JOURNAL OF COMPUTATIONAL PHYSICS 100, 163–178 (1992)

Variational Curve and Surface Grid Generation*

STANLY STEINBERG

Department of Mathematics and Statistics, University of New Mexico, Albuquerque, New Mexico 87131

AND

PATRICK ROACHE

Ecodynamics Research Associates, Inc., P.O. Box 8172, Albuquerque, New Mexico 87198

Received July 26, 1989; revised August 21, 1991

Variational algorithms that control the lengths of grid lines, cell areas, and the orthogonality of grid lines can be used for generating boundary-conforming grids on surfaces. Additional geometric control is provided by using a reference grid, while solution adaptivity is achieved by using weights. In a typical application, the reference grid can be used to produce an exponential compression of the grid at a boundary, while the solution adaptive weights are used to make the grid spacing inversely proportional to the gradient (when the gradient is large) of some solution being computed on the grid. The grid is adapted on both the interior and boundary of the surface. The algorithm performs these tasks with exceptional precision, as demonstrated in the examples presented here.

CONTENTS

1. INTRODUCTION

In the paper [9], the authors present a new variational algorithm for generating grids on surfaces. The paper emphasizes the development of algorithms based on direct geometric intuition. Functionals for controlling the length of grid lines, areas of grid cells, and angles between grid lines are introduced, as well as reference grids which are used to highly compress the grids along boundaries (see also [2, 3]). Solution adaptive weights are not discussed, but can be easily added to the theory (as is done in this paper).

In some applications, the surface grid generator has convergence difficulties for surfaces of modest shape. These difficulties are, in fact, a result of a bifurcation of the solution of the discrete grid-generation equations [11]. The results described here show how to formulate the discrete approximations to the Euler–Lagrange equations for the variational problem so that the bifurcation is either eliminated or significantly delayed, and thus grids can be generated on most surfaces of interest. The main idea is to keep the Euler–Lagrange equations symmetric. Because these equations are nonlinear, there are certain subtleties involved, particularly for the area functional.

For several years, the authors have been using numerical geometric quality measures to judge the performance of numerical grid-generation algorithms. A description of these measures has not been published; therefore, some information is included here. The measures are far more critical than graphical displays, so only a few plots are shown in the paper. (The plots look exactly as one would imagine.)

The main points of this paper are: (1) simple geometric intuition is used to create a variational problem for generating grids with certain desired properties; (2) appropriate discretization of the Euler–Lagrange equations for the variational problem produces algorithms adequate for generating grids; (3) grid quality measures should be used to judge the success of the algorithm; (4) and the generated grids satisfy the desired intuitive criteria. In fact, the algorithms match intuition exceptionally well and this makes them relatively easy to use.

A related variational algorithm is introduced by Saltzman [8] and a differential-geometric approach is considered by Warsi [14]. Grid generation on curves

* This work was partially supported by the Office of Naval Research and the Army Research Office.

581/100/1-12

0021-9991/92 $5.00

is discussed by Eiseman [5]. Over 30 grid-generation algorithms have been considered by Knupp [6]; the most promising are being extended to surface-grid generators. Also, Arina [1] discusses adaptive orthogonal coordinates.

Most of the algorithms that are important for surface-grid generation also apply to curve-grid generation. The curve problems are simpler to analyze, implement, and study numerically. Consequently, this paper begins with a discussion of curves and then extends these results to surfaces. Moreover, planar regions are a special case of surfaces, so the surface algorithm is specialized to this case. This special algorithm eliminates the need to parameterize the full region and is more efficient than the general surface algorithm.

Previously, two variational curve grid-generation algorithms have been studied by the authors [11]. These algorithms are derived by first solving the Euler–Lagrange equation for the second derivative term and then discretizing the resulting equation (see Eq. (2.37) and (2.45) of [11]). When the equation is solved for the second derivative term, a first derivative term is created and this then results in an asymmetry which causes serious problems. In this paper, the Euler–Lagrange equation is left in symmetric form and is differenced in such a way that the resulting discrete equations are symmetric when Q and W are constant (see Eq. (3.33) below). This results in a substantial improvement in the algorithm. In addition, reference grids and solution adaptivity are consider in this paper but were not considered in [11].

This paper is organized as follows: first, the quality measures and symmetric differencing are described. Second, curve grid generation is presented, and a solution adaptive weight is introduced. This is followed by a set of numerical experiments, which show that the grid-bifurcation problem is less significant and that the algorithm does reference and solution adaption as predicted by the theory. In fact, the algorithm produces grids of exceptional quality; however, for certain solution adaptive weights convergence is delayed. Changes in the algorithm to reduce this problem will be considered in the future. Third, surface grid generation is described and numerically tested. Only a few tests are needed to confirm that the surface results are consistent with the curve case. The reduction of the surface algorithm to a planar algorithm and the dependence of grids on the parameterization are discussed in Section 7.

2. PRELIMINARIES

Two general ideas are critical to this work. First, the numerical algorithms perform better if the main parts of the differential equations that are used to compute the grids are kept in symmetric form, and this symmetry is preserved in the difference equations. Moreover, the quality of the generated grids is best judged by using numerical measures closely related to the variational principals that are used to generate the grids.

2.1. *Quality Measures*

The geometric properties of the grids are measured using averages and deviations. As discussed below, the variational methods provide algorithms with direct geometric intuition; how well the intuitive requirements are met is estimated using these measures.

The curve grid generator attempts to divide a curve into n segments, where the length of the grid segments L_l are required to be proportional to some given quantities a_l, that is, $L_l \propto a_l$, $1 \leqslant l \leqslant n$. In this case, an average value is defined by

$$\text{aver} = \frac{1}{n} \sum_{l=1}^{n} \frac{L_l}{a_l}. \tag{2.1}$$

Then the deviation from the average is defined by

$$\text{dev} = \frac{\sqrt{(1/n) \sum_{l=1}^{n} ((L_l/a_l) - \text{aver})^2}}{\text{aver}}. \tag{2.2}$$

In the case of surfaces, two length deviations are computed. One is for segments in the first logical direction, and another is for the second logical direction. The code reports the average of the two deviations.

Like the length control, the area control is supposed to produce grids where the area of each cell A_l is proportional to some given quantities b_l. So again, if there are m cells, an average value is defined by

$$\text{aver} = \frac{1}{m} \sum_{l=1}^{m} \frac{A_l}{b_l} \tag{2.3}$$

and the deviation is defined as it is in the length case. The area of a cell is measured by the magnitude of the cross product of two edge vectors for a pair of adjoining edges.

If the deviation is normalized by the average, then $100 * \text{dev}$ represents a percentage deviation or error. It is easy to see that, if the deviations are all approximately the same as the average, then the deviation is about 1 or 100%. This is not good and means that the algorithm is failing to generate a reasonable grid. If the deviation is about 0.1 or 10%, then the method has a significant effect. However, defects in the generated grid can be seen graphically. If the deviation is about 0.01, then the intuitive requirements are well satisfied and the grid defects are not noticeable graphically. Experience indicates that grids with a quality measure less than 0.3 are adequate.

The orthogonality control is supposed to produce orthogonal grids. If the grid is nearly orthogonal, then the cosines of the angles between the grid lines are

approximately zero and, consequently, the average of the cosines should be close to zero. Thus there is no need to calculate an average value; only a deviation is needed. Because the average value nearly is zero, it is not reasonable to normalize the deviation by the average. If θ_l, $1 \leqslant l \leqslant m$, are the angles between the grid lines, then the deviation of the cosines of the angles is defined by

$$\text{dev} = \sqrt{\frac{1}{m} \sum_{l=1}^{m} \cos^2(\theta_l)}. \tag{2.4}$$

The code reports arc cos(dev) in degrees.

Note that quality measures, for grids that are good, are computed by subtracting nearly equal numbers. On single precision machines, this leads to significant relative errors, so only the order of magnitude of the quality measures is considered significant.

2.2. *Symmetric Differencing*

As shown below, it is possible to write the principal part of the Euler–Lagrange equations for the grid-control functionals in a symmetric form. The notion of symmetry corresponds to the concept of formally self-adjoint operators when the operators are linear. Because the operators that appear in the grid-generation problem are nonlinear, the symmetry conditions are written explicitly below. Numerical experimentation shows that discretizing the symmetric differential equations so that the resulting difference equations are also symmetric substantially improves the numerical algorithms. The symmetry conditions written explicitly below correspond to the coefficient matrix of the difference scheme being symmetric. See [12] for more details on symmetric differencing.

