JOURNAL OF COMPUTATIONAL PHYSICS 119, 142–148 (1995)

Mesh Generation Using Vector-Fields

P. KNUPP

Ecodynamics Research Associates, P.O. Box 9229, Albuquerque, New Mexico 87119

Received March 21, 1994; revised December 1, 1994

A method for generating structured meshes from vector fields is described. The method is based on a weighted variational principle that minimizes a least-squares fit to the inverse Jacobian matrix. The weight is constructed from the vector fields by requiring local alignment of the mesh with the fields and by prescribing local length scales. The resulting grid generator is a weighted form of the Laplace grid generation equations. The method is demonstrated in 2D using a subsurface flow-field derived from hydrologic simulations.

1. INTRODUCTION

The idea of solving Laplace's equations as a Dirichlet problem has enjoyed widespread acceptance as a means of generating unfolded boundary-fitted meshes on arbitrary domains (the idea goes back at least as far as the paper [18]). In practice, these equations are little used due to the need to better adapt the mesh in particular portions of the domain to the shape of the domain or to features of the solution to the hosted physical equations. To meet this need a wide range of grid generation methods have been proposed.[1] Elliptic methods based on ad hoc weightings of Laplace's equations have been proposed to adapt the mesh to the domain geometry [16, 17] or to permit solution adaptivity [1, 10, 8]. Some approaches abandon elliptic grid generation methods altogether in order to gain speed and precise mesh control [5, 14].

Variational generators control meshes through weighted combinations of functionals for smoothness, area, and orthogonality [2, 7, 15]. Although effective in some instances, these methods may lack convexity in the variational principle (particularly if insufficient attention is paid to the relative weightings between principles). Weighting also creates scaling problems when weighting dimensionally incompatible principles. Nonetheless, variational methods remain attractive in that weighted forms seem less likely to produce ad hoc generators than the other approaches.

Nearly all of the methods mentioned so far attempt to control the elements of the metric tensor (e.g., by controlling cell-lengths or orthogonality [12]). Surprisingly little attention has been paid to using the vector fields associated with the solution to the hosted equations as a means of assisting in the mesh generation step. Alignment of the computational grid with the vector-field associated with flow streamlines has long been known to be an effective, if little-used, means of improving the accuracy of a calculation [4, 13]. It is clear that such a capability would also permit generation of boundary-fitted coordinates on problems in which user-specified vector-fields are derived from fluxes, solution gradients, vorticity vectors, or even the domain boundary.

Giannakopoulos and Engel proposed a variational principle that minimizes the cross-product of the mesh tangents with the vector-field in order to provide a mesh alignment capability [6]. The present author has observed that the matrices involved in the variational principle of Giannakopoulos and Engle are singular which suggests that the corresponding grid generation equations are, at best, degenerate elliptic [9, p. 224]. If true, non-smooth grids may be expected. This expectation is supported by the subsequent effort of Brackbill to improve the regularity of the Giannakopoulos and Engle functional [3].

The present method (presently referred to as the vector-field adaptive or VFA method) also uses the idea of alignment with given vector fields, but does so indirectly via control of the inverse Jacobian of the transformation. The method is based on a variational principle which is a weighted form of Brackbill and Saltzman's "smoothness" principle. Consequently, the grid generation equations are a weighted form of the Winslow equations and preserve ellipticity. The method has the attraction of being easily incorporated as a modification to existing elliptic mesh generation codes.

It is desirable to minimize the number and complexity of the control functions/arbitrary parameters used in any grid generator. The experience of those working in the area suggests that the use of arbitrary parameters in grid generation may be unavoidable due to the complexity of the geometries involved and the requirement of adaptivity. On the other hand, if at least the control functions can be rigorously derived as part of the theoretical development of the grid generator, then the arbitrary parameters can likely be used with more confidence. The main purpose of the present method is to show that previous weighted

[1] By and large these methods have proved effective in controlling the mesh; however, the process generally lacks automation. The lack of automation is a significant limitation of current algorithms. The present paper does not purport to address this issue.

0021-9991/95 $6.00

forms of the Winslow generator may be replaced by a less ad hoc generator in which the weights have a clear interpretation as the elements of the inverse Jacobian matrix of the mapping. In the bargain one gets a general alignment capability, provided one has access to or can construct at least one vector-field on the domain.

2. THE VECTOR-FIELD ADAPTIVE METHOD

In practice constructing vector-fields on an arbitrary domain may be a difficult, if not formidable, task. The most natural application of this method would be in the case of *r*-type adaptive mesh generation for parabolic problems. In this case, an initial mesh would be used to start the calculation. As the solution evolves in time the flux-field may be used as one of the vector fields. The other vectors may then be constructed from the first by taking orthogonal complements (a trivial task in two dimensions). In this section it is assumed that the three vector-fields from which the weights are to be constructed are known. The method is presented for the three-dimensional case, with the restriction to two dimensions following naturally.

