JOURNAL OF COMPUTATIONAL PHYSICS 106, 77–91 (1993)

Computing Minimal Surfaces via Level Set Curvature Flow

DAVID L. CHOPP*

Mathematics Department, University of California, Berkeley, California 94720

Received October 17, 1991; revised June 18, 1992

1. INTRODUCTION

Given a finite union Γ of fixed closed curves in R^3, there is an associated problem of finding a surface of least area which has Γ as its boundary. This is equivalent to finding a surface of constant mean curvature zero having Γ as its boundary. In this paper, a new approach to numerically solving this problem is introduced. The surface is represented as a level set of a global function $\Phi: R^3 \to R$. A given initial surface evolves according to mean curvature flow until a steady state solution is achieved. A new system of interpolatory boundary conditions are used to maintain the connection between the moving surface and the fixed boundary contour.

Minimal surfaces arise naturally in many physical models, the most familiar being soap films fitting to a wire boundary. In this context, a soap film is modeled as two adjacent thin films acted upon by surface tension. The film is assumed to have negligible mass and constant pressure on both sides. A relationship between the shape of the surface and the pressure is obtained from Laplace's equation

$$0 = p = T\kappa,$$

where p is the pressure difference across the film, T is the surface tension of the film, and κ is the mean curvature; thus, surfaces of constant mean curvature zero provide a close approximation to soap film shapes.

Minimal surfaces also appear in the study of flexible and inextensible films [13], biology [5], translation nets [8], relativity theory [17–19], quantum string theory [16], medical technology [23], and architecture [10].

In general, the solution to the minimal surface problem is not unique. For a given boundary Γ, there may be several distinct surfaces with constant mean curvature zero. A surface is *locally minimal* with respect to Γ if any small perturbation of the surface increases the total surface area.

* This work was supported in part by the Applied Mathematics Subprogram of the Office of Energy Research U.S. Department of Energy under Contract DE-AC03-76SF-00098 and by an International Business Machines Graduate Fellowship.

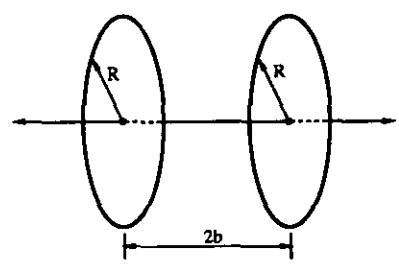

FIG. 1. A two ring boundary arrangement.

A surface is *globally minimal* if it has the least area of any surface which has Γ for its boundary. If a surface is locally minimal, then that surface has mean curvature zero (see Nitsche [14]). The converse is not necessarily true, since non-locally minimal surfaces with mean curvature zero exist. Such surfaces are called *unstable*. Nonetheless, the accepted definition of minimal surface includes all surfaces with mean curvature zero.

An example of a boundary having multiple minimal surfaces is given by two concentric rings as in Fig. 1. With rings close enough together in relation to the ring radii, three topologically different solutions exist. One solution consists of two flat disks (Fig. 2); a second, called the catenoid, is topologically equivalent to a cylinder (Fig. 3), and the third is similar to a catenoid with a disk sewn into the center (Fig. 4).

One might think that a catenoid solution exists for any pair of rings regardless of the separation distance. However, this is not the case. Only the two-disk solution exists when

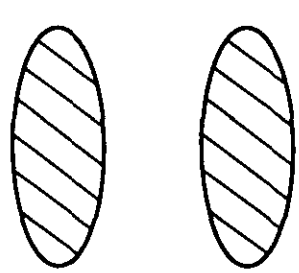

FIG. 2. Two disks.

0021-9991/93 $5.00

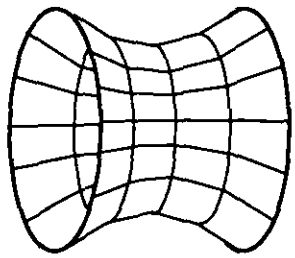

FIG. 3. Euler's catenoid.

the rings are greater than a threshold distance apart. Thus, the topology of a potential solution does not depend solely on the boundary. This is an important point which affects the way minimal surfaces are computed. Obviously, this restricts the versatility of a minimal surfaces algorithm.

There are a number of existing numerical methods for solving Plateau's problem. One class of methods, including those of Concus [3], Hoppe [12], Elcrat and Lancaster [6], and Greenspan [9], represent the surface in non-parametric form. These methods are restricted to surfaces with a functional representation of the form $z = f(x, y)$. A second class of methods, including those of Brakke [1], Wagner [22], Hinata *et al.* [11], and Coppin and Greenspan [4], use a network of connected marker particles to represent parametric surfaces. It is not trivial to know how and when to restructure the connections to simulate a change in topology. In summary, each of these methods are either incapable or require ad hoc intervention by the user to change topology during computing. However, as discussed above, the topology of a solution may not always be predicted. Without the ability to change topology, spurious solutions can result (see [22]). We note that unstable minimal surfaces are difficult to compute, since perturbations due to inexact computer approximations cause a minimum seeking algorithm to flow away from unstable minimums towards a nearby stable minimum.

In this paper, we introduce a new minimal surface solver which is capable of making topology changes naturally.

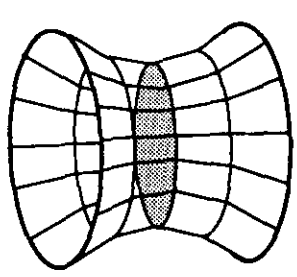

FIG. 4. Catenoid with disk.

2. LEVEL SET CURVATURE MOTION

In a paper by Osher and Sethian [15], a technique for following interfaces propagating with curvature dependent speed was introduced which allows topological changes. This section is a summary of the relevant portions of that paper.

To begin, consider a smooth two-dimensional surface S in R^3. For mean curvature flow, at each point x of S, the velocity of x is in the direction normal to S with magnitude equal to the local mean curvature of S at x. The case of a one-dimensional surface in the plane along with the velocity vectors of the surface is depicted in Fig. 5. More precisely, let S_0 be an initial smooth surface and $x(0) = x_0 \in S_0$. Then the equation of motion for the point $x(t)$ is given by

$$\begin{aligned} \frac{dx}{dt}(t) \cdot n &= \kappa(x(t)) \\ x(0) &= x_0, \end{aligned} \tag{1}$$

where n is the unit normal to S at the point x and $\kappa(x)$ is the mean curvature of S at x. Recall that mean curvature is the average of the two principle curvatures, the sign of which depends on the chosen normal to S. However, changing the sign of the normal vector only changes the sign of the curvature. Therefore, for this equation of motion it is not necessary to specify which normal is used.

This Lagrangian type formulation is well suited to the marker particle type of method. But in [20], Sethian noted that while this formulation for the curvature flow model is relatively easy to implement, it can lead to computational difficulties. He shows that instabilities can result if marker particles get too close together.

A different approach introduced by Osher and Sethian is to model curvature flow in an Eulerian coordinate system. To begin, the surface is represented as a level set of some function Φ: $R^3 \to R$. An equation of motion for Φ equivalent to the Lagrangian formulation may be found by treating every level set $\Phi^{-1}(C)$ as a separate curvature flow problem. This leads to an equation of motion for Φ over the entire domain. For the details, see Osher and Sethian [15] and Sethian [20].

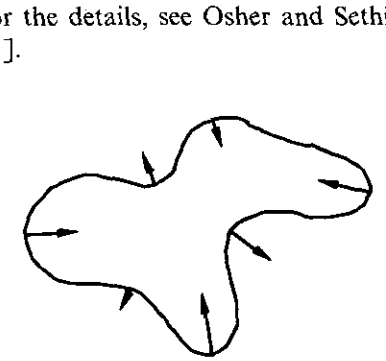

FIG. 5. One-dimensional curvature flow velocity field.

Consider an arbitrary level set $\{x \in R^3 : \Phi(x, t) = C\}$. Implicit differentiation with respect to t in this equation gives

$$\nabla\Phi(x, t) \cdot \frac{dx}{dt}(t) + \Phi_t(x, t) = 0. \tag{2}$$

Note that $-\nabla\Phi/\|\nabla\Phi\|$ is a unit normal to the level set $\Phi^{-1}(C)$, so by combining Eqs. (1) and (2),

$$\begin{aligned}\Phi_t &= \|\nabla\Phi\| \frac{-\nabla\Phi}{\|\nabla\Phi\|} \cdot \frac{dx}{dt}(t) = \|\nabla\Phi\|\, n \cdot \frac{dx}{dt}(t) \\ &= \kappa(x(t))\, \|\nabla\Phi\|. \end{aligned} \tag{3}$$

The mean curvature $\kappa(x(t))$ can be represented solely as a function of Φ and its derivatives by

$$\kappa = \frac{\begin{pmatrix}\Phi_{xx}(\Phi_y^2 + \Phi_z^2) + \Phi_{yy}(\Phi_x^2 + \Phi_z^2) + \Phi_{zz}(\Phi_x^2 + \Phi_y^2) \\ -2\Phi_{xy}\Phi_x\Phi_y - 2\Phi_{yz}\Phi_y\Phi_z - 2\Phi_{xz}\Phi_x\Phi_z\end{pmatrix}}{2(\Phi_x^2 + \Phi_y^2 + \Phi_z^2)^{3/2}}. \tag{4}$$

Therefore, the equation of motion for the surface $\Phi^{-1}(C)$ has been changed into a quasi-linear second-order parabolic partial differential equation which is independent of the level set value C.