Differential equations are written in the logical variables (ξ, η), so these variables are used to describe the symmetric differences. Define the half-step central-difference and central-average operators by

$$\delta_\xi f(\xi, \eta) = \frac{f(\xi + \Delta\xi/2, \eta) - f(\xi - \Delta\xi/2, \eta)}{\Delta\xi}, \tag{2.5}$$

$$\delta_\eta f(\xi, \eta) = \frac{f(\xi, \eta + \Delta\eta/2) - f(\xi, \eta - \Delta\eta/2)}{\Delta\eta}, \tag{2.6}$$

$$\mu_\xi f(\xi, \eta) = \frac{f(\xi + \Delta\xi/2, \eta) + f(\xi - \Delta\xi/2, \eta)}{2}, \tag{2.7}$$

$$\mu_\eta f(\xi, \eta) = \frac{f(\xi, \eta + \Delta\eta/2) + f(\xi, \eta - \Delta\eta/2)}{2}. \tag{2.8}$$

Note that the full-step central-difference operator can be written in terms of the half-step difference and average,

$$\delta_\xi \mu_\xi f(\xi, \eta) = \frac{f(\xi + \Delta\xi, \eta) - f(\xi - \Delta\xi, \eta)}{2\,\Delta\xi}. \tag{2.9}$$

Most terms in the grid generation equations can be differenced using the following rules:

$$\frac{\partial}{\partial\xi}\left(a(\xi, \eta)\frac{\partial f(\xi, \eta)}{\partial\xi}\right) \approx \delta_\xi((\mu_\xi a(\xi, \eta))(\delta_\xi f(\xi, \eta))), \tag{2.10}$$

$$\frac{\partial}{\partial\xi}\left(a(\xi, \eta)\frac{\partial f(\xi, \eta)}{\partial\eta}\right) \approx \delta_\xi \mu_\xi(a(\xi, \eta)(\delta_\eta \mu_\eta f(\xi, \eta))). \tag{2.11}$$

The mixed-derivative term is not symmetric. However, the combination

$$\frac{\partial}{\partial\xi}\left(a(\xi, \eta)\frac{\partial f(\xi, \eta)}{\partial\eta}\right) + \frac{\partial}{\partial\eta}\left(a(\xi, \eta)\frac{\partial f(\xi, \eta)}{\partial\xi}\right) \tag{2.12}$$

is symmetric; the second rule applied twice to this expression produces a symmetric difference expression.

The area-control differential equations contain unusual terms; for example, the above rules do not apply to

$$\frac{\partial}{\partial\xi}\left(a(\xi, \eta)\frac{\partial g(\xi, \eta)}{\partial\xi}\frac{\partial g(\xi, \eta)}{\partial\eta}\frac{\partial f(\xi, \eta)}{\partial\eta}\right). \tag{2.13}$$

Again, such terms occur in pairs, with the ξ and η derivatives interchanged. Such combinations are differenced to produce symmetric schemes using the rule:

$$\frac{\partial}{\partial\xi}\left(a(\xi, \eta)\frac{\partial g(\xi, \eta)}{\partial\xi}\frac{\partial g(\xi, \eta)}{\partial\eta}\frac{\partial f(\xi, \eta)}{\partial\eta}\right) \approx \mu_\eta\,\delta_\xi((\mu_\xi \mu_\eta a(\xi, \eta))(\mu_\eta\,\delta_\xi\, g(\xi, \eta)) \times (\mu_\xi\,\delta_\eta\, g(\xi, \eta))(\mu_\xi\,\delta_\eta f(\xi, \eta))). \tag{2.14}$$

This rule is derived from techniques which are explained in [12]. The above rule does not immediately produce a nearest neighbor scheme, but a scheme can be generated by replacing $\Delta\xi$ by $\Delta\xi/2$ and $\Delta\eta$ by $\Delta\eta/2$.

2.3. *Symmetric Difference Equations*

The above rules are used to produce difference equations by replacing the $\Delta\xi$ notation by an index notation. In one dimension, the stencil (i.e., coefficients of the difference scheme) names are l, c, and r with the meanings "left," "center," and "right." The equations that are solved have the form

$$l_i f_{i-1} + c_i f_i + r_i f_{i+1} = g_i, \qquad 2 \leqslant i \leqslant n-1. \tag{2.15}$$

Here l_i, c_i, r_i, and g_i are given for $2 \leqslant i \leqslant n-1$, while f_i, $1 \leqslant i \leqslant n$, are computed. Two additional conditions are

needed. They are given by the boundary conditions, which are Dirichlet conditions of the form $f_1 = a$, $f_n = b$, for grid-generation equations on regions with fixed boundaries. The stencils l_i, c_i, and r_i are needed for $2 \leqslant i \leqslant n-1$. The system of difference equations is said to be symmetric, if

$$l_i = r_{i-1}, \qquad 3 \leqslant i \leqslant n-1. \tag{2.16}$$

Thus, all of the l_i can be computed from l_2 and r_i.

In two dimensions, the stencil names are *lu*, *u*, *ru*, *l*, *c*, *r*, *ld*, *d*, and *rd* with the meanings "left upper," "upper," and so forth. A general nearest-neighbor difference equation for one unknown function, $f_{i,j}$, $1 \leqslant i \leqslant m$, $1 \leqslant j \leqslant n$, has the form

$$\begin{aligned} &lu_{i,j} f_{i-1,j+1} + u_{i,j} f_{i,j+1} + ru_{i,j} f_{i+1,j+1} \\ &\quad + l_{i,j} f_{i-1,j} + c_{i,j} f_{i,j} + r_{i,j} f_{i+1,j} \\ &\quad + ld_{i,j} f_{i-1,j-1} + d_{i,j} f_{i,j-1} \\ &\quad + rd_{i,j} f_{i+1,j-1} = g_{i,j}, \end{aligned} \tag{2.17}$$

for $2 \leqslant i \leqslant m-1$, $2 \leqslant j \leqslant n-1$. These difference equations are symmetric, provided that

$$l_{i,j} = r_{i-1,j}, \qquad 3 \leqslant i \leqslant m-1, \quad 2 \leqslant j \leqslant n-1, \tag{2.18}$$

$$d_{i,j} = u_{i,j-1}, \qquad 2 \leqslant i \leqslant m-1, \quad 3 \leqslant j \leqslant n-1, \tag{2.19}$$

$$lu_{i,j} = rd_{i-1,j+1}, \qquad 3 \leqslant i \leqslant m-1, \quad 2 \leqslant j \leqslant n-2, \tag{2.20}$$

$$ld_{i,j} = ru_{i-1,j-1}, \qquad 3 \leqslant i \leqslant m-1, \quad 3 \leqslant j \leqslant n-1. \tag{2.21}$$

The stencils *l*, *d*, *lu*, and *ld* are computed from these equations and from $l_{2,j}$, $d_{i,2}$, $lu_{2,j}$, $lu_{i,n-1}$, $ld_{2,j}$, and $ld_{i,2}$. Again, for more details, see [12].

3. CURVES

In the paper [9], the authors use geometric intuition to derive variational principles for generating grids on curves, surfaces, planar regions and in volumes. For curves, all principles are essentially the same. In [11], the authors study two algorithms for numerically solving the Euler–Lagrange equations for the variational principle and show that these algorithms have a serious bifurcation problem that severely limits their usefulness. In this section, a new algorithm is introduced that preserves the underlying symmetry of the variational principle. This either eliminates or significantly delays the bifurcation problem. In addition, the reference grid and solution adaptivity parts of the algorithm are shown to work exceptionally well (this was not considered in [11]). For more motivation, details, and notation, see Steinberg and Roache [9, 11].

A curve in three-dimensional space is described parametrically,

$$\mathbf{v}(r) = (x(r), y(r), z(r)), \qquad 0 \leqslant r \leqslant 1, \tag{3.22}$$

and is assumed to be smooth. The metric is needed in the grid generation equations and is defined by

$$\begin{aligned} Q(r) &= \left\| \frac{d\mathbf{v}(r)}{dr} \right\|^2 \\ &= (x'(r))^2 + (y'(r))^2 + (z'(r))^2. \end{aligned} \tag{3.23}$$

(Several notations are used for derivatives: $dx/dr \equiv x_r \equiv x'$.) Recall that the length of an element of the curve is given by

$$\sqrt{Q(r)}\, dr. \tag{3.24}$$

Note that any parameterization provides a discrete grid when discrete increments in the parameterization variable are used to step out discrete points. This leads naturally to the powerful concept of the parameterization being a "continuum grid." A new continuum grid is generated by calculating a re-parameterization of a curve in the form

$$r = r(\xi), \qquad 0 \leqslant \xi \leqslant 1. \tag{3.25}$$

In these discussions, the variables x, y, and z are called physical; r is called parameter, and ξ is called logical. In the logical variable, the length of an element is

$$\sqrt{Q(r)}\, r'(\xi)\, d\xi. \tag{3.26}$$

Both a reference grid (see [9]) and a solution adaptive weight are used to control the grid. If a one-dimensional reference grid is given by $\alpha(\xi)$, $0 \leqslant \xi \leqslant 1$, then the derivative $\omega = \alpha'$ is needed in the grid-generation equations. The reference grid is typically used to obtain an exponential compression of the grid near a boundary by choosing α as an exponential. The use of solution-adaptive weights has not been discussed in our previous work, but the theory provided in [9] makes it easy to set up the required variational problem. Let $u = u(x, y, z)$ be the solution of some physical problem for which an adaptive grid is desired, and let

$$u(r) = u(x(r), y(r), z(r)). \tag{3.27}$$

Recall that the chain rule gives

$$u_\xi = u_r / r_\xi. \tag{3.28}$$

Whether or not the weight W depends on r is critical in the variational problem, so set

$$W(r) = 1 + a\left(\frac{u_r(r)}{r'}\right)^2 \tag{3.29}$$

which is, in fact,

$$W(r) = 1 + a(u_\xi)^2. \tag{3.30}$$

The addition of 1 is used to keep the weight constant when the derivative of u is small and to keep the weight from becoming zero. The a is for scaling. Note that $\sqrt{W} \approx a u_\xi$ when u_ξ is large.