2.1. *The 3D Vector-Field*

Let a simply-connected domain $\Omega \subset R^3$ be given. On this domain three vector-fields $\mathbf{V}_1(\mathbf{x})$, $\mathbf{V}_2(\mathbf{x})$, and $\mathbf{V}_3(\mathbf{x})$ with components V_i^k, $i, k = 1, 2, 3$ are given. Since non-folded meshes are wanted, it is assumed that the vector-field triplet is oriented such that $\mathbf{V}_1 \cdot (\mathbf{V}_2 \times \mathbf{V}_3) > 0$ on Ω. In general, the boundary of Ω need not be aligned with one of the vector-fields, although this would make alignment of the interior grid much easier to achieve.

Define the unit vectors $\mathbf{U}_i = \mathbf{V}_i / |\mathbf{V}_i|$ and the matrix $\mathcal{U}$ with elements $\mathcal{U}_{i,k} = U_k^i$. The determinant μ of $\mathcal{U}$ is positive.

Let $U = \{(\xi, \eta, \zeta) | 0 < \xi, \eta, \zeta < 1\}$ be the logical space. The mapping $\mathbf{x}(\xi, \eta, \zeta)$ from logical to physical space is determined by minimizing the weighted variational principle to be described shortly. The weight is constructed by requiring the tangents to the mapping to be **aligned** with the vector-fields,[2] i.e., $\mathbf{x}_{\xi_i} = \ell_i \mathbf{U}_i$ with the ℓ_i being positive scale factors. This requirement can be expressed as a condition on the Jacobian matrix,

$$\mathcal{J} = \mathcal{U}\mathcal{L} \tag{1}$$

where $\mathcal{L} = \text{diag}(\ell_1, \ell_2, \ell_3)$. Let $\lambda = \ell_1\ell_2\ell_3$ be the determinant of $\mathcal{L}$. From (1) the cell volume is $\sqrt{g} = \mu\lambda$. For brevity, let $\mathcal{T}(\mathbf{x}) = \mathcal{U}\mathcal{L}$ be the weight matrix.

To develop a variational principle it is best to state the alignment condition in terms of the contra-variant tangents. This is because $\mathcal{T}$ is a **physical-space** weight function and the Euler–Lagrange equations are easier to derive in terms of a contra-variant-based principle when a physical space weight is used.

Inverting (1) gives $\mathcal{J}^{-1} = \mathcal{S}$, where $\mathcal{S} = \mathcal{T}^{-1}$. The contra-variant variational principle is to be minimized over the set of admissible functions satisfying the boundary data,

$$I[\nabla_x\xi, \nabla_x\eta, \nabla\zeta] = \int_\Omega G\, dx\, dy\, dz, \tag{2}$$

where

$$G = |\mathcal{J}^{-1} - \mathcal{S}|^2. \tag{3}$$

The principle is thus a least-squares fit of the inverse Jacobian matrix to the alignment condition (1). The principle is a weighted generalization of the "smoothness" variational principle [2].

The second-variation of the functional can be expressed in terms of the 4×4 Hessian matrix $\mathcal{H}$ (see [9, p. 176]). For the G in (3), the Hessian matrix is simply $\mathcal{H} = \text{diag}(1, 1, 1, 1)$. The functional therefore satisfies the necessary condition for convexity.

The Euler–Lagrange equation is of primary interest since it gives the grid generation equations:

$$\text{div}_x(\mathcal{J}^{-1} - \mathcal{S}) = \mathbf{0}. \tag{4}$$

This clearly has the solution $\mathcal{J}^{-1} = \mathcal{S}$, boundary conditions permitting. Thus the alignment condition (1) is potentially a solution to the grid generation equations.

In two-dimensions, Eq. (4) is recognized as the pair of Poisson equations,

$$\nabla_x^2\xi = \frac{\partial S_{11}}{\partial x} + \frac{\partial S_{12}}{\partial y}, \tag{5}$$

$$\nabla_x^2\eta = \frac{\partial S_{21}}{\partial x} + \frac{\partial S_{22}}{\partial y}, \tag{6}$$

where S_{ij} are the elements of $\mathcal{S}$.