An interesting example of a topological change using this method was computed by Sethian [21]. He considered an initial shape of a dumbbell, two large spheres connected by a cylinder. As predicted, the shrinking of the spheres is slower than the shrinking of the cylinder, so the dumbbell eventually breaks in the center. The picture of a slice of the object as it shrinks is depicted in Fig. 6. To see how this relates to the level curves of Φ, Fig. 7 shows the various stages of the surface $\Phi^{-1}(0)$ and the corresponding graph of $\Phi(x, y, 0, t)$. Note how the topology changes when the level set $\Phi^{-1}(0)$ contains the center critical point of Φ, where $\nabla\Phi = 0$. In his computation it should be noted that Sethian used a staggered grid around the center point where $\nabla\Phi = 0$ to avoid the zero denominator in Eq. (4).

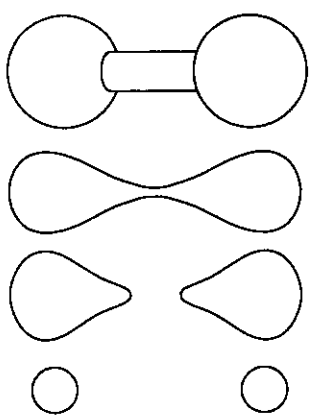

FIG. 6. Motion of a dumbbell under mean curvature.

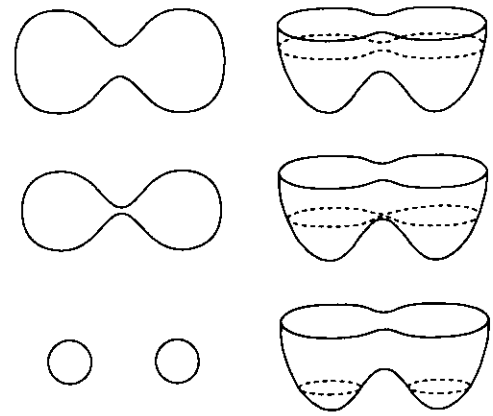

FIG. 7. Breaking dumbbell vs. $w = \Phi(x, y, 0, t)$.

Thus, this new level set approach to curvature flow problems provides a way to change topology naturally without the need for ad hoc decisions. This property can be exploited to create a new minimal surface algorithm.

Osher and Sethian in [15, 20] developed a numerical method based upon this Eulerian formulation. In those papers a more general curvature-dependent speed function was considered. For the case of speed equal to mean curvature (see [15]), any type of time derivative method, for example, Euler's method, Runge–Kutta's method, or an implicit method, can be employed. For the space derivatives, central differences are used to approximate the curvature terms. Thus, for Euler's method, the numerical scheme for modelling curvature flow reduces to

$$\Phi_{ijk}^{n+1} := \Phi_{ijk}^n + \Delta t * K(\Phi^n),$$

where $K(\Phi^n)$ is a finite difference approximation to the curvature from Eq. (4). In this case, central differences for all derivatives of Φ in the interior of the domain. At the boundary of the domain one-sided finite differences are used.

In order to construct $\Phi(x, 0)$ for a given initial surface S, the signed distance function from the initial surface is used. For the signed distance, the magnitude of Φ is given by $|\Phi(x, 0)| = \text{dist}(x, S)$. Given an orientation for the normal n of S, the sign of Φ is determined by the requirement that $d\Phi/dn|_S = 1$. As noted above, for mean curvature flow it does not matter which side of the surface has positive values of Φ.

3. COMPUTATION OF MINIMAL SURFACES

In this section we develop a new minimal surface algorithm using the level set curvature flow model discussed

above. The basic idea is to compute minimal surfaces by attaching a surface to a given boundary and letting it move according to its mean curvature.

Since we are primarily concerned with making the numerator of the curvature formula tend to zero, the minimal surface algorithm presented here leaves out the denominator for the curvature flow. This is permissible because the denominator is only a scaling factor for the distance between level sets. The reinitialization process described below ensures that this scaling factor remains uniform everywhere on the surface except at points where $\nabla\Phi = 0$. When $\nabla\Phi = 0$, the numerator of the curvature is also zero, so regularizing the curvature formula results in the same flow as when just the numerator of curvature is used. Unless otherwise noted, curvature flow in this section refers to the curvature formula without the denominator.

3.1. *The Basic Algorithm*

If the level set approach is applied to the minimal surface problem with boundary, there is no reason to expect the surface to remain attached to the boundary. We must anchor the surface on the boundary. Thus, a first attempt at a minimal surface algorithm based on curvature flow might look like this:

ALGORITHM 1.

Step 1. Move surface according to curvature flow.

Step 2. Reattach surface to boundary.

Step 3. Go to step 1.

The algorithm will use the level set formulation of curvature flow to follow the surface $\Phi^{-1}(0)$. In this formulation, the boundary contour divides $\Phi^{-1}(0)$ into two disjoint pieces, $\Phi^{-1}(0) = I \cup E$, where E is defined to be the *exterior set* and the compact *interior set* defined by I also contains the boundary.

Consider a boundary contour Γ consisting of a ring of radius R in the xy-plane. One might take the initial guess of a solution as a hemisphere S of radius R attached to the ring. To realize S as a level set of some function Φ, the surface S must be continued out to the edges of the domain of Φ. One way to continue the surface is to let E be the xy-plane minus the disk bounded by Γ, so that $\Phi^{-1}(0)$ would appear as in Fig. 8. Now Φ can be constructed by choosing the sign of Φ to be, say, negative below the surface and positive above. Clearly, the portion of the surface that is of interest is the set I. The set E is only needed to construct Φ.

The process of reattaching the surface to the boundary in this numerical method follows from the idea that Φ must only be altered locally at the boundary so that $\Phi \equiv 0$ at the boundary. Doing this involves answering two questions. First, how does one model a one-dimensional contour on a three-dimensional rectangular grid? Second, how can Φ be altered locally so that $\Phi \equiv 0$ at the boundary?

In general, the boundary conditions will be represented as a vector equation of the form $v_{\text{dep}} \leftarrow Av_{\text{ind}}$, where

$$v_{\text{dep}} = \begin{bmatrix} \Phi(g_{d,1}) \\ \Phi(g_{d,2}) \\ \vdots \\ \Phi(g_{d,m}) \end{bmatrix}, \qquad v_{\text{ind}} = \begin{bmatrix} \Phi(g_{i,1}) \\ \Phi(g_{i,2}) \\ \vdots \\ \Phi(g_{i,n}) \end{bmatrix},$$

$g_{d,j}$, $g_{i,j}$ are distinct grid points, and A is an $m \times n$ matrix. The vector v_{ind} is the *independent vector* and consists of the values of Φ at all the *independent grid points* $g_{i,j}$. The value of Φ at independent grid points is entirely determined by curvature flow. The vector v_{dep} is the *dependent vector* and consists of the values of Φ at all the *dependent grid points* $g_{d,j}$. At dependent grid points, the value of $\Phi(g_{d,j})$ is determined entirely by Av_{ind}. Since the matrix A only depends on the fixed boundary Γ, the coefficients of A are constant for all time. Let the entries of the matrix A be called the *dependency coefficients*. The computation of the dependency coefficients can be broken down into a linear combination of smaller matrix blocks of size 1×1 and 2×2.

For the remainder of this section let $\Gamma = \{\gamma(s) \in R^3 : s \in [0, P]\}$, where $\gamma(s)$ is continuous with $\gamma(0) = \gamma(P)$, be a single boundary contour.

As $\gamma(s)$ traverses through the grid, there are three ways it can interact with the grid. Define a *zero-point* to be a grid point g such that $g \in \Gamma$. Define an *edge* to be a line segment L connecting two adjacent grid points such that $L \cap \Gamma \neq \emptyset$. Define a *pane* to be a rectangular region R bounded by four connecting coplanar edges such that $R \cap \Gamma \neq \emptyset$. An example of each is depicted in Fig. 9.

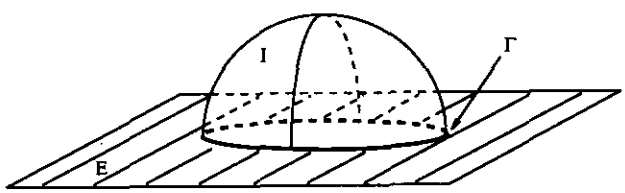

FIG. 8. $\Phi^{-1}(0) = E \cup I$

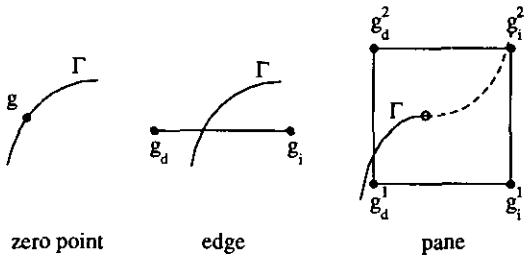

FIG. 9. Interaction between a curve and a grid.

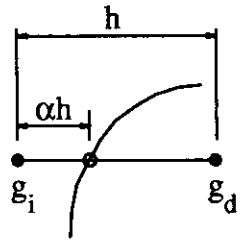

FIG. 10. Dependency calculation of an edge.

