This illustrates a strong attraction of the coordinate generating system to boundaries with convex curvature and a repulsion from concave boundaries.

The convergence rates of the PSOR, ADI, and FLAGG

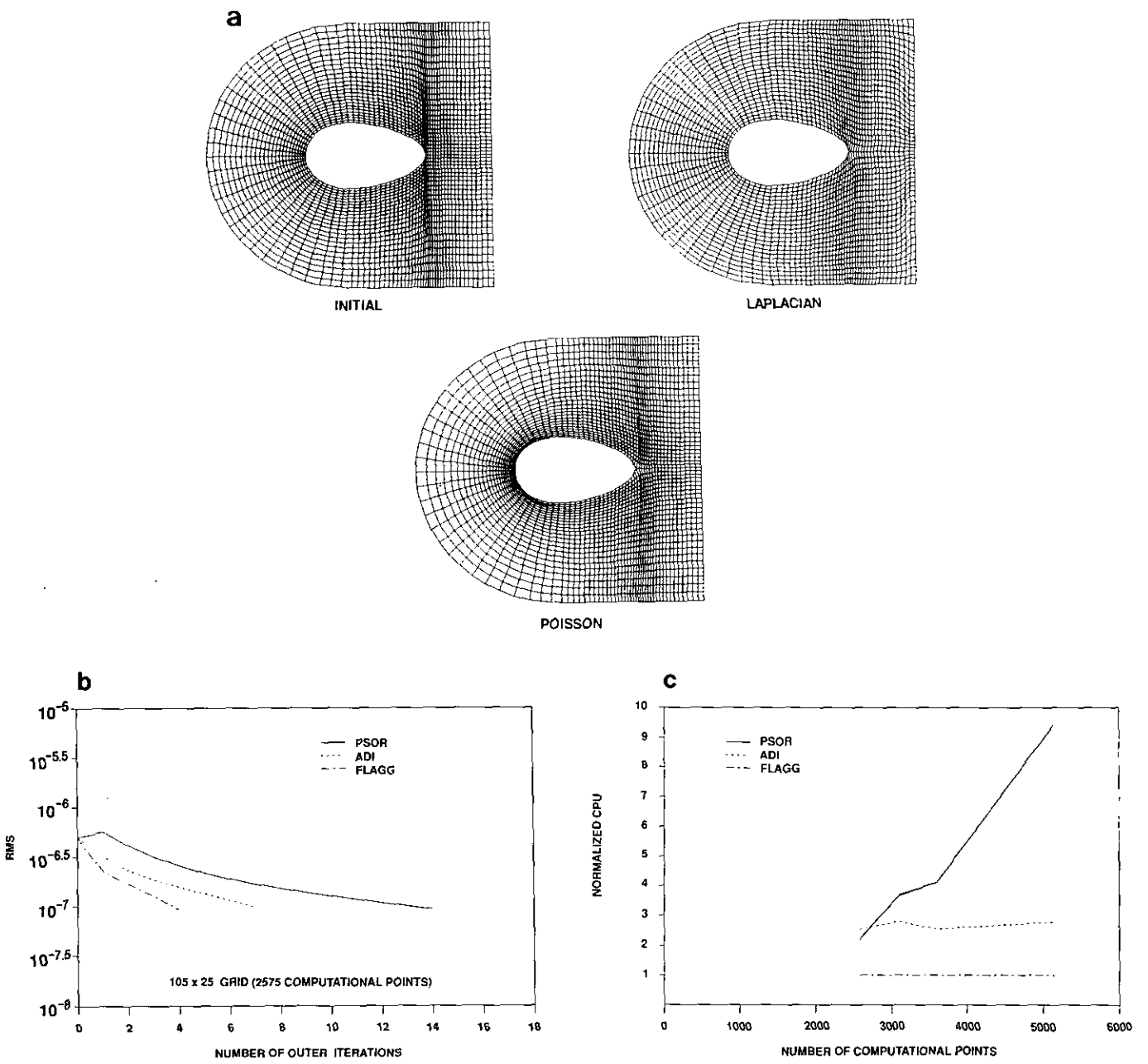
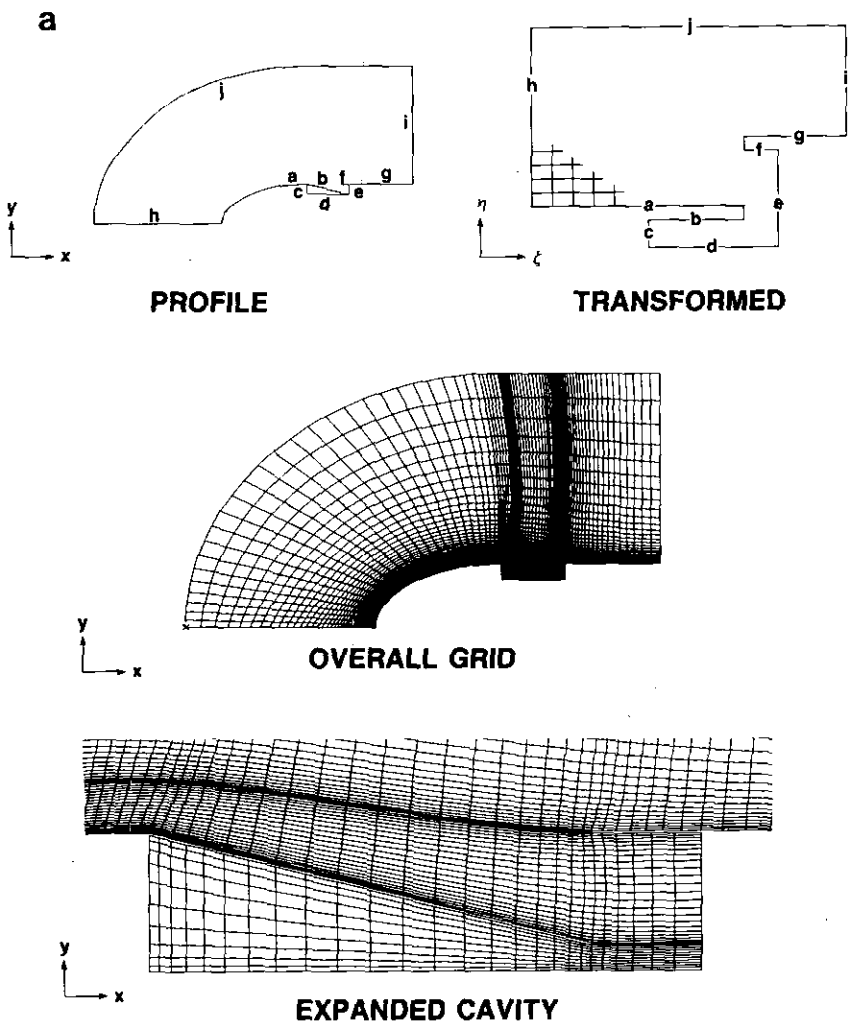
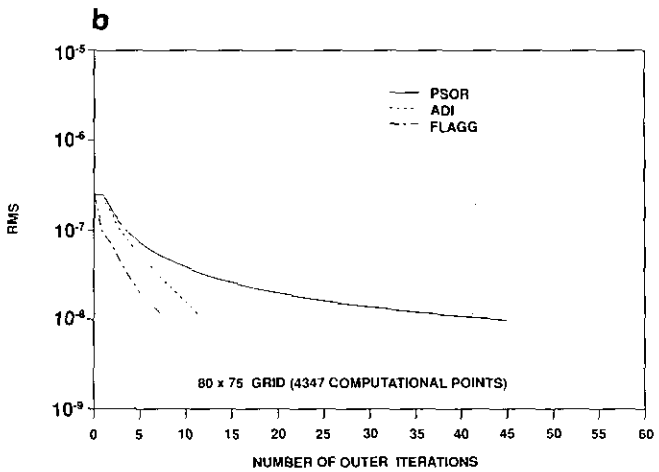