Inverting (4) with $\mathcal{S} = 0$ results in the well-known equations $\mathcal{Q}_w\mathbf{x} = \mathbf{0}$, where

$$\mathcal{Q}_w\mathbf{x} = g_{22}\mathbf{x}_{\xi\xi} - 2g_{12}\mathbf{x}_{\xi\eta} + g_{11}\mathbf{x}_{\eta\eta}. \tag{7}$$

A six-line derivation of these equation is given in [9, p. 154]; the derivation may be generalized to the case $\mathcal{S} \neq 0$ to get the weighted inverted equations:

$$\mathcal{Q}_w\mathbf{x} = -\sqrt{g}\, \mathcal{J}[\nabla_\xi\mathcal{S}]\mathcal{C}, \tag{8}$$

with $\mathcal{C} \equiv \sqrt{g}\, \mathcal{J}^{-T}$.

[2] This, of course, assumes that the designer of the mesh knows *à priori* which of the three tangents can be best aligned with $\mathbf{U}_1$, which with $\mathbf{U}_2$, and which with $\mathbf{U}_3$. In complicated geometries, it may be necessary to change the correspondance in different parts of the domain.

Explicitly, the equations in two dimensions are a weighted form of Winslow's grid generation equations:

$$g_{22}\mathbf{x}_{\xi\xi} - 2g_{12}\mathbf{x}_{\xi\eta} + g_{11}\mathbf{x}_{\eta\eta} = -\sqrt{g}\,\mathcal{J}\mathbf{W} \tag{9}$$

with

$$\mathbf{W} = \begin{Bmatrix} (S_{11})_\xi y_\eta - (S_{12})_\xi x_\eta - (S_{11})_\eta y_\xi + (S_{12})_\eta x_\xi \\ (S_{21})_\xi y_\eta - (S_{22})_\xi x_\eta - (S_{21})_\eta y_\xi + (S_{22})_\eta x_\xi \end{Bmatrix}. \tag{10}$$

That $\mathcal{J}^{-1} = \mathcal{S}$ is a solution to (4) is clear. It is perhaps not as clear that it is also a solution to the inverted equations (8), but this may be verified using the identity

$$\mathcal{Q}_w\mathbf{x} \equiv -\sqrt{g}\,\mathcal{J}[\nabla_\xi \mathcal{J}^{-1}]\mathcal{C} \tag{11}$$

(the latter is easily derived using the methods in [9, pp. 143–154]). As a consequence, any smooth mesh on Ω can be produced by this grid generator by using the proper boundary conditions and by using a weight $\mathcal{S}$ constructed by computing the Jacobian matrix of the given mesh and inverting it.[3] It should be clear from this discussion that it would be easy to construct a simple example in which vector-fields derived from a conformal map would generate a mesh that exactly satisfied the alignment condition (1). It is more interesting to examine the performance of the method on a general problem. Before doing so, it is necessary to consider the as-yet undetermined elements of the matrix $\mathcal{L}$.

2.2. *The Scale Factors as Control Functions*

Since (1) implies that $\ell_i = \sqrt{g_{ii}}$, the scale factors control the lengths of the sides of the mesh cells. These lengths are almost completely arbitrary, being only weakly constrained by the requirement that determinant of $\mathcal{L}$ be related to the volume ρ of the domain as

$$\rho = \int_U \mu\lambda\, d\xi\, d\eta\, d\zeta. \tag{12}$$

In practice, however, the scale factors need to be selected carefully so that cell sizes are not excessively large or small. Each ℓ_i may vary spatially, but they should be roughly of the same order of magnitude as the length of the domain Ω in the given direction.

One approach to constructing the scale factors begins by defining average lengths in each direction:

$$L_{ii} = \int_U \sqrt{g_{ii}}\, d\xi\, d\eta\, d\zeta. \tag{13}$$

[3] The question of how to perform the various discretizations involved in order that the constructed $\mathcal{S}$ exactly recovers the original mesh is interesting, but it is tangential to the objectives of this paper.

In practice, only rough estimates of these average lengths need be made, so the integral given above can be either estimated or computed with low accuracy.

Define the ratio $r_i = |\mathbf{V}_i|/|\overline{\mathbf{V}}_i|$, where the average velocity is

$$|\overline{\mathbf{V}}_i| = \frac{1}{\rho}\int_\Omega |\mathbf{V}_i|\, dx\, dy\, dz. \tag{14}$$

Let $f_i \mid R^+ \to R^+$ be a positive real-valued function of r_i such that $f_i' \le 0, \bar{f}_i \le f_i(0) < \infty$, and $f_i(1) = \bar{f}_i$. Let $\bar{f}_i$ be the average value of f_i over the logical domain:

$$\bar{f}_i = \int_U f_i(r_i)\, d\xi\, d\eta\, d\zeta. \tag{15}$$

The quantities $\bar{f}_i$ and $|\overline{\mathbf{V}}_i|$ may be computed with low accuracy. The following form for the scale factors was selected:

$$\ell_i = L_{ii}\frac{f_i}{\bar{f}_i}. \tag{16}$$

For $r_i > 1$ this construction gives smaller than average lengths while $r_i < 1$ gives larger than average lengths. In general it is difficult to satisfy the requirement (15), but approximate equality is all that is needed. For μ approximately equal to one, this construction can be shown to roughly satisfy (12).