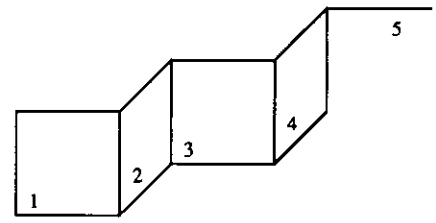

FIG. 12. A sample chain.

The boundary Γ is discretized into a piecewise linear path with vertices consisting of all the intersections of Γ with zero-points, edges, and panes in the grid. Keeping Γ fixed then means fixing those vertices with respect to the grid. The boundary conditions needed for fixing the vertices require that independent and dependent grid points are designated and that the resulting dependency coefficients are calculated.

For a zero-point g, the point itself is defined to be dependent with no dependency coefficient. Since g is always on the boundary and $\Phi = 0$ on the boundary, then $\Phi(g) \leftarrow 0$.

For an edge, one end point must be designated independent, the other dependent. In general, the point closer to the interior set I is chosen as independent. In this case, $\Phi(g_d) \leftarrow C\Phi(g_i)$, where $C < 0$ is a constant for all time. To calculate C, suppose the intersection point of Γ and the edge is given by $\alpha g_d + (1-\alpha)\, g_i$ with $\alpha \in (0, 1)$. Linear interpolation implies that $C = (\alpha - 1)/\alpha$.

For a pane, two adjacent points must be chosen to be independent, the other two are then dependent. Again, the two points sharing a common edge nearest to the set I are chosen as independent. In this case the two points will straddle the intersection of I with the pane. The dependency relation is then a 2×2 matrix equation. Let the intersection point be given by $(1-\alpha-\beta)\, g_{i,1} + \alpha g_{i,2} + \beta g_{d,1}$. Then the dependency relation is given by

$$\begin{bmatrix} \Phi(g_{d,1}) \\ \Phi(g_{d,2}) \end{bmatrix} \leftarrow \frac{1}{\beta} \begin{bmatrix} \alpha+\beta-1 & -\alpha \\ \alpha-1 & \beta-\alpha \end{bmatrix} \begin{bmatrix} \Phi(g_{i,1}) \\ \Phi(g_{i,2}) \end{bmatrix}. \tag{5}$$

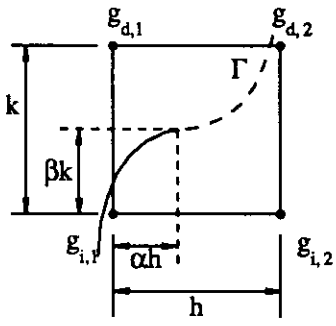

FIG. 11. Dependency calculation of a pane.

Unfortunately, the choices made for edges and panes are not isolated (see Figs. 10 and 11). There may be grid points which belong to more than one pane or edge. Define a *link* to be either a pane or an edge. Define a *chain* to be an ordered list of links with any two consecutive links sharing at least one common grid point. A simple example of a chain is depicted in Fig. 12. As a point $\gamma(s)$ traverses the grid, a string of connected panes and edges are intersected.

In general, the complete boundary conditions for Γ are not one large chain, but instead are made up of any number of disjoint chains. These chains can be separated in any number of ways; for example, Γ may intersect opposite facing panes, a pane opposite a parallel edge, two opposite edges, etc. Some chain separators are shown in Fig. 13.

Define a chain to be *consistent* if every grid point contained in the chain is labelled either independent or dependent, but not both. The goal of choosing independent and dependent points is to construct consistent chains. In the chain shown in Fig. 12 there are exactly the four consistent labellings shown in Fig. 14.

Unfortunately, there exist chains which do not permit any consistent labellings. Such chains may arise when the grid is too coarse to adequately model a given boundary. To get around this problem, moving the boundary with respect to the grid by some small amount ε may help. If nudging the boundary does not help, then only refining the mesh will eliminate such chains.

Note that the intersection points between Γ and the edges and panes of the grid are strictly local events; i.e., the location of the intersection of Γ and pane one in Fig. 12 should have no influence over the location of the intersection of Γ in pane four. However, an inconsistent chain can lead to dependencies stretching all along the length of a chain.

The structure of the chains also affects the calculation of the dependency coefficients. The number of dependent and

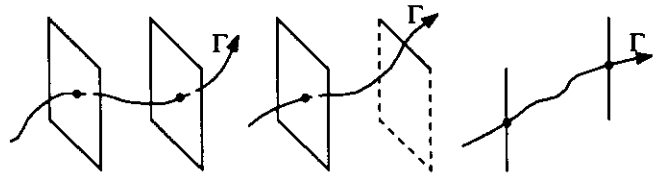

FIG. 13. Example chain separators.

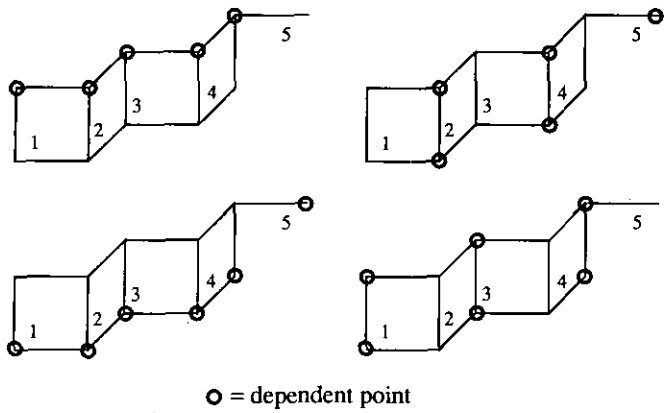

FIG. 14. Four consistent labellings.

independent points within a chain may not be the same, so the boundary conditions may be an overdetermined system of equations. For example, consider the labelled two pane chain in Fig. 15. The dependency relations for such a system are given by

$$\begin{bmatrix}\Phi(g_{d,1})\\ \Phi(g_{d,2})\end{bmatrix} \leftarrow A_1 \begin{bmatrix}\Phi(g_{i,1})\\ \Phi(g_{i,2})\end{bmatrix},$$

$$\begin{bmatrix}\Phi(g_{d,1})\\ \Phi(g_{d,2})\end{bmatrix} \leftarrow A_2 \begin{bmatrix}\Phi(g_{i,3})\\ \Phi(g_{i,4})\end{bmatrix}.$$

The values $\Phi(g_{d,1})$ and $\Phi(g_{d,2})$ are overdetermined. One option for solving the overdetermined system is to average the values $\Phi(g_{d,1})$ and $\Phi(g_{d,2})$; thus

$$\begin{bmatrix}\Phi(g_{d,1})\\ \Phi(g_{d,2})\end{bmatrix} \leftarrow \frac{1}{2}\left(A_1 \begin{bmatrix}\Phi(g_{i,1})\\ \Phi(g_{i,2})\end{bmatrix} + A_2 \begin{bmatrix}\Phi(g_{i,3})\\ \Phi(g_{i,4})\end{bmatrix}\right).$$

Note that if the surface is a plane, then the averaging solution is correct. As the mesh is refined the surface locally becomes more like a plane. Therefore, as the mesh is refined, the error in this averaging process decreases.

The construction of the numerical boundary conditions is not strictly unique, but there is an optimal set of boundary conditions given an interior set I. As described above, the optimal boundary conditions are those for which the dependent points are furthest from I. If given boundary curve Γ, non-optimal boundary conditions is equivalent to solving the minimal surface problem for a perturbed boundary curve Γ'. The amount of the perturbation is no greater than the mesh size of the grid. In practice, this has always resulted in a perturbed solution I' which is close enough to the desired solution to be able to compute the optimal boundary conditions. Thus, finding the numerical boundary conditions for a given boundary curve Γ is a two-step process. First, an initial guess at the correct boundary conditions is made and the surface with those boundary conditions is generated. The second step of the boundary generation process is to readjust the boundary using the computed surface as an approximation to a solution. Once the boundary is properly adjusted, a better solution can be computed with the new boundary using the previously computed surface as the new initial surface.

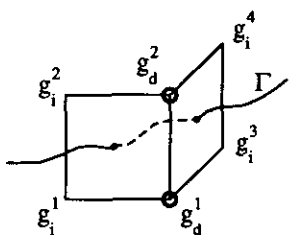

FIG. 15. Overdetermined labelling.

3.2. *Example Boundary Construction: A Circle*

To illustrate, an example boundary is computed for a circle. Consider a circle of radius $3\frac{1}{2}$ parameterized as

$$x(\theta) = \tfrac{3}{2}, \qquad y(\theta) = 5 + \tfrac{7}{2}\sin(\theta),$$
$$z(\theta) = \tfrac{9}{2} + \tfrac{7}{2}\cos(\theta)$$

Define the grid to have dimensions $2 \times 9 \times 8$ centered about the origin with uniform space step $h = 1$. This is an exceedingly large space step, but is used here to simplify the arithmetic. For now, suppose that the interior set I is in the positive x direction. This would be the case if the initial surface is to be like a cylinder, for example, and this circle is on one end. The relationship between the boundary curve and the grid are shown in the two views in Fig. 16.

In this example there are 14 chains, four of which are shown in Fig. 17. By symmetry, it is only necessary in this example to compute the boundary conditions for the first four chains.