The main idea is to produce grids on the curve so that the grid spacing is proportional to the spacing of the reference grid and inversely proportional to a derivative of the solution,

$$\sqrt{Q(r(\xi))}\, r'(\xi) \propto \frac{\omega(\xi)}{\sqrt{W(r(\xi))}}. \tag{3.31}$$

(From now on, assume $r = r(\xi)$.)

The appropriate variational integral (see [9]) for generating such a grid is

$$I = -\frac{1}{2}\int_0^1 \frac{Q(r)\, W(r)}{\omega(\xi)} r_\xi^2\, d\xi. \tag{3.32}$$

The factor in front of the integral is used for convenience. The Euler–Lagrange equation for the minimization of this integral is

$$\frac{\partial}{\partial \xi}\left(\frac{Q(r)\, W(r)}{\omega(\xi)} r_\xi(\xi)\right) = \frac{1}{2}\left(\frac{W(r)}{\omega(\xi)}\frac{\partial Q(r)}{\partial \xi} + \frac{Q(r)}{\omega(\xi)}\frac{\partial W(r)}{\partial \xi}\right) r_\xi(\xi). \tag{3.33}$$

Note that the left-hand side of the previous equation (principal part) is symmetric.

It is easy to see that, in the case of curves, the Euler–Lagrange equation is integrable; assuming all functions are positive, one integration gives

$$r_\xi \frac{\sqrt{Q(r)\, W(r)}}{\omega(\xi)} = \text{const}, \tag{3.34}$$

which confirms the intuitive discussion given above. This information is not used in the following discussion, because the surface Euler–Lagrange equations are not integrable; we want the methods that we use for curves to extend to the surface problem.

3.1. *Numerical Algorithm for Curves*

The Euler–Lagrange equation (3.33) is solved iteratively. Given an approximate solution for $r = r(\xi)$, the right-hand side of Eq. (3.33) is evaluated using central differences. The left-hand side of (3.33) is differenced by using the schemes discussed in the Section 2.3 on symmetric differencing. This produces a system of nonlinear symmetric finite-difference equations for r. The coefficients of the nonlinear scheme are evaluated using the current approximation for the solution. Then, the resulting linear difference equations are solved, using Gaussian elimination without pivoting (using a standard tridiagonal solver).

4. NUMERICAL VALIDATION FOR CURVES

The curve grid-generation algorithm produces a grid that is the best possible solution, in the least squares sense, of the proportionality given in the previous section. How well the algorithm does this is what is tested in this section. Note that the theory is given in the continuum, so it is expected that the proportionality will be off by a truncation error, and these discrepancies will decrease with increasing grid resolution. The grid length–quality measure is used to evaluate the numerical results.

Unless otherwise stated, all computer runs are done: with no relaxation factor in the nonlinear iteration; with a stopping criterion of the grid changing less than one part in 10^5; and with a random grid as a starting point for the nonlinear iteration. The convergence requirements are far more stringent than the requirements for productions codes. Recall that previous algorithms show anomalous behavior [11]. The question is: how well are those problems controlled?

4.1. *Convergence Rate*

The numerical scheme is second order, as the following tests confirm. The problems used to test the convergence rate are re-parameterizations of the unit interval

$$\begin{aligned} x &= \frac{e^r - 1}{e - 1}, \qquad y = 0, \quad z = 0, \\ x &= \frac{\ln(r+1)}{\ln(2)}, \qquad y = 0, \quad z = 0, \end{aligned} \tag{4.35}$$

where $0 \leqslant r \leqslant 1$. If the parameter space r is divided into equal segments, then the previous parameterizations produce (significantly) irregular grids in the physical space x (y and z are trivial). The algorithm changes each of these grids to a uniform grid in physical space.

The data in Table I was computed by starting with a uniform grid in parameter space and then forcing the convergence to full machine precision. The results for the

TABLE I

Convergence Rates for the Curve Generator

	Exp			Log		
n	n-Max	n-Dev	Ratio	n-Max	n-Dev	Ratio
5	0.1188	0.2839	1.030	0.01587	0.03549	1.004
9	0.1089	0.2620	1.010	0.01310	0.03002	1.001
17	0.09943	0.2429	1.003	0.01170	0.02701	1.000
33	0.09389	0.2311	1.001	0.01019	0.02352	1.000
65	0.08953	0.2187	1.000	0.009947	0.02710	1.000
129	0.1399	0.4682	1.000	0.04959	0.2430	1.000

exponential map are given in columns two through four while the results for the logarithm are presented in columns five through seven. Because the algorithm is second order, some quantities are normalized by multiplying by the square of the number of grid points so that the resulting column will be nearly constant. In this table, n is the number of grid points, n-max is the maximum norm of the difference between the identity map and the computed grid multiplied by n^2, n-dev is the deviation of the grid lengths multiplied by n^2, and *ratio* is the ratio of the lengths of the longest and shortest segments. In the last row, where $n = 128$, the single precision arithmetic is beginning to contaminate the results. The test clearly confirm that the algorithm is second order.

4.2. *Equal Spacing*

One of the goals of this work is to develop an algorithm that generates quality grids on a nominal curve (height one, width one). A parabola

$$y = 4\varepsilon r(1-r), \qquad x = r, \quad z = 0, \quad 0 \leqslant r \leqslant 1, \tag{4.36}$$

of height ε is used for a test curve. The simplest type of grid is one that has equal segment lengths. The algorithm generates such a grid, if the weights are made trivial: $W(r) \equiv 1 \equiv \omega(\xi)$. In Table II, $\varepsilon = 1$, n is the number of grid points, *itr* is the of number nonlinear iterations used, and *dev* is the deviation of the grid lengths. This table shows that the algorithm under consideration does its job well. Note the decrease in *dev* with resolution; in fact,

$$\text{dev} \propto 1/n. \tag{4.37}$$

This relationship holds in a wide range of examples.

TABLE II

Nominal Curve

n	Itr	Dev
3	2	0.0000
5	14	0.1245
9	25	0.075
17	29	0.032
33	42	0.010
65	46	0.003
129	54	0.0007
257	61	0.0002
513	55	0.0001

TABLE III

Bifurcation Values

n	Value
21	∞
41	$3.50 \leqslant \varepsilon \leqslant 3.75$
81	$3.25 \leqslant \varepsilon \leqslant 3.50$
161	$3.75 \leqslant \varepsilon \leqslant 4.25$

Previous versions of this algorithm have bifurcation problems [11]. In the algorithm presented here, this problem is significantly reduced. Again, this difficulty is studied using the parabola (4.36). Typically, for a given number of nodes, the solution to the discrete equations bifurcates for sufficiently large ε. Table III gives bifurcation values for the new algorithm. The first column gives the number of grid points, while the second column gives an estimate for the value of ε at which the solution of the nonlinear equations bifurcate. The value of ε was resolved in increments of 0.25. For large n, the values of ε given by this algorithm are twice as good as the values given by any other version of the algorithm; the performance for coarse grids is exceptional.

Table IV illustrates the lack of bifurcation for a grid containing 21 points. Here ε is the height of the curve. As before, *itr* is the number of nonlinear iterations, and *dev* is the deviation of lengths.

Additional tests were run on the curve

$$x = r, \qquad y = 0, \qquad z = \frac{r}{2} + \varepsilon \sin(\pi r), \qquad 0 \leqslant r \leqslant 1, \tag{4.38}$$

TABLE IV

No Bifurcation

ε	Itr	Dev
0.25	10	0.001
0.50	22	0.004
1.00	27	0.022
2.00	51	0.064
4.00	125	0.091
5.00	140	0.094
10.00	170	0.102
20.00	182	0.103

TABLE V

Tall Curve

n	Itr	Dev	Bifurcation
3	2	0.000	No
5	16	0.186	No
9	38	0.154	No
17	111	0.114	No
33	107	0.103	Yes
65	>400	?	Yes

which appears as the boundary values of a surface used to test the surface grid generator. This curve is more difficult to grid than the quadratic, but the results are still rather similar, so not much is presented here. In the case where $n = 21$ the bifurcation point occurs for $\varepsilon > 10.0$. If a nonlinear relaxation factor of $\frac{1}{2}$ rather than 1 is used, then the number of nonlinear iterations needed to meet a nonlinear tolerance of 10^{-4} is reduced by a factor of three.

For the quadratic curve, Table V illustrates the bifurcation of the grid for $\varepsilon = 25$ (the height of the curve is 25, while its width is 1). The last column indicates if the grid has bifurcated (Note. The iteration count decreases after the bifurcation). Also, note that *dev* decreases with increasing resolution.

It is, of course, desirable to have an algorithm that has no bifurcation. Numerous algorithms are being tested (see [6]); most show significant anomalous behavior that involves either the existence of multiple solutions or bifurcation of the solution of the nonlinear equations. A few show promise and are undergoing further testing. None of the promising algorithms do equi-spacing, but rather, they attempt to produce a "quality" grid. As far as we know, the algorithms presented in [8, 14] have not been tested for such problems.