Arbitrariness in the method is thus mainly in the selection of the functional form of the f_i. The numerical examples to follow suggest that these control functions can be simple functions with few parameters.

3. NUMERICAL RESULTS

The first example is provided by the flow-field $\mathbf{V}$ in Fig. 1 which was computed using the SECO2D groundwater flow code with a uniform mesh [11]. The white areas represent near-stagnation regions in the flow. Note that the boundaries of the domain do not coincide with streamlines, so the method cannot hope to achieve perfect alignment with this field. Nevertheless, there is a general tendency for the streamlines to be vertical, meandering from the top to the bottom boundary. This suggests that one assign the vector fields as $\mathbf{V}_2 = -\mathbf{V}$ and $\mathbf{V}_1 = \mathbf{V}^\perp$ so that the tangents $\mathbf{x}_\xi$ and $\mathbf{x}_\eta$ are aligned with $\mathbf{V}^\perp$ and $-\mathbf{V}$, respectively. The matrix $\mathcal{U}$ is then completely defined. The scale factors ℓ_i must also be defined. The simplest possibility is to set the scale factors to the lengths of the domain ($\ell_1 = L_{11}$ and $\ell_2 = L_{22}$). This case adapts the mesh to the directions implied by the vector field and specifies orthogonality and uniform cell lengths.

A 40 × 46 mesh was computed using the weight $\mathcal{S}$ constructed from this flow-field and the grid generator (8). Discrete velocity data was computed by SECO2D on a 46 × 53 uniform grid; velocities for the weight matrix $\mathcal{S}$ were obtained from this data using a bilinear interpolation scheme. The initial boundary

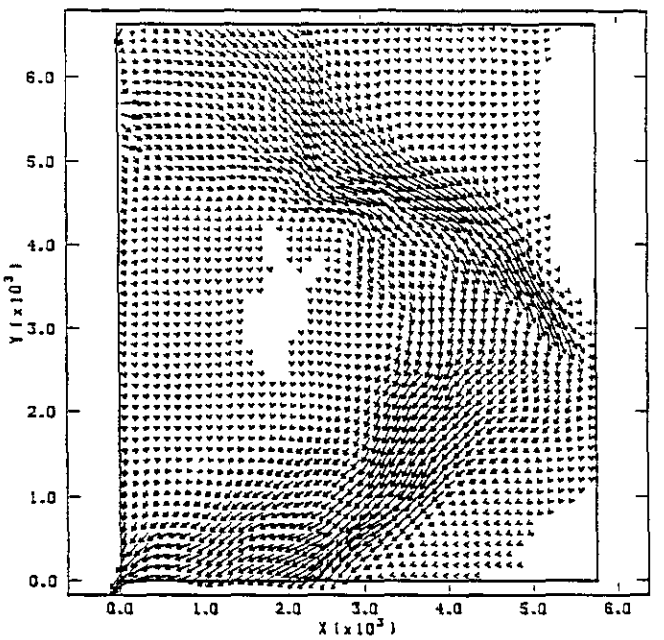

FIG. 1. SECO2D flow-field.

parameterization consists of a stretch along the top and bottom boundaries to roughly match locations where the velocities are large.

The result is given in Fig. 2; it is seen that the mesh has deviated only a little from the initial stretched tensor product mesh. The improvement in alignment is rather poor (the root-mean-square deviation from the desired alignment direction is 21.2°, compared to 22.4° for the initial tensor product mesh). The problem is not the grid generation algorithm, but the boundary—alignment cannot be much improved because the flow is not well aligned with the boundary near the top and bottom portions of the main flow.

To further adapt the mesh to the flow-field it makes sense to require that the horizontal cell-spacing be small when the magnitude of the flow is large. The function

$$f_1(r_1) = \bar{f}_i \exp \alpha(1 - r_1), \tag{17}$$

with α a positive parameter, qualitatively produces this behavior. Note that both the f_i satisfy the constraints required in the previous section (provided α is approximately one-half). This weighting and vector field give

$$|\overline{\mathbf{V}}| = 2.236 \text{ m/s} \tag{18}$$

$$\mathbf{V}_M = 10.66 \text{ m/s} \tag{19}$$

$$\bar{f}_1 = 1.02. \tag{20}$$

Figure 3 gives the resulting mesh (with $\alpha = 0.4$). In general, the grid echoes the properties of the flow-field: relatively large cells in the three stagnation zones (upper and lower right, center) and skinny cells in the main flow-path. The mesh is smooth and the cells not badly skewed. Because the cell sizes are now