Chain 1. This chain consists of only one link, an edge connecting grid points $g_{1,5,8}$ and $g_{2,5,8}$. The inter-

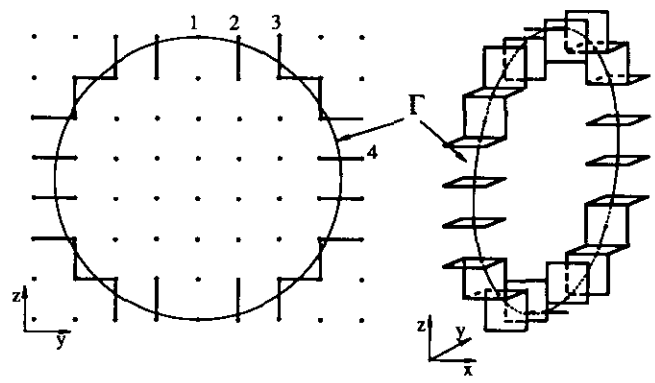

FIG. 16. Example of a grid with a boundary curve.

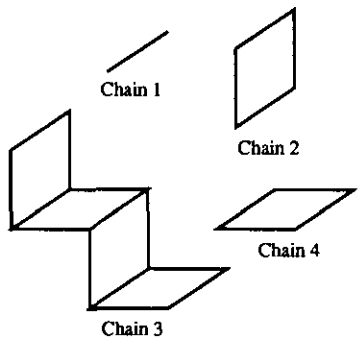

FIG. 17. Example chains.

section occurs when $\theta = 0$ which corresponds to the point $\frac{1}{2}g_{1,5,8} + \frac{1}{2}g_{2,5,8}$. Since the set I is in the direction of positive x, then the grid point $g_{2,5,8}$ should be the one labelled independent. Using the formula in Eq. (5) with $\alpha = \frac{1}{2}$ gives the equation

$$\Phi(g_{1,5,8}) \leftarrow -\Phi(g_{2,5,8}).$$

Chain 2. This chain also consists of only one link, a pane connecting the grid points $g_{1,6,7}$, $g_{1,6,8}$, $g_{2,6,7}$, and $g_{2,6,8}$. The intersection point occurs at the point $(\frac{3}{2}, 6, (9+3\sqrt{5})/2)$. With respect to the grid, this corresponds to the point $(1-\alpha-\beta)g_{2,6,7} + \alpha g_{2,6,8} + \beta g_{1,6,7}$, where $\alpha = (3\sqrt{5}-5)/2$ and $\beta = \frac{1}{2}$. As in the previous chain, the independent points should be $g_{2,6,7}$ and $g_{2,6,8}$. Plugging this into Eq. (5) gives

$$\begin{bmatrix}\Phi(g_{1,6,7})\\ \Phi(g_{1,6,8})\end{bmatrix} \leftarrow \begin{bmatrix}3\sqrt{5}-6 & 5-3\sqrt{5}\\ 3\sqrt{5}-7 & 6-3\sqrt{5}\end{bmatrix}\begin{bmatrix}\Phi(g_{2,6,7})\\ \Phi(g_{2,6,8})\end{bmatrix}.$$

Chain 3. This chain is made up of four connected panes. Choosing dependencies is similar to the previous one-pane segment. In this case, the dependent points should be the grid points $g_{1,j,k}$ and the independent points should be $g_{2,j,k}$. It is easy to see that this chain is then consistent; i.e., no grid points are labelled both dependent and independent. Once the dependencies are established, then the calculation of the dependency matrices proceeds as in Chain 2 to obtain the four matrix equations

$$\begin{bmatrix}\Phi(g_{1,7,7})\\ \Phi(g_{1,7,8})\end{bmatrix} \leftarrow A\begin{bmatrix}\Phi(g_{2,7,7})\\ \Phi(g_{2,7,8})\end{bmatrix},$$

$$\begin{bmatrix}\Phi(g_{1,7,7})\\ \Phi(g_{1,8,7})\end{bmatrix} \leftarrow B\begin{bmatrix}\Phi(g_{2,7,7})\\ \Phi(g_{2,8,7})\end{bmatrix},$$

$$\begin{bmatrix}\Phi(g_{1,8,6})\\ \Phi(g_{1,8,7})\end{bmatrix} \leftarrow C\begin{bmatrix}\Phi(g_{2,8,6})\\ \Phi(g_{2,8,7})\end{bmatrix},$$

$$\begin{bmatrix}\Phi(g_{1,8,6})\\ \Phi(g_{1,9,6})\end{bmatrix} \leftarrow D\begin{bmatrix}\Phi(g_{2,8,6})\\ \Phi(g_{2,9,6})\end{bmatrix}.$$

Let A have entries a_{ij}, and similarly for B, C, and D. Then the four matrix equations are combined into the one matrix equation by averaging to obtain

$$\begin{bmatrix}\Phi(g_{1,7,8})\\ \Phi(g_{1,7,7})\\ \Phi(g_{1,8,7})\\ \Phi(g_{1,8,6})\\ \Phi(g_{1,9,6})\end{bmatrix} \leftarrow \begin{bmatrix} a_{22} & a_{21} & 0 & 0 & 0\\ \frac{a_{12}}{2} & \frac{a_{11}+b_{11}}{2} & \frac{b_{12}}{2} & 0 & 0\\ 0 & \frac{b_{21}}{2} & \frac{b_{22}+c_{22}}{2} & \frac{c_{21}}{2} & 0\\ 0 & 0 & \frac{c_{12}}{2} & \frac{c_{11}+d_{11}}{2} & \frac{d_{12}}{2}\\ 0 & 0 & 0 & d_{21} & d_{22}\end{bmatrix} \times \begin{bmatrix}\Phi(g_{2,7,8})\\ \Phi(g_{2,7,7})\\ \Phi(g_{2,8,7})\\ \Phi(g_{2,8,6})\\ \Phi(g_{2,9,6})\end{bmatrix} \tag{6}$$

Chain 4. The process for this single pane chain is identical to that of Chain 2.

The remainder of the boundary can be constructed easily by using symmetry, or it can be done the same way as above.

3.3. *Reininitialization and the "Tentpole Phenomenon"*

Using curvature flow with boundary conditions to anchor the surface to the boundary contour produces a very simple first attempt at an algorithm for computing minimal surfaces.

Unfortunately, the algorithm does not work because of inaccuracy at the boundary. To resolve the inaccuracy, some investigation into the theory of curvature flow is necessary. For mean curvature flow without boundaries using the level set representation, Evans and Spruck [7] proved that the level sets of the function Φ do not change distance relative to each other as time progresses. When boundaries are introduced, however, this is not true. Instead, a structure resembling a tentpole can appear as in Fig. 18.

Unless the surface $S = \Phi^{-1}(0)$ is extended so that all the level sets near the boundary have zero mean curvature, the level sets on one side of S will collapse together onto the boundary, while on the other side they separate and flow away from the boundary. In the wake of the boundary, as

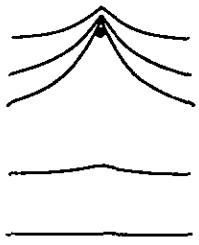

FIG. 18. The tentpole phenomenon.

the level sets flow downstream, eventually catastrophic round-off affects the computation of the curvature near the boundary and instability ensues.

To avoid the tentpole phenomenon the surfaces are reinitialized. At regular time intervals, the surface $\Phi^{-1}(0)$ is located and the signed distance from $\Phi^{-1}(0)$ to each point is computed. The formula for reinitialization can be expressed as

$$\Phi(x) \leftarrow \text{sign}(\Phi(x))\,\text{dist}(x, \Phi^{-1}(0)). \tag{7}$$

The process of reinitialization effectively moves the nonzero level sets so that they are equally spaced as they would be for flow without boundary. The zero level set remains fixed.

Adding reinitialization to Algorithm 1 produces the final algorithm for computing minimal surfaces:

ALGORITHM 2.

Step 1. Move by curvature flow

Step 2. Reattach the surface to the boundary

Step 3. Reintialize Φ

Step 4. Go to step one.

3.4. *Reducing Computing Costs*

Considered as separate pieces, the steps for computing minimal surfaces given above can be computationally expensive. In the simplest case, one time step in curvature flow is an $O(n^3)$ operation, reattaching the boundary is $O(n)$, and reinitializing the surface is $O(n^6)$. To see that reinitialization is of order $O(n^6)$, note that it is an $O(n^3)$ operation to locate a level set on a grid and an additional $O(n^3)$ operation to compute the distance from a point on

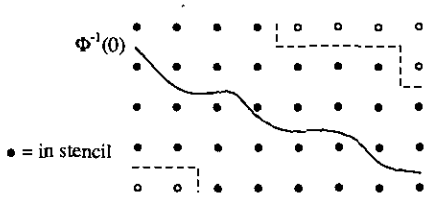

FIG. 19. Two-dimensional stencil diagram.