4.3. *Solution Adaption*

To test the solution-adaptive part of the algorithm, a trivial curve,

$$x = r, \qquad y = 0, \qquad z = 0, \qquad 0 \leqslant r \leqslant 1, \tag{4.39}$$

is used, and the reference grid is set to the identity, $\omega(\xi) \equiv 1$. With this setup, the only effects come from the solution-adaptive weight $W(r)$. Two types of weights are considered: exponential and polynomial. The exponential weights are used to model smooth adaptivity functions while the polynomials are used to model less smooth adaptivity functions. Also, the weights are given as functions of ξ. However, when a value of the weight is looked up: (1) $r = r(\xi)$ is computed; (2) the values of $(x(r), y(r), z(r))$ are found; and (3) the value of the weight is found. Therefore, the value of the weight depends implicitly on r. Our goal is to provide algorithms that can adjust grid length by an order of magnitude. However, adjusting the grid by a factor of two is significant, particularly in multi-dimensional problems. In the following tables, n is the number of points in the grid, *itr* is the number of nonlinear iterations (recall that the convergence criterion is strict), *prescribed ratio* ($= R$) is the ratio the maximum to minimum values of the prescribed weight, *computed ratio* is the ratio of the maximum to minimum length of the computed grid, and *dev* is the deviation of grid lengths.

TABLE VI

Exponential Weight

n	Itr	Prescribed ratio	Computed ratio	Dev
17	13	2.0	1.782	0.04
33	13	2.0	1.883	0.02
65	13	2.0	1.939	0.01

Recall that if the ratio of the maximum to minimum value of the weight is R^2, then the ratio of the maximum to minimum grid lengths should be R. First, the response to a modest skewed weight is tested. In Table VI the weight is taken as a Gaussian exponential with $R = 2$,

$$W(\xi) = e^{-a(x(\xi) - 2/5)^2}, \qquad a = \frac{50 \ln(2)}{9} \approx 3.85. \tag{4.40}$$

(Note that $x = r$.) As the values in the Table VI show, the algorithm performs very well, because the *computed ratio* rapidly approaches the *prescribed ratio* 2. Also, the deviations are good and improve with increasing resolution.

For Table VII a symmetric, but stronger weight with prescribed ratio $R = 10$ is used. The symmetry guarantees that the center point of the solution grid is at the center of the interval. This is checked to guarantee that the grid has not bifurcated. The weight is

$$W(\xi) = e^{-8 \ln(R)(x(\xi) - 1/2)^2}, \tag{4.41}$$

TABLE VII

Exponential Weight

n	Itr	Prescribed ratio	Computed ratio	Dev
9	12	10.0	2.811	0.33
17	7	10.0	4.189	0.20
33	8	10.0	5.712	0.11
65	7	10.0	7.153	0.06
129	7	10.0	8.306	0.03
257	7	10.0	9.071	0.02

TABLE VIII

Trouble Values

n	Itr	Prescribed ratio	Computed ratio	Dev
9	20	13.6	3.036	0.36
17	18	35.0	6.219	0.29
33	20	38.5	10.020	0.19

TABLE IX

Continuous Weight

n	Itr	Computed ratio	Dev
17	110	1.980	0.082
33	124	1.989	0.045
65	133	1.996	0.024

and the nonlinear tolerance for the next two tables is one part in 10^3. For such a strong adaptivity function, the number of iterations is small, the ratios are good and converge to the correct value, and the deviations are reasonable.

It is not possible to use the symmetric algorithm for an arbitrary weight. In Table VIII the value R of the prescribed ratio is increased until the code has problems, typically floating point exceptions or iterative divergence. The column labelled *prescribed ratio* indicates a value of R just below where the code has problems. This is not a bifurcation point, because the nominal algorithm described in [11] can be used to generate grids for much larger values of the prescribed ratio. Unfortunately, the nominal algorithm does not generalize to surfaces. The values of *ratio* and *dev* are not particularly good, because the algorithm is being pushed to its limits.

For the polynomial examples, the weight functions are chosen so that $R = 2$. The weight function W is chosen as a piece-wise polynomial, so that the resulting weight function has a certain degree of smoothness and large constant regions. Thus, the polynomial parts are either constant or the lowest degree polynomial that satisfies either the first, the first and second, or all of the following conditions:

$$P_{a,b,c,d}(a) = c, \qquad P_{a,b,c,d}(b) = d, \tag{4.42}$$

$$P'_{a,b,c,d}(a) = 0, \qquad P'_{a,b,c,d}(b) = 0, \tag{4.43}$$

$$P''_{a,b,c,d}(a) = 0, \qquad P''_{a,b,c,d}(b) = 0. \tag{4.44}$$

The weight function is chosen (with some asymmetry) as

$$\begin{aligned} W(r) &= 1.0, && 0.0 \leqslant r \leqslant 0.2, \\ W(r) &= P_{0.2,0.3,1.0,4.0}(r), && 0.2 \leqslant r \leqslant 0.3, \\ W(r) &= 4.0, && 0.3 \leqslant r \leqslant 0.5, \\ W(r) &= P_{0.5,0.6,4.0,1.0}(r), && 0.5 \leqslant r \leqslant 0.6, \\ W(r) &= 1.0, && 0.6 \leqslant r \leqslant 1.0. \end{aligned} \tag{4.45}$$

In Table IX the weight is composed of linear polynomials and is continuous, but not differentiable. In Table X the weight is composed of cubic polynomials and is continuously differentiable, but not twice differentiable. In Table XI the weight is composed of cubic polynomials and is twice continuously differentiable, but not thrice differentiable. The derivatives of all the polynomials $(b-a)\,P_{a,b,c,d}(r)$ are positive for $a < r < b$, so the nonconstant polynomial parts of the weight are monotonic. In fact, for $a = 0$, $b = 1$, $c = 0$, and $d = 1$ the polynomials are $P(r) = r$, $P(r) = r^2(3 - 2r)$, $P(r) = r^3(10 - 15r + 6r^2)$.

The algorithm is somewhat sensitive when piece-wise polynomial weights are used, so in these cases a nonlinear relaxation factor of 0.1 is used (not all cases need a factor this small). Also, the initial grid is equi-spaced in r.

Again, the ratios and deviations are excellent. A better solution technique or tuned relaxation factor would easily speed up the algorithm. For the polynomial weights, the grid-generation problem becomes more difficult as the smoothness of the weight is increased. This is unexpected, and we have no clear explanation for this behavior. However, we do note that, as the degree of the transition polynomial increases, for the interval $[a, b]$, the values of the polynomial remain bounded, the maximum values of the derivatives increase slowly, the maximum values of the second derivatives increase significantly, the maximum value of the third derivatives increase rapidly, and so forth. This is a possible source of the slow convergence.

4.4. *Reference Grids*

To test the reference-grid concept, again, a trivial curve

$$x = r, \qquad y = 0, \qquad z = 0, \qquad 0 \leqslant r \leqslant 1, \tag{4.46}$$

is used, and the solution-adaptive weight is set to the iden-

TABLE X

Differentiable Weight

n	Itr	Computed ratio	Dev
17	103	1.975	0.092
33	126	1.985	0.051
65	141	1.994	0.026

TABLE XI

Twice Differentiable Weight

n	Itr	Computed ratio	Dev
17	269	1.972	0.098
33	224	1.983	0.052
65	301	1.994	0.028

tity, $W(r) \equiv 1$. With this setup, the only effects come from the reference-grid weight $\omega(\xi)$. When a reference grid $\alpha(\xi)$ is used to determine $\omega(\xi)$, then

$$\omega(\xi) = \alpha_\xi(\xi). \tag{4.47}$$

In this case, the Euler–Lagrange equation becomes

$$\frac{r_{\xi\xi}}{r_\xi} = \frac{\alpha_{\xi\xi}}{\alpha_\xi}. \tag{4.48}$$

Note that this equation is linear, so the code computes the solution in one nonlinear iteration. In addition, if both the first and second derivatives of r and α are differenced in the same way, then the solution produced by the code is $r = \alpha$; i.e., the reference grid is replicated. Compression by five orders of magnitude over nine grid points is trivial to achieve, the compression being limited only by machine word length.

5. SURFACES

Though development of grid generators for surfaces is analogous to that for curves, two functionals play a central role: one is for segment-length control, and the other is for cell-area control. A third functional for orthogonality control is also implemented, but this is not as significant. One important point is that the solution-adaptive weights for each type of control must be chosen in a consistent fashion. A nice aspect of a reference grid is that the reference weights are always consistent. The discussion begins with a brief review of the material in [9] for surface-grid generation. Let

$$\mathbf{v} = \mathbf{v}(r, s) = (x(r, s),\ y(r, s),\ z(r, s)), \qquad 0 \leqslant r,\ s \leqslant 1, \tag{5.49}$$

define a surface. The problem at hand is to generate a new "continuum grid" by reparameterizing the surface,

$$r = r(\xi, \eta), \qquad s = s(\xi, \eta), \qquad 0 \leqslant \xi,\ \eta \leqslant 1, \tag{5.50}$$

so that the resulting (continuum) grid has some desired properties. In terms of the reparameterization

$$\begin{aligned} x(\xi, \eta) &= x(r(\xi, \eta), s(\xi, \eta)), \\ y(\xi, \eta) &= y(r(\xi, \eta), s(\xi, \eta)), \\ z(\xi, \eta) &= z(r(\xi, \eta), s(\xi, \eta)). \end{aligned} \tag{5.51}$$

Then, the surface is also given by

$$\mathbf{v} = \mathbf{v}(\xi, \eta) = (x(\xi, \eta),\ y(\xi, \eta),\ z(\xi, \eta)). \tag{5.52}$$

Here, x, y, and z are physical variables; r and s are parameter variables; and ξ and η are logical variables.