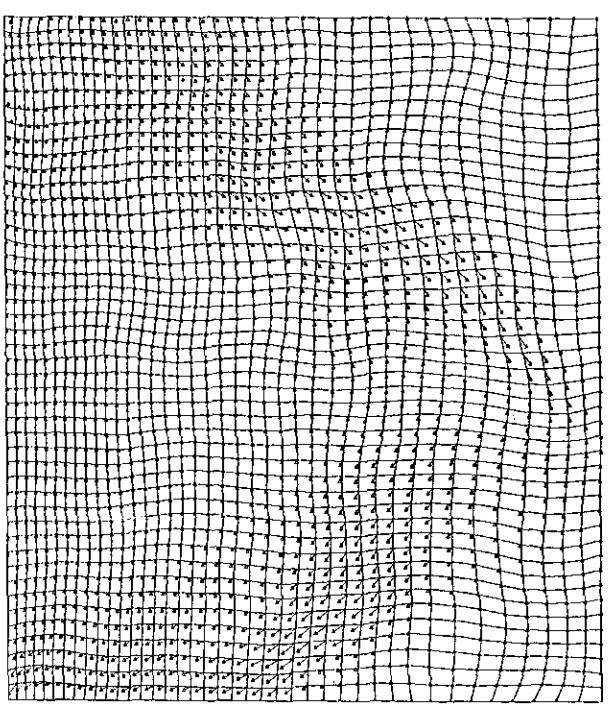

FIG. 2. Grid adapted to SECO2D flow-field: Uniform length control.

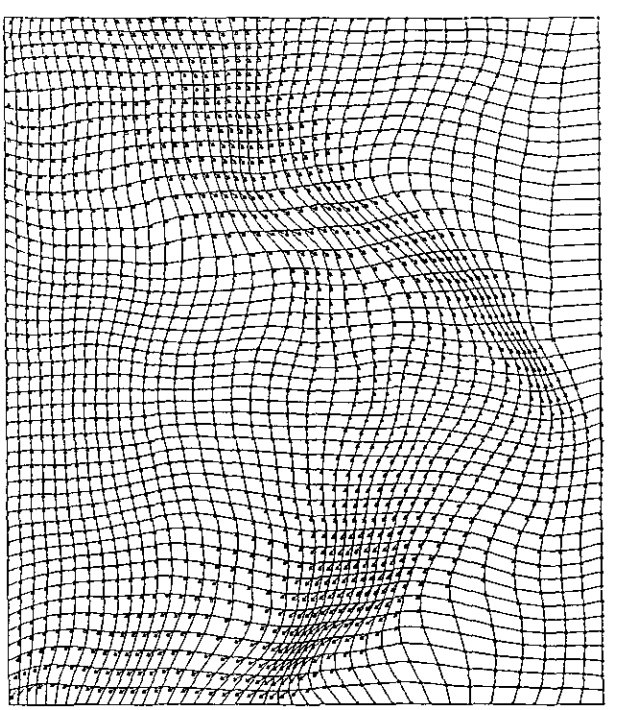

FIG. 3. Grid adapted to SECO2D flow-field: Non-uniform length control.

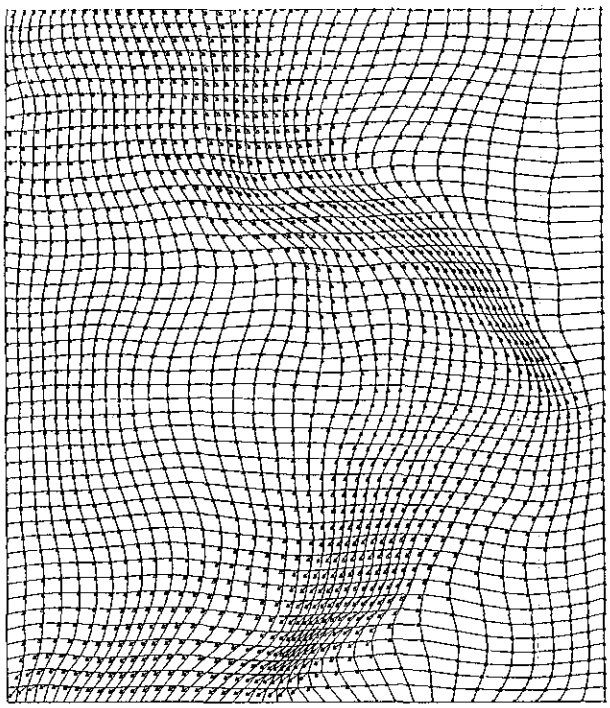

FIG. 4. Grid adapted to SECO2D flow-field: Non-orthogonal vector field.