FIG. 4. C-type grid example problem. (a) Initial, Laplacian, and Poisson phases of grid convergence for the coarsest grid (105 x 25 grid with 2575 computational points). (b) Convergence behavior of the coarsest grid. (c) Normalized CPU times (with respect to the FLAGG time) using the PSOR, ADI, and FLAGG iterative solvers.

schemes are displayed in Fig. 3b for the coarsest O-type grid. Like the H-shape, RMS values exceed the initial condition (algebraic grid) when the control functions were activated in the Laplacian grid. This suggests that the intermediate step of generating a Laplacian grid to serve as the initial condition for beginning convergence of the Poisson grid, should be eliminated. In this case, damping the rate of change in the control functions to inhibit divergent behavior would start with the initial algebraic guess. The CPU times in Fig. 3c show that in the coarsest O-type grid, the pair of periodic Neumann type boundary conditions along the re-entrant boundaries, nullified the advantage of using the ADI or FLAGG schemes that was previously seen for the H-shape. As the O-type grid was refined, however, the computational savings of the FLAGG routine were significant compared to the PSOR method and moderate over the ADI technique. For instance, by doubling the grid size to 60 x 40 (2320 computational points), the PSOR method required





c

Forward Facing Cavity 80x75 Grid	
Solver	CPU
PSOR	4.19
ADI	1.97
FLAGG	1.00

FIG. 5. *Forward facing cavity problem (80 × 75 grid; 4347 computational points).* (a) *Boundary mapping strategy, domain grid point distribution,* and fine grid details within cavity region. (b) Convergence behavior of grid using the PSOR, ADI, and FLAGG iterative solvers. (c) Normalized CPU times of each solver with respect to FLAGG time.

392% more CPU time relative to FLAGG. Comparatively, the ADI technique was much faster, but still needed 225% additional time. This result is not only a consequence of increasing the number of computational points in the O-type grid, but is primarily caused by allowing the points along the branch cut to move to achieve a smooth grid with continuous coordinate derivatives throughout the physical domain.

The last example topology is a C-type grid as displayed in Fig. 4. Each phase of the coarsest grid is illustrated in Fig. 4a while the convergence behaviors and normalized CPU times of the three solution methods are shown in Figs. 4b and 4c, respectively. Like the O-type, the points along the branch cut remained fixed during convergence of the Laplacian grid which slightly perturbs the ξ lines between the body contour and the outer boundary. By allowing these points to float during convergence of the Poisson grid, the perturbations disappeared. The convergence curves for the coarsest grid (105×25 grid; 2575 computational points) shows the RMS values after the Laplacian grid was attained and the control functions activiated. Once again, the FLAGG routine achieved convergence of the coarsest C-type grid in fewer iterations than the PSOR and ADI schemes.

The boundary conditions of the C-type grid can be viewed as a compromise between those in the H- and O-type topologies; that is, periodic Neumann type boundary conditions are partially imposed along the lowest curvilinear line [1]. A comparison of the CPU times shown in Fig. 4c with those in Figs. 2c and 3c reflect this. When the C-type grid was refined, the computational saving of the FLAGG scheme compared to the PSOR and ADI methods was greater than that for the H-shape, but less than for the O-type.

The final problem used for comparing the computational efficiency of the three solvers has a geometry configured as a forward-facing cavity. This type of cavity often appears in marine vessels. Its basic geometric features are profiled in Fig. 5a. The forward facing cavity possesses an entrance doorway (b) which is hinged at the forward point. Usually, these cavities are designed either for an operational purpose, such as to permit access to the exterior, or to serve as a gateway for deployment of devices such as array moorings, buoys, and tethered vehicles.