The tangent vectors to coordinate lines are needed for the grid generation equations. The tangent vectors to parameter coordinates are

$$\frac{\partial \mathbf{v}}{\partial r}, \frac{\partial \mathbf{v}}{\partial s}, \tag{5.53}$$

while the tangent vectors to logical coordinates are

$$T_\xi = \frac{\partial \mathbf{v}}{\partial \xi}, \qquad T_\eta = \frac{\partial \mathbf{v}}{\partial \eta}. \tag{5.54}$$

Now, the reference grid is two-dimensional, so let

$$(\alpha(\xi, \eta), \beta(\xi, \eta)), \qquad 0 \leqslant \xi, \eta \leqslant 1, \tag{5.55}$$

define a planar reference space. Then, after the grid is discretized, define the reference weight ω_1 as the length of a parameter grid line in ξ direction,

$$\omega_1(\xi, \eta) = \sqrt{\alpha_\xi^2(\xi, \eta) + \beta_\xi^2(\xi, \eta)}. \tag{5.56}$$

The weight ω_2 is defined similarly, using the η derivative. There are two solution adaptive weights: W_1 for the ξ direction, and W_2 for the η direction.

5.1. *Length Control*

The length functional is defined in terms of the surface metrics:

$$\begin{aligned} P = P(r, s) &= \left\| \frac{\partial \mathbf{v}}{\partial r} \right\|^2 \\ &= \left(\frac{\partial x}{\partial r}\right)^2 + \left(\frac{\partial y}{\partial r}\right)^2 + \left(\frac{\partial z}{\partial r}\right)^2, \end{aligned} \tag{5.57}$$

$$\begin{aligned} Q = Q(r, s) &= \frac{\partial \mathbf{v}}{\partial r} \circ \frac{\partial \mathbf{v}}{\partial s} \\ &= \frac{\partial x}{\partial r}\frac{\partial x}{\partial s} + \frac{\partial y}{\partial r}\frac{\partial y}{\partial s} + \frac{\partial z}{\partial r}\frac{\partial z}{\partial s}, \end{aligned} \tag{5.58}$$

$$\begin{aligned} R = R(r, s) &= \left\| \frac{\partial \mathbf{v}}{\partial s} \right\|^2 \\ &= \left(\frac{\partial x}{\partial s}\right)^2 + \left(\frac{\partial y}{\partial s}\right)^2 + \left(\frac{\partial z}{\partial s}\right)^2. \end{aligned} \tag{5.59}$$

If

$$F(r, s, a, b) = Pa^2 + 2Qab + Rb^2, \tag{5.60}$$

then the functional to be minimized is

$$-\frac{1}{2}\int\left(\frac{W_1(r,s)}{\omega_1(\xi,\eta)}F(r,s,r_\xi,s_\xi) + \frac{W_2(r,s)}{\omega_2(\xi,\eta)}F(r,s,r_\eta,s_\eta)\right)d\xi\,d\eta. \tag{5.61}$$

In [9], the norms of tangent vectors to the logical-coordinate lines on the surface are given by

$$\begin{aligned}\|T_\xi\|^2 &= F(r,s,r_\xi,s_\xi),\\ \|T_\eta\|^2 &= F(r,s,r_\eta,s_\eta),\end{aligned} \tag{5.62}$$

so that, intuitively, the minimization problem produces grids with

$$\|T_\xi\| \propto \frac{\omega_1(\xi,\eta)}{\sqrt{W_1(r,s)}}, \qquad \|T_\eta\| \propto \frac{\omega_2(\xi,\eta)}{\sqrt{W_2(r,s)}}. \tag{5.63}$$

Introduce

$$\frac{\partial F}{\partial r}(r,s,a,b) = \frac{\partial P}{\partial r}a^2 + 2\frac{\partial Q}{\partial r}ab + \frac{\partial R}{\partial r}b^2, \tag{5.64}$$

$$\frac{\partial F}{\partial s}(r,s,a,b) = \frac{\partial P}{\partial s}a^2 + 2\frac{\partial Q}{\partial s}ab + \frac{\partial R}{\partial s}b^2. \tag{5.65}$$

Then, the Euler–Lagrange equations for the minimization of the functional are:

$$\begin{aligned}&\frac{\partial}{\partial\xi}\left(\frac{W_1}{\omega_1}(Pr_\xi + Qs_\xi)\right) + \frac{\partial}{\partial\eta}\left(\frac{W_2}{\omega_2}(Pr_\eta + Qs_\eta)\right)\\ &\quad= \frac{1}{2}\left(\frac{F(r,s,r_\xi,s_\xi)}{\omega_1}\frac{\partial W_1}{\partial r} + \frac{F(r,s,r_\eta,s_\eta)}{\omega_2}\frac{\partial W_2}{\partial r}\right)\\ &\quad\quad+ \frac{1}{2}\left(\frac{W_1}{\omega_1}\frac{\partial F}{\partial r}(r,s,r_\xi,s_\xi) + \frac{W_2}{\omega_2}\frac{\partial F}{\partial r}(r,s,r_\eta,s_\eta)\right),\end{aligned} \tag{5.66}$$

$$\begin{aligned}&\frac{\partial}{\partial\xi}\left(\frac{W_1}{\omega_1}(Qr_\xi + Rs_\xi)\right) + \frac{\partial}{\partial\eta}\left(\frac{W_2}{\omega_2}(Qr_\eta + Rs_\eta)\right)\\ &\quad= \frac{1}{2}\left(\frac{F(r,s,r_\xi,s_\xi)}{\omega_1}\frac{\partial W_1}{\partial s} + \frac{F(r,s,r_\eta,s_\eta)}{\omega_2}\frac{\partial W_2}{\partial s}\right)\\ &\quad\quad+ \frac{1}{2}\left(\frac{W_1}{\omega_1}\frac{\partial F}{\partial s}(r,s,r_\xi,s_\xi) + \frac{W_2}{\omega_2}\frac{\partial F}{\partial s}(r,s,r_\eta,s_\eta)\right).\end{aligned} \tag{5.67}$$

The derivatives with respect to the parameter variables are difficult to compute, while the derivatives with respect to the logical variables are easy to compute. Therefore, the former is eliminated in favor of the latter by using the chain rule

$$\begin{aligned}\frac{\partial}{\partial r} &= \frac{1}{J}\left(+s_\eta\frac{\partial}{\partial\xi} - s_\xi\frac{\partial}{\partial\eta}\right),\\ \frac{\partial}{\partial s} &= \frac{1}{J}\left(-r_\eta\frac{\partial}{\partial\xi} + r_\xi\frac{\partial}{\partial\eta}\right).\end{aligned} \tag{5.68}$$

Here J is the Jacobian of the transformation between the (ξ, η) and the (r, s) variables,

$$J = J\begin{pmatrix} r & s\\ \xi & \eta\end{pmatrix} = \det\begin{bmatrix}\frac{\partial r}{\partial\xi} & \frac{\partial s}{\partial\xi}\\ \frac{\partial r}{\partial\eta} & \frac{\partial s}{\partial\eta}\end{bmatrix}. \tag{5.69}$$

5.2. *Area Control*

The area functional is defined in terms of the square of the element of surface area

$$S(r,s) = J^2\begin{pmatrix} x & y\\ r & s\end{pmatrix} + J^2\begin{pmatrix} x & z\\ r & s\end{pmatrix} + J^2\begin{pmatrix} y & z\\ r & s\end{pmatrix}. \tag{5.70}$$

In logical coordinates, the chain rule gives the element as

$$\sqrt{S(r,s)}\,J\begin{pmatrix} r & s\\ \xi & \eta\end{pmatrix}. \tag{5.71}$$

If each side of a cell is changed by a factor, then the area is changed by the product of the factors. Therefore, the product of the solution adaptive weights is used in the area algorithm. It is natural to use the areas of reference cells $A(\xi, \eta)$ as a weight for the area functional. In the case where the reference grid is rectangular, $A(\xi, \eta) = \omega_1(\xi, \eta)\,\omega_2(\xi, \eta)$. The grid is required to satisfy

$$\sqrt{S(r,s)}\,J\begin{pmatrix} r & s\\ \xi & \eta\end{pmatrix} \propto \frac{A(\xi,\eta)}{\sqrt{W_1(r,s)\,W_2(r,s)}} \tag{5.72}$$

(see [9] for more details), and, consequently, the functional to be minimized is

$$-\frac{1}{2}\int\frac{S(r,s)\,W_1(r,s)\,W_2(r,s)}{A(\xi,\eta)}J^2\begin{pmatrix} r & s\\ \xi & \eta\end{pmatrix}d\xi\,d\eta. \tag{5.73}$$