adapted to the magnitude of the velocity, the mesh is certainly better adapted to the flow-field than the uniform cartesian mesh with which the flow-field was originally computed. Grid lines are somewhat better aligned with the vector-field, particularly in the middle right portion of the domain (now the root-mean-square deviation for the flow direction is 19.1°). Due to the constraints of the boundary data, the mesh is unable to completely align with the flow-field. Alignment can be increased somewhat by increasing the parameter α, but at the price of overly large cells appearing in the stagnation region at the upper right. For large enough values of α the method will produce a folded mesh (this behavior is common to all forms of the inhomogeneous Winslow equations). The functional form of the control function is apparently of little importance. For example,

$$f_1 = \begin{cases} 2, & r_1 \le \frac{1}{4}, \\ 1/\sqrt{r_1}, & \text{otherwise}, \end{cases} \tag{21}$$

gives a mesh similar to that in Fig. 3.

The fact that this method can accommodate non-orthogonal vector-field pairs is illustrated next. It seems reasonable to expect that alignment would be more achievable if one relaxed the requirement of orthogonality. With this goal in mind, redefine $\mathbf{V}_1$ by the parameter weighted form

$$\mathbf{V}_1 = (1 - \mu)\mathbf{V}^{\perp} + \mu(1, 0)^{\mathsf{T}}. \tag{22}$$

The idea is that the tangent $\mathbf{x}_\eta$ will still try to align with $\mathbf{V}_2 = -\mathbf{V}$, but that the tangent $\mathbf{x}_\xi$ may now align with the unit vector (1, 0) to match the boundary data better (the value $\mu = 1$ gives this case while $\mu = 0$ gives the original case). Unfortunately, no solution was obtained for $\mu = 1$ because some of the node points moved outside the domain during the initial portion of the iteration. The iteration is forced to halt in this case because there is no velocity data available outside the domain. It is not clear whether or not the iteration would have converged had such data been available. A solution was obtained for $\mu = 0.5$ (Fig. 4). The plotted result is almost indistinguishable from that in Fig. 3, but at least it shows that it is possible to use non-orthogonal vector-field pairs. The root-mean-square angle deviation corresponding to Fig. 4 is 18.5°—a slight improvement on Fig. 3.

Alignment can be improved considerably by changing the boundary parameterization. Figure 5 shows a case in which the boundary parameterization (fixed throughout the calculation) was modified so that the corners of the logical domain map to the points shown in physical space. Alignment with the flow-fields is improved compared to the result in Fig. 4 (in Fig. 5 the root-mean-square angle deviation is 14.8°). Although the cells near the corner of the domain in Fig. 5 are probably not acceptable for calculation of the flow-field, the robustness of the method is demonstrated. The example suggests that when domain boundaries align reasonably well with the flow-field the interior alignment may be fairly good. For comparison, the grid generated by the Laplace equations (no weights) for this

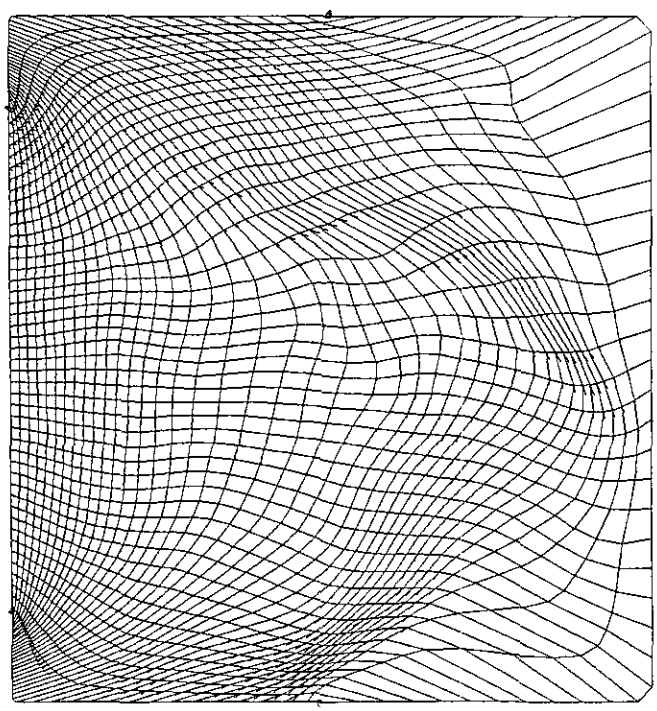

FIG. 5. Adaption with alternate boundary parameterization.