The overall mapping strategy for the forward facing cavity was introduced in a slightly different form by Jordan and Spaulding [28] and later refined by Salari and Roache [29]. In Fig. 5a, one can see that the cavity entrance region, above the entrance doorway (b), is exposed directly to the oncoming flow. Thus, the respective physical boundaries are mapped onto the curvilinear boundary lines so that the mesh is easily refined within that specific area. The mesh below the entrance doorway is independent of the point spacing directly above it, until rearward of the doorway, where the two constitute the aft cavity area. Separating these two regions permits local control over each for the purpose of minimizing the total number of computational points. This mapping introduces two internal boundaries and four convex points that pose no special considerations when using the FLAGG routine.

A single grid (80×75) comprising 4347 computational points was generated for the forward-facing cavity geometry and is displayed in Fig. 5a. From a CFD standpoint, this grid would probably be adequate for analysis of a potential or low Reynolds number flow. Figure 5b presents the convergence behavior of the three solution methods after the Laplacian grid was attained. Figure 5c shows the individual CPU times for each method, normalized with respect to the time required by the FLAGG routine. The convergence curves show a substantial reduction in the total number of outer iterations using the FLAGG routine in comparison to the PSOR method and a moderate savings over the ADI technique. During convergence of the forward-facing cavity grid, no sign of unstable behavior was detected. Comparisons of the CPU times show that the PSOR and ADI methods required 419% and 197% more computational effort than FLAGG, respectively. Generation of the forward-facing cavity grid using the FLAGG scheme required approximately 30 s of CPU time on a MicroVAX 3400 computer.

CONCLUSION AND EXTENSIONS

The details of a new solution scheme, FLAGG, developed specifically for the numerical generation of two-dimensional grids is presented. The scheme was tested, in terms of computational efficiency, against the more popular point SOR and ADI methods for generating three example grid topologies (H, O, and C) commonly used in CFD applications, and a practical problem geometry configured as a forward facing cavity. Based upon direct comparisons among the CPU times required for each method, the FLAGG scheme is the most efficient for generating large grids of practical interest. However, the relative savings in computational effort is highly dependent on the grid topology chosen and its associated boundary conditions. For example, in the O-type grid, the computational savings of FLAGG over the point SOR and ADI methods was substantially higher than for the H- and C-type configurations. This result is due to the pair of periodic Neumann boundary conditions imposed along two of the four boundaries in the O-type grid.

Although the test cases shown were limited to two-dimensions, this restriction is not inherent in the FLAGG solution scheme. The scheme is equally applicable for generating three-dimensional grids and arbitrary surfaces. Three-dimensional grids could be numerically generated by

visualizing the new solution scheme as an implicit solver of a planar surface passing through the cube (transformed volume) in a direction which is normal to the surface. The scheme would sweep through the volume analogous to the manner a line solver sweeps along a planar surface. In this case, the terms involving derivatives of the direction coordinate would be treated as knowns and included as part of the residuals. Arbitrary surface generation in three-dimensional space also poses no difficulties. For example, in a paper by Warsi and Tiarne [30], the basic elliptic operator is two-dimensional. The FLAGG solution strategy could be applied directly to this approach. Thus, the FLAGG scheme has sufficient versatility for generating three-dimensional grids as well as arbitrary surfaces in three-dimensional space.

ACKNOWLEDGMENTS

This research was sponsored by the Office of Naval Technology under the Independent Exploratory Development program (Dr. K. M. Lima, coordinator) at the Naval Undersea Warfare Center. The authors wish to thank the JCP reviewers for their useful comments to improve the content of this manuscript.