The Euler–Lagrange equations are

$$+\frac{\partial}{\partial\xi}\left(\frac{JSW_1W_2}{A}\right)s_\eta-\frac{\partial}{\partial\eta}\left(\frac{JSW_1W_2}{A}\right)s_\xi$$
$$=\frac{1}{2}\frac{\partial}{\partial r}(SW_1W_2)\frac{J^2}{A}, \tag{5.74}$$
$$-\frac{\partial}{\partial\xi}\left(\frac{JSW_1W_2}{A}\right)r_\eta+\frac{\partial}{\partial\eta}\left(\frac{JSW_1W_2}{A}\right)r_\xi$$
$$=\frac{1}{2}\frac{\partial}{\partial s}(SW_1W_2)\frac{J^2}{A}. \tag{5.75}$$

The left-hand side of these equations can be put into symmetric form:

$$+\frac{\partial}{\partial\xi}\left(\frac{JSW_1W_2}{A}s_\eta\right)-\frac{\partial}{\partial\eta}\left(\frac{JSW_1W_2}{A}s_\xi\right)$$
$$=\frac{1}{2}\frac{\partial}{\partial r}(SW_1W_2)\frac{J^2}{A}, \tag{5.76}$$
$$-\frac{\partial}{\partial\xi}\left(\frac{JSW_1W_2}{A}r_\eta\right)+\frac{\partial}{\partial\eta}\left(\frac{JSW_1W_2}{A}r_\xi\right)$$
$$=\frac{1}{2}\frac{\partial}{\partial s}(SW_1W_2)\frac{J^2}{A}. \tag{5.77}$$

These equations are in a symmetric "quasi-uncoupled" (i.e., nonlinear coupling only) form

$$+\frac{\partial}{\partial\xi}(+\alpha_1r_\xi-\beta_1r_\eta)+\frac{\partial}{\partial\eta}(-\beta_1r_\xi+\gamma_1r_\eta)$$
$$=\frac{1}{2}\frac{\partial}{\partial r}(SW_1W_2)\frac{J^2}{A}, \tag{5.78}$$
$$+\frac{\partial}{\partial\xi}(+\alpha_2s_\xi-\beta_2s_\eta)+\frac{\partial}{\partial\eta}(-\beta_2s_\xi+\gamma_2s_\eta)$$
$$=\frac{1}{2}\frac{\partial}{\partial s}(SW_1W_2)\frac{J^2}{A}, \tag{5.79}$$

where

$$\alpha_1=\frac{SW_1W_2s_\eta^2}{A},$$
$$\beta_1=\frac{SW_1W_2s_\xi s_\eta}{A}, \tag{5.80}$$
$$\gamma_1=\frac{SW_1W_2s_\xi^2}{A},$$
$$\alpha_2=\frac{SW_1W_2r_\eta^2}{A},$$
$$\beta_2=\frac{SW_1W_2r_\xi r_\eta}{A}, \tag{5.81}$$
$$\gamma_2=\frac{SW_1W_2r_\xi^2}{A}.$$

5.3. *Orthogonality Control*

The orthogonality control attempts to keep the grid lines orthogonal by keeping the inner product of tangent vectors to the grid lines zero (see [9] for more details). The orthogonality functional is also defined in terms of the surface metrics $P=P(r,s)$, $Q=Q(r,s)$, and $R=R(r,s)$ that are defined in the section on length control. The inner product of the two vectors tangent to the logical coordinate lines is

$$F=Pr_\xi r_\eta+Q(r_\xi s_\eta+r_\eta r_\xi)+Rs_\xi s_\eta, \tag{5.82}$$

so the integral to be minimized is

$$\int F^2\,d\xi\,d\eta. \tag{5.83}$$

Let

$$F_r=P_rr_\xi r_\eta+Q_r(r_\xi s_\eta+r_\eta r_\xi)+R_rs_\xi s_\eta, \tag{5.84}$$
$$F_s=P_sr_\xi r_\eta+Q_s(r_\xi s_\eta+r_\eta r_\xi)+R_ss_\xi s_\eta, \tag{5.85}$$

and

$$A=Pr_\eta+Qs_\eta, \qquad B=Qr_\eta+Rs_\eta, \tag{5.86}$$
$$C=Pr_\xi+Qs_\xi, \qquad D=Qr_\xi+Rs_\xi. \tag{5.87}$$

Then the Euler–Lagrange equations are

$$\frac{\partial}{\partial\xi}(A^2r_\xi)+\frac{\partial}{\partial\eta}(C^2r_\eta)+\frac{\partial}{\partial\xi}(ABs_\xi)+\frac{\partial}{\partial\eta}(CDs_\eta)$$
$$=FF_r, \tag{5.88}$$
$$\frac{\partial}{\partial\xi}(ABr_\xi)+\frac{\partial}{\partial\eta}(CDr_\eta)+\frac{\partial}{\partial\xi}(B^2s_\xi)+\frac{\partial}{\partial\eta}(D^2s_\eta)$$
$$=FF_s. \tag{5.89}$$

5.4. *Numerical Algorithm for Surfaces*

The numerical algorithm generates a grid on the interior and boundary of the surface simultaneously; that is, the interior and boundary Euler–Lagrange equations are solved using a simultaneous iteration. The interior grid depends on the boundary grid but not conversely. The interior algorithm uses a linear combination of length, area, and orthogonality control [9], while the boundary algorithm is obtained by reducing the interior length control to the boundaries. Both the interior and boundary algorithms use the reference grid and solution adaptive weights.

The boundary algorithm given by this approach differs from that for curves given in Section 4 of this paper. To see this, compare the formulas for the reference grid weights for curves (4.47) and surfaces (5.56). In the surface case there is a square root of a sum of squares that simplifies to a single

term in the curve case. In Section 4, this simplification was used while here it is not applicable. This results in the difference formulas for the reference weights that differ by a truncation error. As a consequence, the reference grid will not be exactly replicated on the boundary, even in simple problems. The replication of the reference grid in the interior, especially for nontrivial examples, is even more difficult (see [2]). However, in rectangular regions, replication is obtained up to a second-order truncation error, so that, in the limit, reference grids are reproduced exactly.

Given an approximate solution for r and s, the right-hand sides of the Euler–Lagrange equations are evaluated using central differences. The left-hand sides are differenced using the schemes discussed in Section 2. This produces a system of nonlinear symmetric finite-difference equations for r and s. The coefficients of the difference scheme are evaluated using the current approximate solution. Then, the terms in the first equation that involve s as an unknown are lagged one iteration, while the terms in the second equation that involve r are lagged. The resulting linear equations for the surface and boundary are solved using an SOR algorithm. (SOR is used for simplicity; other algorithms could be used to gain speed.)

6. NUMERICAL VALIDATION FOR SURFACES

First the convergence rate for the surface generator is tested by using it to convert a distorted parameterization of the unit square to the identity parameterization $x = r$, $y = s$. Note that the unit square is a trivial surface and that the identity map is the obvious solution of the grid generation equations. In this test and the next, the reference weight and solution adaptive weight are set equal to one. In this case, the grid generator will try to produce a uniform grid. This is tested on two surfaces. The orthogonality functional is tested separately. Finally the reference weight and solution adaptivity are tested. The surface grid generator does perform well and is capable of producing suitable grids for a wide range of problems. Again, the performance of the algorithms is judged using the quality measures for length, area, and orthogonality. In general, the initial grid is equi-distributed in r and s.

6.1. *Convergence Rate*

The numerical scheme is second order, as the following test confirms. The problem used to test the convergence rate is a re-parameterization of the unit square,

$$x = r + \frac{\sin(\pi r)\sin(\pi s)}{2\pi},$$
$$y = s + \frac{\sin(\pi r)\sin(\pi s)}{2\pi}, \qquad (6.90)$$
$$z = 0,$$

TABLE XII

Convergence Rate for Surface Generator

n	Itr	Dev-length	Dev-area	angle	Max-norm	Mean-norm	Midpoint x	Midpoint y
5	10	0.011	0.019	88.73	0.0080	0.042	0.50546	0.50546
17	13	0.0031	0.0050	89.64	0.0022	0.022	0.50143	0.50143
33	15	0.00088	0.0013	89.90	0.00060	0.012	0.50041	0.50039
65	31	0.00065	0.00089	89.94	0.00043	0.018	0.50040	0.50031

where $0 \leqslant r, s \leqslant 1$. If the parameter space (r, s), is divided into equal squares, then the previous parameterization produces an irregular grid in the physical space. The algorithm changes this irregular grid to a uniform grid.

The convergence rate test is done with the length and area functionals equally weighted, and the orthogonality functional omitted. The results are reported in Table XII. The nonlinear tolerance was set to 10^{-5} while the linear tolerance was set to 10^{-6} with a maximum number of linear iterations being 100. The grid is n by n. The number of nonlinear iterations needed to meet the tolerance is *itr* while *dev-length* is the normalized standard deviation of the lengths of the cell edges, *dev-area* is the normalized standard deviation of the cell areas, and *angle* gives a measure of the angles between the grid lines (see Section 2.1 for more details).