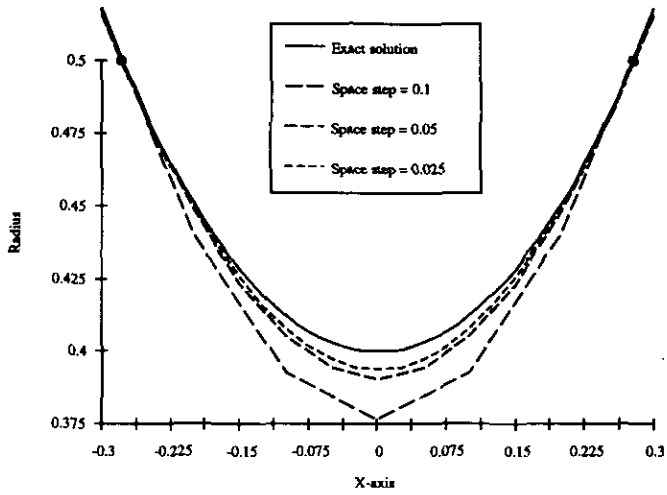

FIG. 20. Convergence of solutions under grid refinement.

the surface to each point on the grid. Even for relatively small values of n, an $O(n^6)$ order computation done every time step is for all practical purposes too costly.

By combining the three steps and making one observation, the overall computing cost can be decreased dramati-

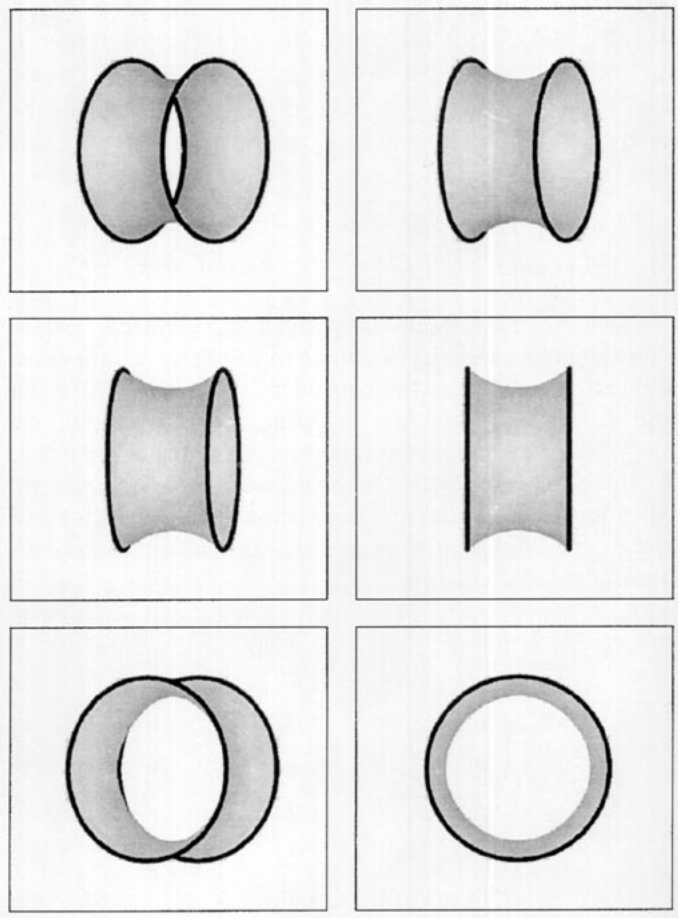

FIG. 21. Euler's catenoid surface.

TABLE I

Computed Error Table

Grid size	L^1	L^2	L^∞
$9\times17\times17$	8.64E − 02	1.63E − 03	2.37E − 02
$17\times33\times33$	6.13E − 02	4.27 − 04	9.33E − 03
$26\times46\times46$	7.42E − 02	3.34E − 04	6.13E − 03

cally. To begin, note that with the introduction of reinitialization, the values of Φ outside of a neighborhood of $\Phi^{-1}(0)$ are extraneous and have no effect on the motion of the set $\Phi^{-1}(0)$. The only contribution of points outside the neighborhood is to hold a sign for the use of locating the zero level set. This observation leads to the conclusion that the only points that need to be reinitialized are those inside of the neighborhood of $\Phi^{-1}(0)$ that are used in computing the motion for $\Phi^{-1}(0)$.

Let two gridpoints $g_{i,j,k}$ and $g_{i',j',k'}$ be *neighbors* if $\max\{|i-i'|, |j-j'|, |k-k'|\} \leqslant 1$. Define the *stencil* of the surface to be those points in a neighborhood of $\Phi^{-1}(0)$.

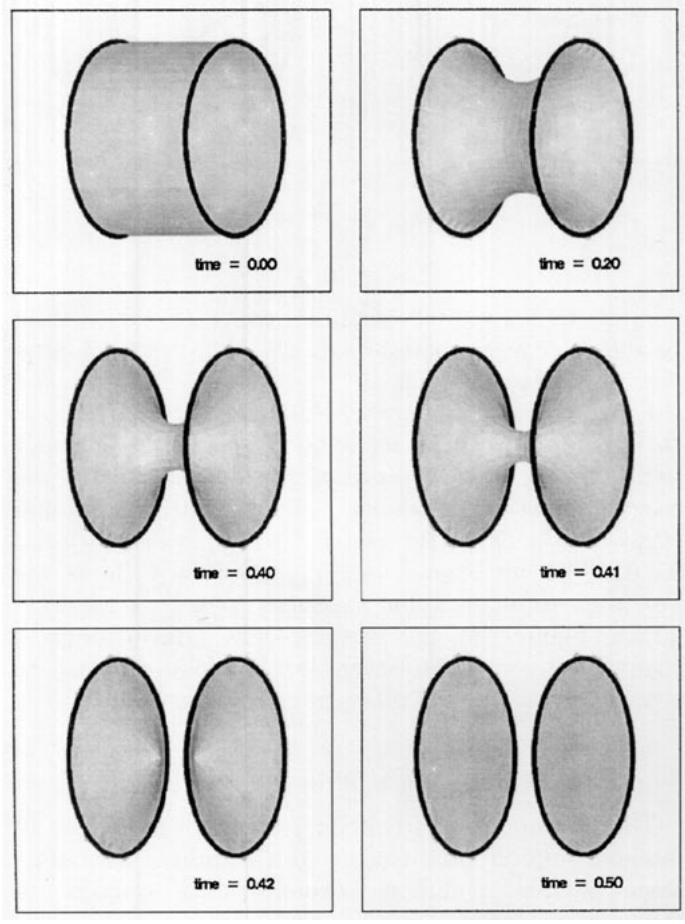

FIG. 22. Splitting catenoid evolution.

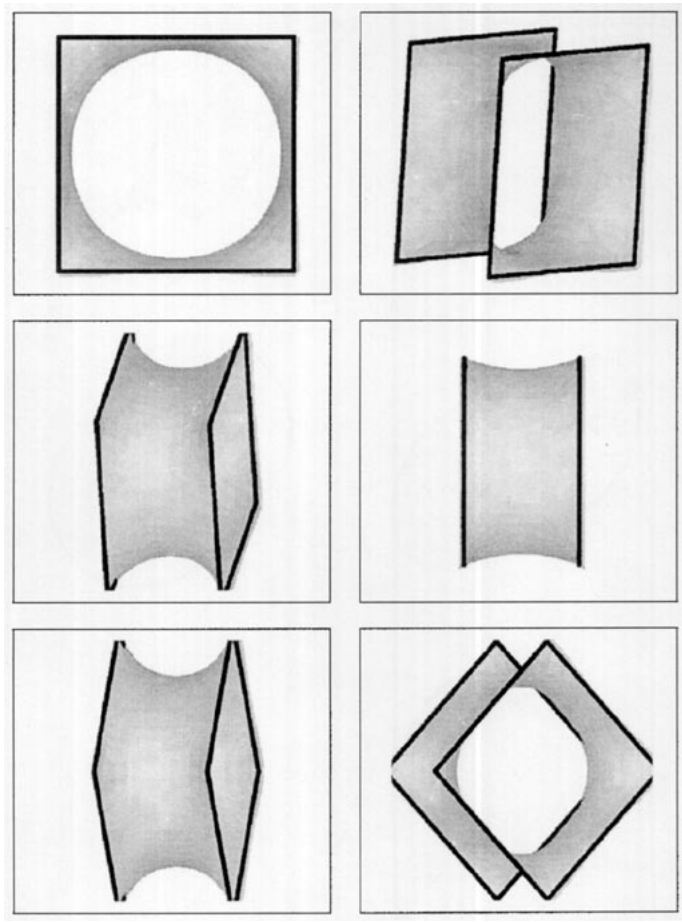

FIG. 23. Square catenoid surface.

More specifically, a point g is in the interior of the stencil if it has a neighbor point g', for which $\Phi(g)\,\Phi(g') \leqslant 0$. A point g is in the stencil if it has a neighbor g' in the interior of the stencil. In other words, a point g is in the stencil if it is next to the surface or is one of the 26 grid points surrounding a point next to the surface. A diagram of the two-dimensional case is shown in Fig. 19.

By restricting reinitialization to the stencil of the surface, then the computational cost of reinitialization drops from $O(n^6)$ to $O(n^3)$. To see this reduction note that if restricted to the stencil, the distance need only be computed for a fixed size neighborhood of each point, since only points near the surface are going to be reinitialized. Thus, the cost of reinitialization reduces to finding the surface and then computing for each point near the surface the distance to a fixed number of points (26) independent of n. Therefore, the total cost of reinitialization decreases to $O(n^3)$.