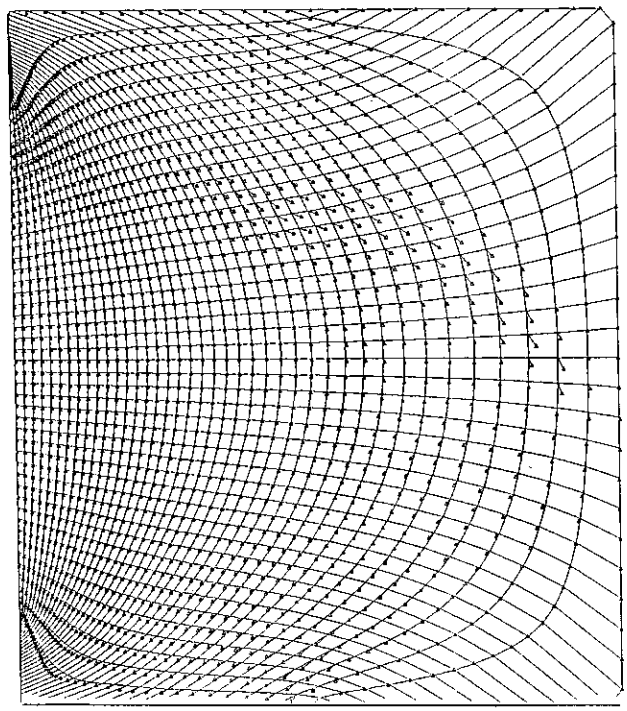

FIG. 6. Laplace grid for alternate boundary parameterization.

domain is shown in Fig. 6. The Laplace grid seems not nearly as well adapted to the flow even though the root-mean-square angle deviation is 15.9°.

Figure 7 shows the method applied to a miscible fluid flow problem. The mesh has little trouble adapting to the flow-field direction in this case because the flow is much better aligned with the boundary of the domain.

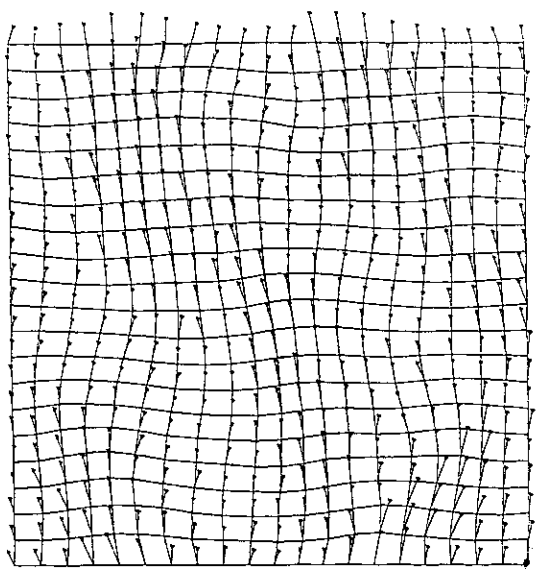

FIG. 7. Grid adapted to miscible displacement flow-field.

4. SUMMARY

A variational theory for a weighted elliptic grid generator that controls the inverse Jacobian of the mapping was presented. The weights are constructed from sets of vector-fields in such a manner that mesh tangents align with the vector-fields in a least-squares sense. Arbitrary scale factors control cell lengths. The method was demonstrated on two flow-fields. Although more tests are needed to definitively establish the strengths and weaknesses of the method, these preliminary results suggest that adapting meshes to the given vector-fields is possible. The method will be particularly effective when the boundaries of the domain coincide with the vector-fields in some way. One drawback of the present method is the arbitrariness of the scale factors. It is hoped that this work will motivate others to make further tests of the method, resulting in better length control functions.

It is tempting to use alignment as the measure of grid quality in the examples given, but this is unfair because the meshes are, in fact, attempting more than just alignment. Recall that a length control function was introduced to create small cell lengths in places where the magnitude of the velocity is large. In addition, the weight construction requested that the grid be orthogonal. The ideal norm for determining whether or not the grids shown are any good is precisely the value of the functional I in Eq. (2)—this measure takes into account both alignment and length control goals. To use the L^2 or maximum norms on just the deviation from alignment changes the criterion by which the meshes were generated.

Because the functional performs a least-squares fit of the Jacobian matrix to the weight matrix, it is the user himself who decides what the ultimate good grid is—it is the grid that satisfies $\mathcal{J}^{-1} = \mathcal{S}$ everywhere, i.e., the one that makes $I = 0$. If the user selects a different weight matrix, then he is, in effect, saying that the best grid is now something different (for better or worse, the burden is thus put entirely upon the user). Of course, in realistic applications one will rarely attain $I = 0$ so the target "good" grid (as defined by the user) is rarely attained. Since the grid generated always minimizes the functional I, the grid is a best-fit to the ultimate "good" grid as defined by the user—that is the strength of the variational method.