The grid is supposed to converge to a uniform grid. The error is computed using both the maximum norm *max-norm* and the mean square measure *mean-norm*. The columns labelled x and y are the coordinates of the midpoint of the grid, and both should converge to 0.5.

Note that the number of nonlinear iterations climbs with n as expected, with a repeated substitution algorithm. The last row, where $n = 65$, has limited interest; because only single precision arithmetic is used, it is not computed accurately. The algorithm is second-order accurate, which is confirmed by the fact that numbers in the columns *dev-length*, *dev-area*, and *max-norm* all decrease by roughly a factor of 4 when n is increased by a factor of 2. The *mean-norm* column should remain constant for a second-order method. Also, note that the convergence of the center point is second order.

6.2. *Uniform Grids*

The results of the tests for two surfaces are presented: the first surface is a quardratic bump, while the second is a wave. The nonlinear tolerance was set to 10^{-4} while the linear tolerance was set to 10^{-5}. The bump grid is 17 by 17 while the wave is 21 by 21. The bump is given by

$$x = r, \qquad y = s, \qquad z = 16\varepsilon(r-1)\,r(s-1)\,s, \quad (6.91)$$

where $0 \leqslant r, s \leqslant 1$ and ε gives the height of the bump.

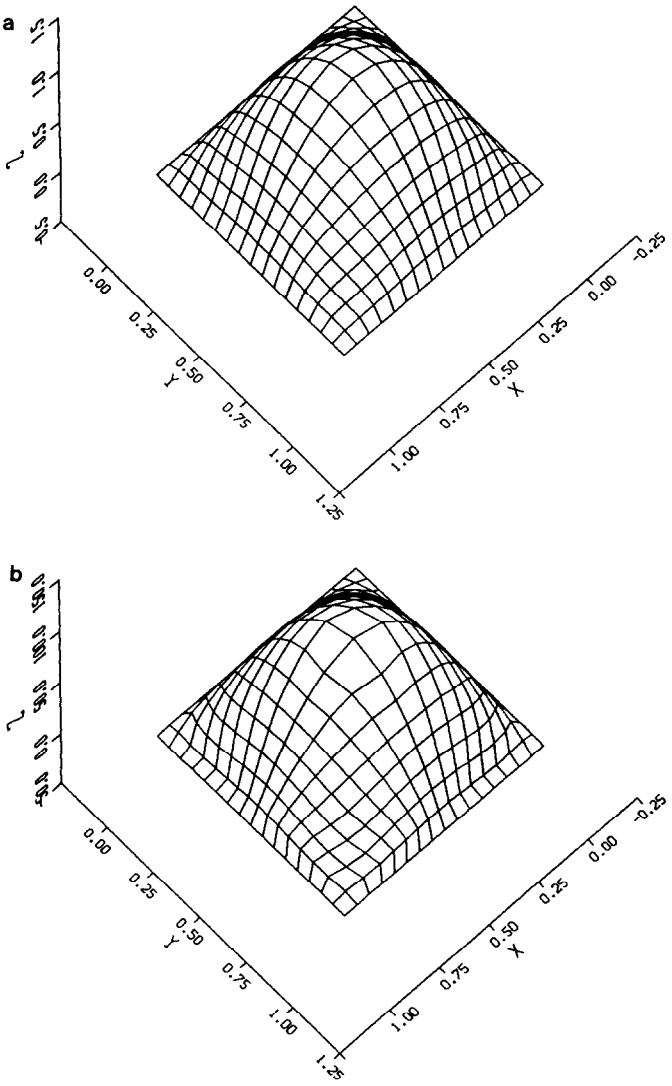

FIG. 1. (a) Bump of height 1; (b) Bump of height 100.

Figure 1a gives the grid for $\varepsilon = 1$ while Fig. 1b gives the grid for $\varepsilon = 16$. The algorithm generates grids for surfaces of any height (that is, at least for $\varepsilon = 1000$). This is in agreement with the results for the quadratic curve where grids were generated on curves of arbitrary height for a 21 by 21 grid (see Table II). Table XIII gives some data for these tests. Both the graphics and quality measures show that the grid is acceptable.

TABLE XIII

Data for Quadratic Curve Tests

Height ε	Itr	Dev-length	Dev-area
1	7	0.15	0.23
100	15	0.36	0.28

TABLE XIV

The Corner Angle at $r = s = 0$ for the Wave Surface

Height ε	θ (in degrees)
0	78
$\frac{1}{2}$	35
1	21
2	12

The second surface is a wave that was previously introduced in [8] and also was studied in [11],

$$x = r, \qquad y = s, \qquad z = \frac{r+s}{2} + \varepsilon \sin(\pi(r+s)), \tag{6.92}$$

where $0 \leqslant r, s \leqslant 1$. This surface is difficult to grid, partly because its corner angles become very acute as ε increases (see Table XIV). The grid generated for this surface by previous algorithms [11] bifurcates for rather small values of ε ($\varepsilon \approx 1$ for a 21 by 21 grid). The new algorithm does much better.

The bifurcation point in ε is located by observing the number of nonlinear iterations needed for convergence. Before the bifurcation point, the number increases with ε; while after the bifurcation, the number decreases. See Table XV for the dependence of the bifurcation point on the size of the grid. The results presented after Eq. (4.38) show that the bifurcation point for the boundary curve is larger than for the full algorithm, so the bifurcation point is determined by the interior algorithm. In a 33 by 33 grid, the algorithm becomes unstable before the bifurcation point; the algorithm can be stabilized using under-relaxation (a factor of $\frac{1}{2}$ was used). (Recall that a relaxation factor also had a significant impact on the convergence of the curve algorithm.) In this case, the bifurcation point satisfies $\varepsilon > 3.1$. This represents a significant improvement over all other algorithms tested.

To clearly demonstrate the quality of a 21 by 21 grid that is generated on the wave for $\varepsilon = 2$, three views are presented in Fig. 2. The views show that the grid is uniform. A grid is also generated for $\varepsilon = \frac{1}{2}$, but is not presented as it looks like

TABLE XV

Bifurcation for a Wave

n	Bifurcation
17	$2.5 \leqslant \varepsilon \leqslant 2.7$
21	$2.5 \leqslant \varepsilon \leqslant 2.6$
33	$\varepsilon > 3.1$

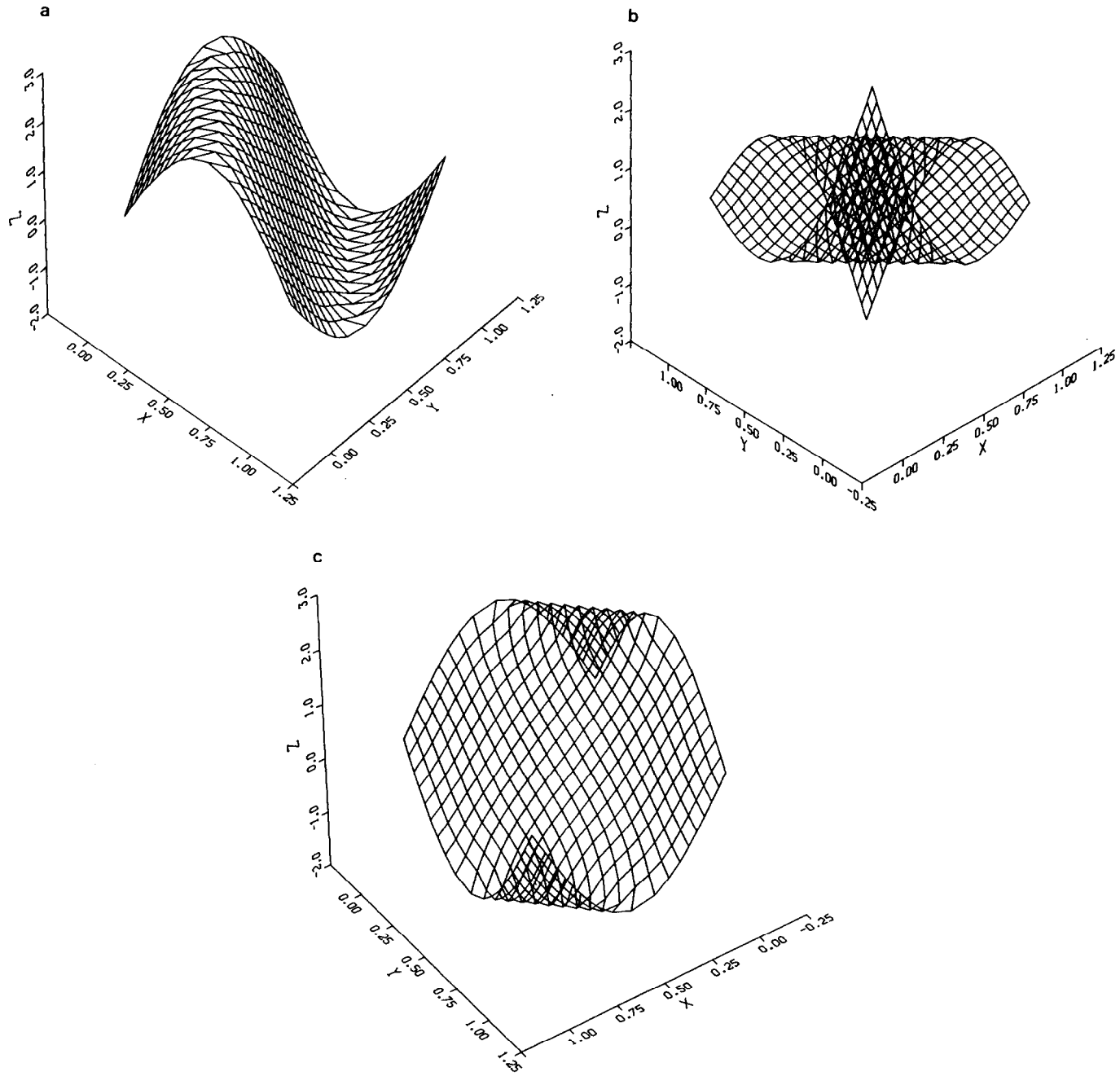

FIG. 2. Three views of a wave.