Furthermore, since the values of Φ outside of the stencil are only sign holders, it is not necessary to move Φ outside of the stencil. Thus, during reinitialization, a list of the points in the stencil is created and passed to the curvature

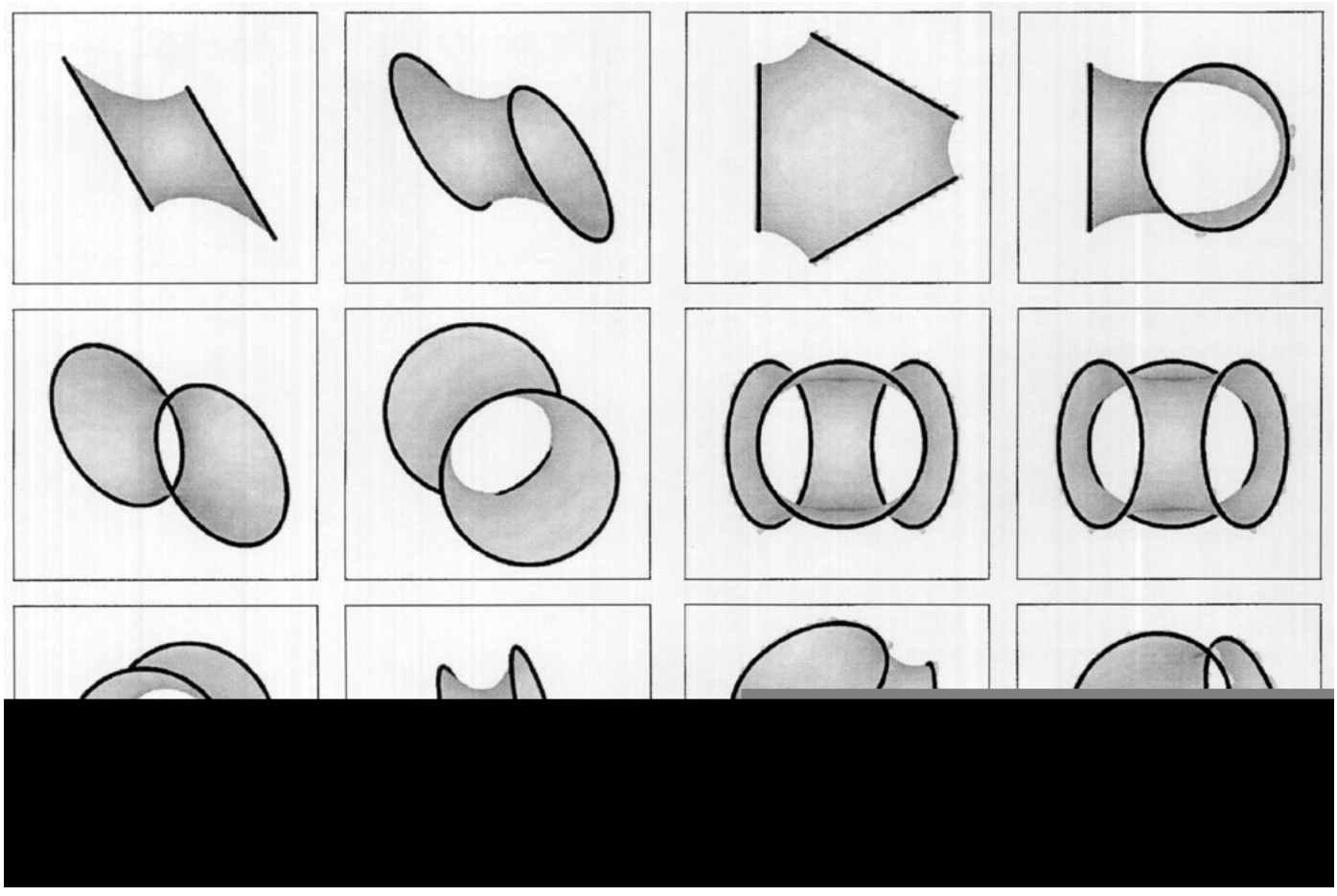

FIG. 24. Offset circles surface.

FIG. 25. Three circles surface.

flow step. The curvature flow subroutine computes the curvature motion for all points in the *interior* of the stencil. This allows the curvature flow step cost to drop from $O(n^3)$ for computing Φ_t everywhere, to $O(n^2)$ for computing Φ_t only within the stencil.

One final side benefit is gained by restricting computation to the stencil. It now no longer matters what type of boundary conditions are used for the boundary of the grid. Since computation is only done within the stencil, only the intersection of the stencil with the grid boundary can influence the motion of Φ. However, perturbations introduced by the grid boundary are stopped entirely by the contour boundary before they can affect the interior portion I. Thus, it is equally acceptable to use one-sided differences at the grid boundary or simply to assume $\Phi_t = 0$ on the grid boundary.

4. NUMERICAL RESULTS

The algorithm presented in the previous chapter was designed to meet the following goals: it should produce accurate results with at least linear converge and it should be able to change topology naturally without special intervention by the user.

From tests against known solutions, the convergence of the algorithm to the exact solution appears to be nearly linear. In Fig. 20 a sequence of grid refinements and the exact solution for the radius of Euler's catenoid solution $r(x) = a \cosh(x/a)$ on the interval $[-b, b]$, with $a = 0.4$ and $b = 0.277259$ are shown (see Fig. 1). Table I shows the absolute error in the radius measured in the L^1, L^2, and L^∞ norms. Figures 22, 26, and 27 show how topological changes can occur. However, the topology can also be complex if a solution of that type exists as in Fig. 34.

4.1. *Examples of Computed Surfaces*

The following pages contain pictures of a number of minimal surfaces computed by this algorithm as well as some sample evolutions demonstrating changes of topology.

In Fig. 21, the computed solution of Euler's catenoid

surface is shown. The rings are of radius 0.5 and positioned at $x = \pm 0.277259$. The radius at the center should be approximately 0.4. The mesh size is $27 \times 47 \times 47$ with space step 0.025 in all directions. The initial surface consisted of a cylinder of radius 0.5.

In Fig. 22 is shown what happens if the initial surface is chosen to be a cylinder as in Fig. 21, but the rings are too far apart for a catenoid solution to exist. In this case, the topology changes so that, instead of a cylinder type surface, two disks are found as the solution. For this surface, the rings have radius 0.5 and are positioned at $x = \pm 0.345$, and the mesh size is $41 \times 41 \times 41$ with space step 0.05.

In Fig. 23, the computed solution of a catenoid type of surface with square ends is shown from different angles. The squares have side length 1.0 and are positioned at $x = \pm 0.275$. The mesh size is $41 \times 41 \times 41$ with space step 0.05. The initial surface was a square tube of side length 1.0.

In Fig. 24, a cylinder type of surface is shown, where the circles are parallel, but with offset centers. The circles are of radius 0.5 with the centers located at $\pm(0.2625, 0.0, 0.25)$ parallel to the yz-plane. The mesh has dimensions $25 \times 45 \times 67$ with space step 0.025 in all directions. The initial surface was a cylinder with oval cross section and principle radii $\frac{1}{2}$ and $\frac{21}{58}$.

In Fig. 25, the boundary consists of three circles with centers equally spaced on a base circle of radius $\frac{2}{3}$ on the xy-plane. The circles each have radius $\frac{2}{3}$. The mesh has dimensions $41 \times 41 \times 41$ with space step 0.05. The initial surface was three half-cylinders joined in the center.

In Fig. 26, the boundary consists of three circles with centers equally spaced on a base circle of radius 0.755 on the xy-plane. The circles each have radius 0.51. The mesh has dimensions $41 \times 51 \times 25$ with space step 0.05. The initial surface was three half-cylinders joined in the center. In this case the rings are too far apart to have a connected minimal surface, hence the surface breaks into three disks.

In Fig. 27, the boundary consists of six squares of side length $\frac{1}{2}$, centered on the coordinate axes. On the x-axis the squares are located at ± 0.375, on the y-axis at ± 0.775 and on the z-axis at ± 1.275. The different distances were chosen to cause the surface to break at three different times. The mesh has dimensions $25 \times 41 \times 61$ with uniform space step