REFERENCES

1. J. F. Thompson, Z. U. A. Warsi, and C. W. Mastin, *Numerical Grid Generation* (North-Holland, New York, 1985).
2. S. V. Patankar, *Numerical Heat Transfer and Fluid Flow* (Hemisphere, Washington, DC, 1980).
3. M. Napolitano and P. Orlandi, *Int. J. Numer. Methods Fluids* **5**, 667 (1985).
4. J. F. Thompson, F. C. Thames, and C. W. Mastin, *J. Comput. Phys.* **15**, 299 (1974).
5. E. H. Atta and J. Vadyak, *AIAA J.* **21**, 1271 (1983).
6. P. D. Thomas and C. K. Lombard, *AIAA J.* **17**, 1030 (1980).
7. D. A. Anderson, *Proceedings, AIAA Twenty-Fourth Aerospace Sciences Meeting*, AIAA 86-0427, 1986.
8. P. D. Thomas, *Proceedings, AIAA Fifth Computational Fluid Dynamics Conference*, AIAA 81-0996, 1981.
9. P. D. Thomas, *Numerical Grid Generation*, edited by J. F. Thompson (North-Holland, New York, 1982).
10. J. F. Thompson, *Proceedings, AIAA Twenty-Fifth Aerospace Science Meeting*, AIAA 87-0275, 1987.
11. P. R. Eiseman, *Numerical Grid Generation*, edited by J. F. Thompson (North-Holland, New York, 1982).
12. Z. U. A. Warsi, *J. Comput. Phys.* **64**, 82 (1986).
13. J. V. Brackbill and J. S. Saltzman, *J. Comput. Phys.* **46**, 342 (1982).
14. S. Steinberg and P. J. Roache, *Num. Methods Part. Diff. Eqs.* **2**, 71 (1986).
15. J. F. Thompson, Program EAGLE Numerical Grid Generation System User's Manual, Vols. II, III, AFATL-TR-87-15, 1987 (unpublished).
16. R. L. Sorenson, *Proceedings, Numerical Grid Generation in Computational Fluid Mechanics '88*, edited by S. Sengupta *et al.* (Pineridge, Swansea, 1988).
17. E. W. Dorrel, Jr., and M. D. McClure, 3DE INGRID; Interactive Three-Dimensional Grid Generation, AEDC-TR-87-40, 1987 (unpublished).
18. S. P. Shanks and J. F. Thompson, *Proceedings, Second International Conference on Numerical Ship Hydrodynamics, 1977*, p. 202.
19. J. F. Thompson, Z. U. A. Warsi, and C. W. Mastin, *J. Comput. Phys.* **47**, 1 (1982).
20. P. R. Eiseman, *Annu. Rev. Fluid Mech.* **17**, 487 (1985).
21. J. F. Middlecoff and P. D. Thomas, *Proceedings, AIAA Fourth Computational Fluid Dynamics Conference*, AIAA 79-1462, 1979.
22. G. E. Schneider and M. Zedan, *Numer. Heat Transfer* **4**, 1 (1981).
23. H. L. Stone, *SIAM J. Numer. Anal.* **5**, 530 (1968).
24. L. E. Johnson and R. D. Riess, *Numerical Analysis* (Addision–Wesley, Philippines, 1977).
25. R. Courant and D. Hilbert, *Methods of Mathematical Physics* (Wiley, New York, 1962).
26. J. F. Thompson, F. C. Thames, and C. W. Mastin, *J. Comput. Phys.* **24**, 274 (1977).
27. T. Sonar, *Grid Generation Using Elliptic Partial Differential Equations*, Report IB 129 - 88/11 (DFVLR Braunschweig, D-3300 Braunschweig, Flughafen, 1988).
28. S. A. Jordan and M. L. Spaulding, *Proceedings, Symposium on Hydrodynamic Performance Enhancement for Marine Applications, 1988*, p. 107.
29. K. Salari and P. J. Roache, *Proceedings, AIAA Twenty-Eighth Aerospace Sciences Meeting and Exhibit, 1990.*
30. Z. U. A. Warsi and W. N. Tiarn, *Proceedings, Numerical Grid Generation in Computational Fluid Mechanics '88*, edited by S. Sengupta *et al.* (Pineridge, Swansea, 1988).