TABLE XVI

Data for the Wave

Height ε	Itr	Dev-length	Dev-area	Dev-angle
$\frac{1}{2}$	21	0.022	0.10	52
2	64	0.059	0.16	20

the one for $\varepsilon = 2$, only a bit better. Data from these computations is presented in Table XVI (a nonlinear relaxation factor of $\frac{1}{2}$ and a uniform initial grid was used).

6.3. *Orthogonality*

The orthogonality control (with no length or area control) is used to generate a grid on a section of a sphere of radius one: $\pi/4 \leqslant \phi \leqslant 3\pi/4$, $0 \leqslant \theta \leqslant \pi$, where ϕ is the polar

angle and θ is the equatorial angle. Spherical coordinates are used to parameterize the sphere; this parameterization produces orthogonal coordinates, so the code does not move the points.

Also, the orthogonality is tested by repeating the run in the convergence rate test, where $n = 17$, with length, area, and orthgonality equally weighted. Note that the limit grid in this problem is orthogonal. The orthogonality control changes the generated grid by less than 5% (some change on the order of truncation error is to be expected).

6.4. *Reference Grid Tests*

To test the reference grid portion of this algorithm, the surface is set to the unit square (so the metrics are trivial) and the solution adaptive weight is set to one. Tests run for 17 by 17 grids in a square planar region typically converge to machine precision in two or, at most, three nonlinear iterations. (The equations are no longer linear because of the area and orthogonality controls.) For the data in Table XVII, the grids were converged to full machine precision. The reference grid is rectangular with an exponential compression along one side. The spacing goes from 2^{-1} to 2^{-n+1}, where n is the number of intervals in the logical direction where the grid is being compressed (in this case, from 0.5 to 0.0003). As discussed before, the reference grid is not exactly replicated. However, the ratios of the grid length are very accurate. In Table XVII the column headings, *dev-length*, *dev-area*, and *dev-angle*, are quality measures, while the row headings tell which controls are turned on (*all* means *length*, *area*, and *orthogonality*). The grids produced by each of the tests are nearly identical.

Because the code converges so fast when a reference grid is used, it is worthwhile to put as much information as possible in the reference grid rather than in the solution-adaptive weights.

6.5. *Solution Adaptivity Tests*

The solution adaptivity is first tested by using the exponential weight given in Eq. (4.40) in one of the logical directions, while the weight in the other direction is taken as one. In the logical direction where the exponential weight is used, the grid is essentially identical to the grid computed by the curve generator, while in the direction where the weight is one, the grid is uniformly spaced. This is true for length, area, length and area, and length and area and orthogonality controls. As the results are essentially the same as those in Table V, they are not presented.

TABLE XVII

Reference Grid Tests

17×17	Dev-length	Dev-area	Dev-angle
Length control	0.0305	0.0610	90.00
Area control	0.0303	0.0610	90.00
Length & area controls	0.0304	0.0609	89.99
All controls	0.0305	0.0610	89.99

TABLE XVIII

Exponential Solution-Adaptive Weights

n	Dev-length	Dev-area	Dev-angle	Itr
17	0.021	0.054	89.98	6
33	0.011	0.026	89.89	6
65	0.010	0.0060	89.52	15

The next test, presented in Table XVIII, uses an exponential weight in each logical direction,

$$W_1(\xi, \eta) = e^{-a(x(\xi,\eta)-x_0)^2},$$
$$W_2(\xi, \eta) = e^{-b(y(\xi,\eta)-y_0)^2}, \tag{6.93}$$

where $a = b = 4$ and $x_0 = y_0 = \frac{2}{5}$. Equally weighted length, area, and orthogonality control is used. If the orthogonality control is removed in the $n = 65$ case, then the algorithm converges in 13 iterations.

7. COMMENTS

Planar regions are special cases of surfaces, so the surface grid generator can be used to grid such regions. However, some explicit parameterization of the region, where the parameters range over a square, is needed. If the planar region is assumed to be in the $x - y$ plane, then

$$x = r, \qquad y = s, \qquad z = 0. \tag{7.94}$$

gives a one-to-one and onto map of (r, s) to (x, y). However, the domain of the mapping is not necessarily square.

In fact, the theory given in Section 5 does not depend on the domain being a square, so it can be applied using the mapping (7.94). In this case, the surface metrics and Jacobian are trivial:

$$P = 1, \qquad Q = 0, \qquad R = 1, \qquad J = 1. \tag{7.95}$$

Now there is no longer a need for the explicit surface parameterization, because the information given in (7.95) is all that is needed in the formulas used in Section 5.

This information was used to convert a surface code to a planar code.

Next, we note that the continuum grids do not depend

on the parameterization. The variational problems are all derived from proportionalities (see (3.31), (5.63), and (5.72)), which are defined in terms of quantities that do not depend on the parameterization. The geometric quantities are defined in terms of the logical variables, namely, lengths of tangent and normal vectors and thus do not depend on the parameterization. It is a simple exercise to confirm this using the chain rule. The solution adaptive weights are defined in terms of the physical variables (the parameterization only appears implicitly) and the reference grid only involves the logical variables. Thus, for a given functional, its value only depends on the geometric object, the solution adaptive weight, and the reference grid. If the parameterization of the object is changed, the values of the functionals do not change.

If the minimization problem that involves linear combinations of the functionals always possessed a unique solution, then the continuum grids would always be unique. The existence and uniqueness for curves is shown in Steinberg and Roache [11]. Similar results are available for the length functional in higher dimensions for some geometries (see Dvinsky [4]). Partial results for the area functional are given in [2, 3].

On the other hand, the discrete grid-generation algorithm does explicitly depend on the parameterization because the parameterization is used to compute many quantities; that is, when the parameterization is changed the values of the functional changes even if the discrete points on the geometric object do not change. When the discrete grid-generation equations have multiple solutions, then the generated discrete grid can change if anything in the algorithm is changed, for example, the initial data or the parameterization. Numerical evidence indicates that the discrete equations possess a unique solution for sufficiently high resolution. However, there are no mathematical results to justify this.

The convergence rate tests clearly demonstrate that the solutions of the discrete equations do depend on the parameterization. In these tests, when intervals or squares are parameterized in different ways, the generated grids are different. However, numerically, the solutions of the grid-generation equations have second-order convergence. Note that the difference schemes are clearly second order. Thus the numerical evidence indicates that generated discrete grids which well resolve the geometry, essentially, do not depend on the parameterization.

Unfortunately, the mathematical possibility exists that there is a continuum problem which has a unique solution and for which every discrete approximation, no matter how high the resolution, has multiple solutions.

REFERENCES

1. R. Arina, in *Numerical Grid Generation in Computational Fluid Mechanics '88, Miami, Florida, 1988*, edited by S. Sengupta *et al.* (Pineridge Press, Swansea, 1988), p. 351.
2. J. E. Castillo, Thesis, Department of Mathematics, University of New Mexico, 1987 (unpublished).
3. J. E. Castillo, S. Steinberg, and P. J. Roache, *J. Comput. Appl. Math.* **20**, 127 (1987).
4. A. S. Dvinsky, in *Numerical Grid Generation in Computational Fluid Mechanics '88, Miami, Florida, 1988*, edited by S. Sengupta *et al.* (Pineridge Press, Swansea, 1988), p. 351.
5. P. Eiseman, *Appl. Math. Comput.* **21**, 233 (1987).
6. P. Knupp, Thesis, Department of Mathematics, University of New Mexico, 1989 (unpublished).
7. *VAX UNIX MACSYMA Reference Manual*, Version 11 (Symbolics, Inc., 1985).
8. J. Saltzman, *J. Comput. Phys.* **63**, 1 (1986).
9. S. Steinberg and P. J. Roache, *Numer. Methods PDEs* **2**, 71 (1986).
10. J. E. Castillo, S. Steinberg, and P. J. Roache, *Appl. Math. Comput.* **28**, 155 (1988).
11. S. Steinberg and P. J. Roache, *J. Comput. Phys.* **91**, 255 (1990).
12. S. Steinberg and P. J. Roache, Technical Report, Department of Mathematics, University of New Mexico, 1990.
13. J. F. Thompson, Z. U. A. Warsi, and C. W. Mastin, *Numerical Grid Generation* (North Holland, New York, 1985).
14. Z. U. A. Warsi, *J. Comput. Phys.* **64**, 82 (1986).