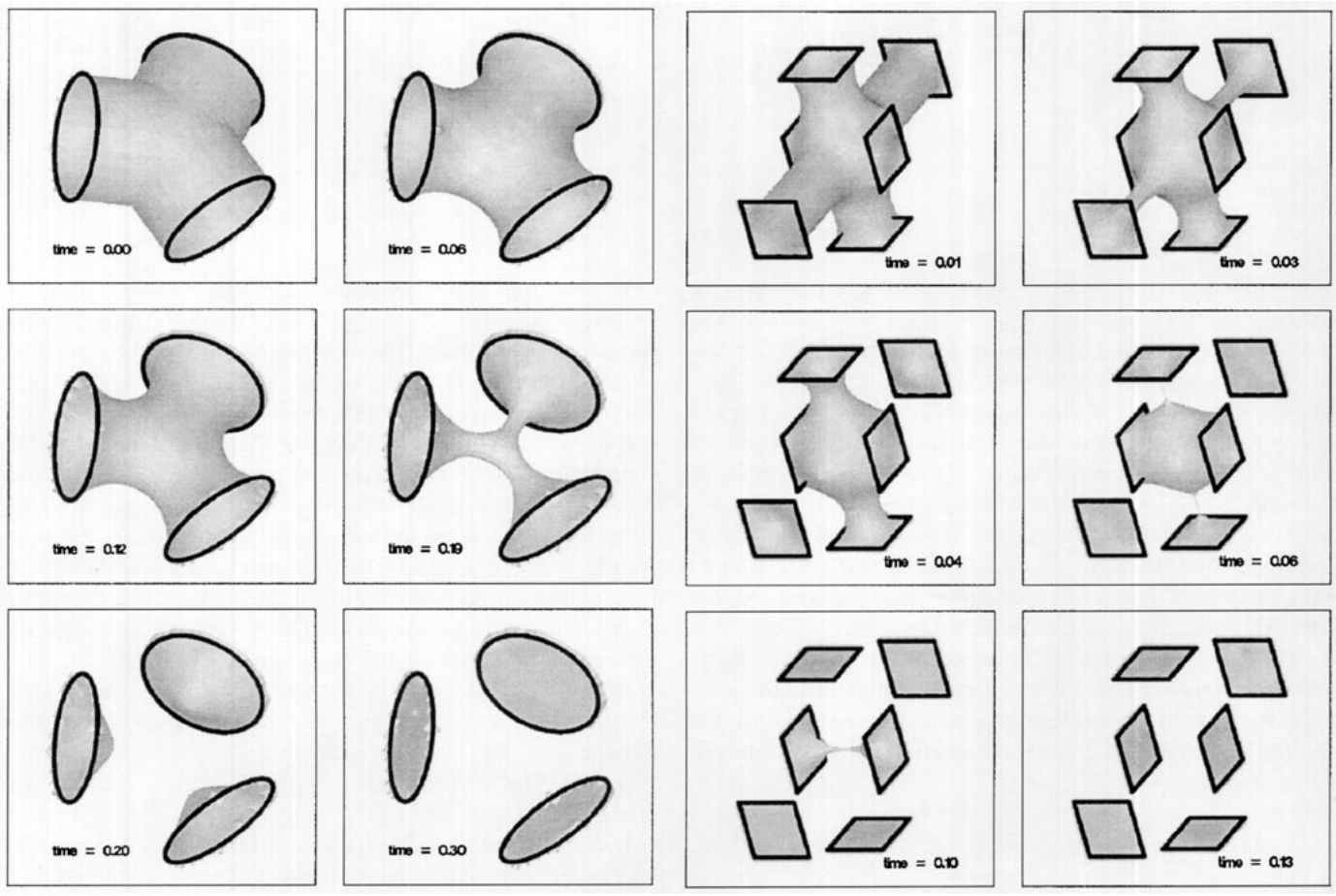

FIG. 26. Three rings splitting evolution.

FIG. 27. Six squares splitting evolution.

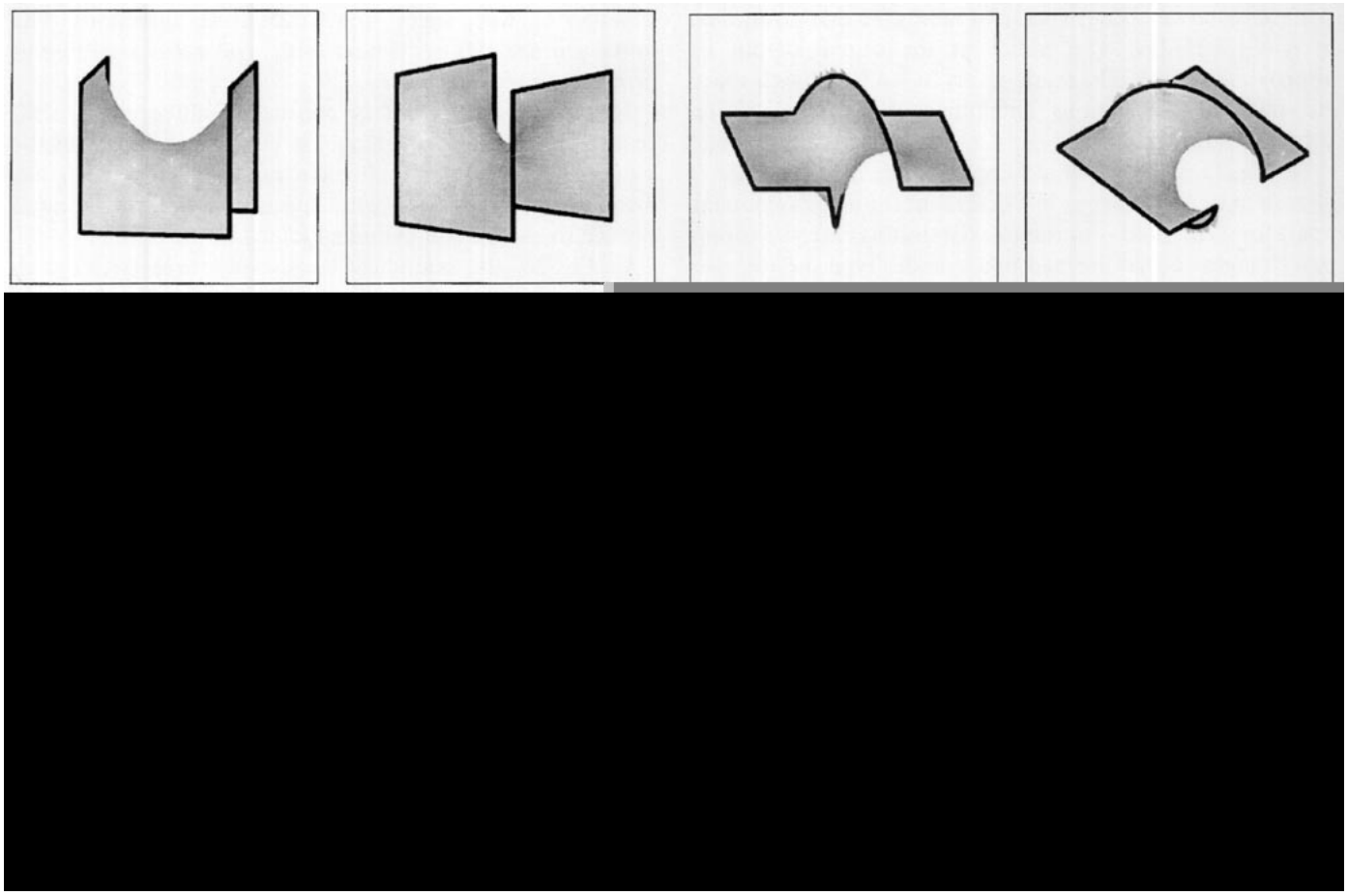

FIG. 28. Square Sherck surface.

FIG. 29. Twisted rectangular strip surface.

0.05. The initial surface was the union of three cylinders with square cross section.

In Fig. 28, the surface computed is known as Sherck's surface. The boundary consists of eight line segments of unit length. The mesh has dimensions $42 \times 42 \times 42$ with space step $\frac{2}{41}$ in all directions. The intial surface consisted of the two parallel half-planes $\{(x, y, z): y = \pm 0.5,\ z < 0.5\}$ and the strip $\{(x, y, z): -0.5 \leqslant y \leqslant 0.5,\ z = 0.5\}$.

In Fig. 29, the boundary is formed by the edge of a twisted rectangle. The width of the rectangle is, and the flat ends have length 0.5. The center twisted portion has length 0.5 as well. The grid dimensions are $42 \times 42 \times 42$ with space step $\frac{2}{41}$. The initial surface was the twisted strip, from which the boundary was derived, extended out to the edge of the grid.

In Fig. 30, the boundary is an oval mapped onto the surface of a cylinder. The exact equation for the boundary curve is given by

$$\gamma(s) = (\tfrac{1}{2}\cos(s) + \tfrac{1}{40}, \sin(\tfrac{3}{2}\sin(s)), \cos(\tfrac{3}{2}\sin(s))).$$

The dimensions of the grid are $46 \times 46 \times 46$ with space step $\frac{2}{45}$. The initial surface was a cylinder of radius $\frac{1}{2}$.

In Fig. 31, the boundary consists of two parallel "bowties" located at $x = \pm 0.1375$, rotated by $\pi/6$ with respect to each other. The bowties consist of two opposite quarter arcs of radius $\frac{1}{2}$, along with the connecting diameter lines. The grid size is $25 \times 49 \times 49$ with space step 0.025. This surface demonstrates the ability of the algorithm to handle singularities in the boundary contour. Two more examples of contour singularities follow.

In Fig. 32, the boundary consists of a square of side length $\frac{1}{2}$ on one end and two squares of diagonal length $\frac{1}{2}$ rotated by $\pi/4$ and offset so that the corners touch on the x-axis. The two ends are located at $x = \pm 0.1375$. The mesh dimensions are $25 \times 25 \times 49$ with space step 0.025.

In Fig. 33, the boundary consists of a square of side length $\frac{1}{2}$ on one end and a "pinwheel" of the same dimensions. The two ends are parallel centered at $x = \pm 0.1375$. The grid dimensions are $25 \times 45 \times 45$ with space step 0.025.