The examples based on the SECO flow-field show only modest improvement in alignment compared to what one achieves with transfinite or Laplace grids. This is because it has not been possible to match the Jacobian to the weight $\mathcal{S}$ which was specified. The main culprit appears to be the boundary data. As noted at the end of Section 2.1, $\mathcal{J}^{-1} = \mathcal{S}$ is a solution to the Euler–Lagrange equations, so one should attain perfect alignment with this method *were there no boundary data to be satisfied* (and were the weight constructed from the gradient of a vector potential). Unfortunately, the SECO example does not illustrate this point well. Nevertheless, the examples show that the VFA grids are better adapted to the flow-field than

grids constructed by transfinite interpolation or the unweighted Laplace generator.

No example is given for the case of three dimensions because a realistic problem was not readily accessible to this investigator. Although the case of three dimensions is covered by the theory presented, it remains to be seen how well it works in practice. The answer probably depends in part on how much work the practioner is willing to do in devising the weight and, in part, on how well the problem lends itself to the overall goal of alignment of the tangents in the interior with the vector-fields, given the constraint of the boundary data.

A large number of extensions of this method are possible. If one has a good base mesh which was constructed in some other way, the weight $\mathcal{S}$ can be computed for that mesh and weighted with any vector-field one wishes to adapt to (this is similar to the approach used in [10]). One might weight the matrix $\mathcal{S}$ constructed from a vector-field with one derived from the boundary tangents, so that near the boundary the grid aligns with the boundary instead of the flow-field. Another intriguing line of investigation is to determine whether or not it is possible to automatically adjust the points on the boundary to maximize the degree of alignment.

Since the method performs a least-squares fit to the inverse Jacobian matrix it seems clear that, in principle, the user may control any of the standard metric properties of the grid (e.g., area, length, orthogonality) by proper construction of the weight. If a vector-field is not available, one can be constructed from a reasonable initial mesh (say from an algebraic generator). The method is particularly applicable to r-type adaptive mesh calculations in which time-varying flux fields may dictate regions of the domain in which mesh refinement and adaptation is needed.

ACKNOWLEDGMENTS

The author thanks Jerry Brackbill, David Dean, Wojciech Golik, Tom Russell, and the reviewers.

REFERENCES

1. D. A. Anderson, *Appl. Math. Comput.* **24,** 211 (1987).
2. J. U. Brackbill and J. S. Saltzman, *J. Comput. Phys.* **46,** 342 (1982).
3. J. U. Brackbill, *J. Comput. Phys.* **108,** 38 (1993).
4. Y. C. Chao and S. S. Liu, *Numer. Heat Transfer, Part B* **20,** 145 (1991).
5. P. R. Eiseman, *Comput. Math. Appl.* **24,** 57 (1992).
6. A. E. Giannakopoulos and A. J. Engel, *J. Comput. Phys.* **74,** 422 (1988).
7. O.-P. Jacquotte, *Comput. Methods Appl. Mech. Eng.* **66,** 323 (1988).
8. Y. N. Jeng and Y. C. Liou, *Numer. Heat Transfer, Part B,* **23,** 135 (1993).
9. P. M. Knupp and S. Steinberg, *The Fundamentals of Grid Generation* (CRC, Boca Raton, FL, 1993).
10. P. J. Roache, K. Salari, and S. Steinberg, *Commun. Appl. Numer. Methods* **7,** 345 (1991).
11. P. J. Roache, P. M. Knupp, S. Steinberg, and R. Blaine, in *Benchmark Test Cases for Computational Fluid Dynamics,* edited by I. Celik and C. J. Freitas, (ASME FED Vol. 93, Book No. H00598, 1990), p. 49.
12. G. Ryskin and L. G. Leal, *J. Comput. Phys.* **50,** 71 (1983).
13. G. R. Shubin and J. B. Bell, *Comput. Methods Appl. Mech. Eng.* **47,** 47 (1984).
14. J. L. Steger and D. S. Chausee, *SIAM J. Sci. Stat. Comput.* **1,** 431 (1980).
15. S. Steinberg and P. J. Roache, *Numer. Methods PDEs* **2,** 71 (1986).
16. J. F. Thompson, F. C. Thames, and C. W. Mastin, *J. Comput. Phys.* **15,** 299 (1974).
17. Z. U. A. Warsi, in *Numerical Grid Generation,* edited by J. F. Thompson (North-Holland, New York, 1982), p. 41.
18. A. Winslow, *J. Comput. Phys.* **2,** 149 (1967).