In Fig. 34, the boundary consists of three parallel squares. The outer two squares have side length one and the middle square has side length two. The mesh dimensions are $25 \times 47 \times 47$ with space step 0.025. This surface is very

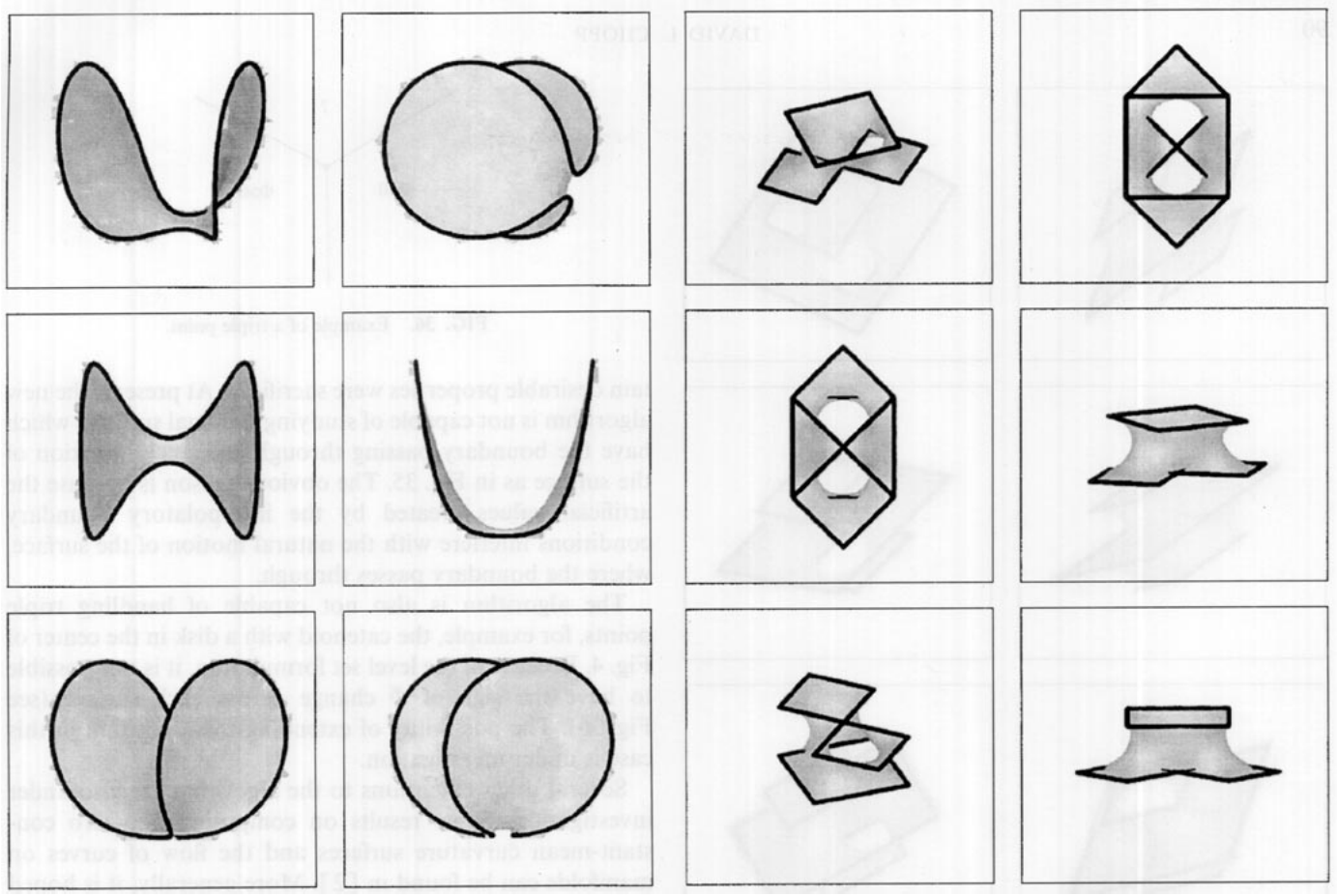

FIG. 31. Twisted bowtie surface.

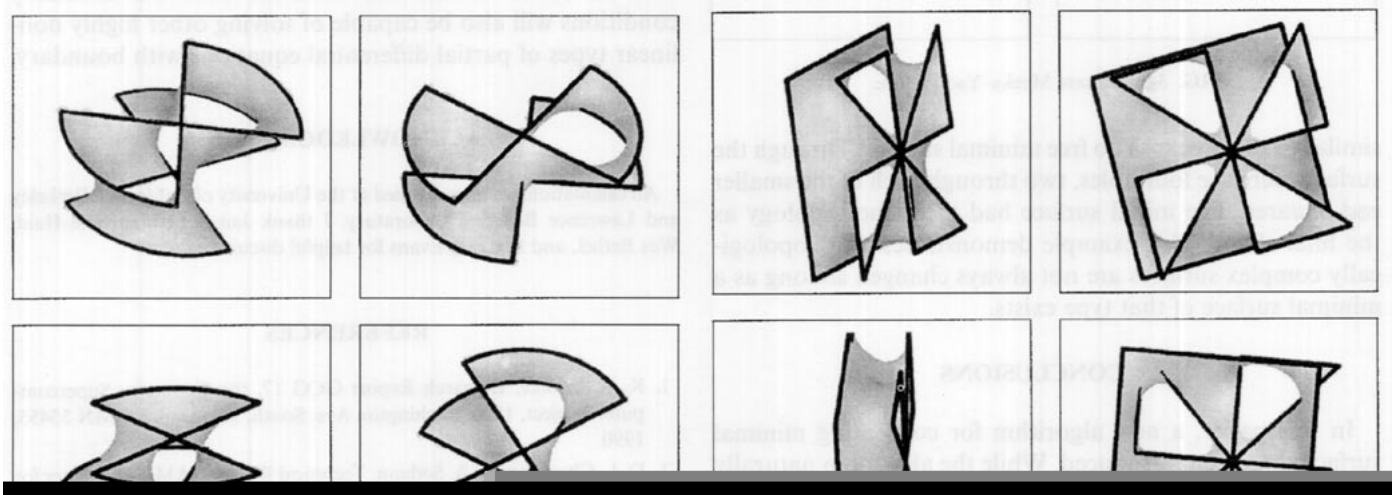

FIG. 33. Square and pinwheel surface.

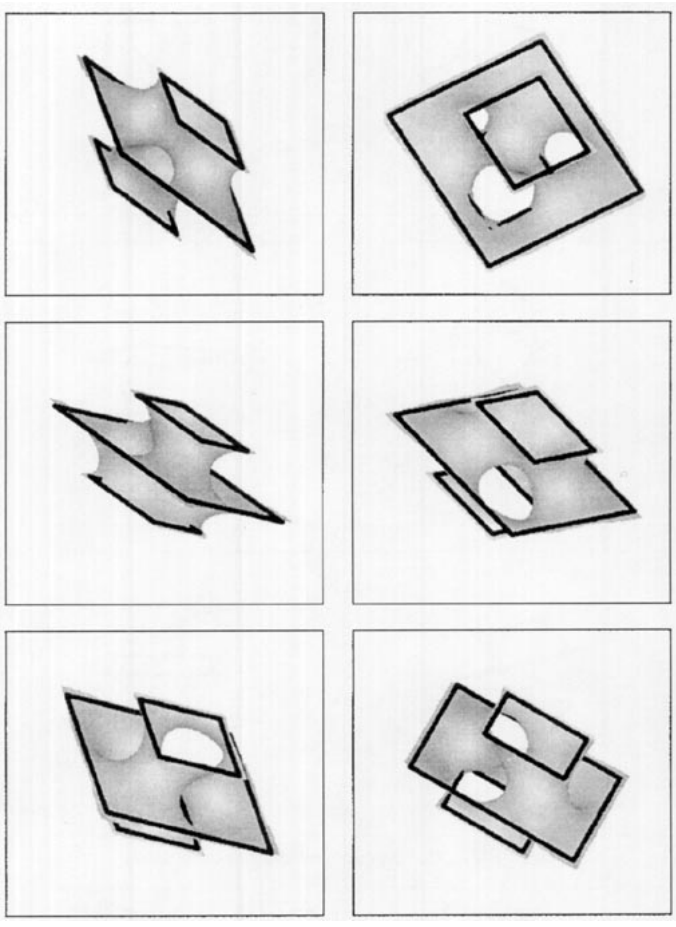

FIG. 34. Square Meeks–Yao surface.

similar to the Meeks–Yao free minimal surface. Through the surface there are four holes, two through each of the smaller end squares. The initial surface had the same topology as the final shape. This example demonstrates that topologically complex surfaces are not always changed so long as a minimal surface of that type exists.

5. CONCLUSIONS

In this paper, a new algorithm for computing minimal surfaces has been introduced. While the algorithm naturally computes topological changes in the evolving surface, certain desirable properties were sacrificed. At present, the new algorithm is not capable of studying minimal surfaces which have the boundary passing through an interior portion of the surface as in Fig. 35. The obvious reason is because the artificial values created by the interpolatory boundary conditions interfere with the natural motion of the surface, where the boundary passes through.

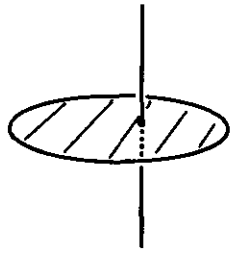

FIG. 35. Surface with boundary passing through interior.

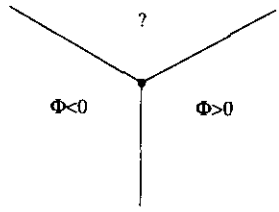

FIG. 36. Example of a triple point.

The algorithm is also not capable of handling triple points, for example, the catenoid with a disk in the center of Fig. 4. Because of the level set formulation, it is not possible to have the sign of Φ change across each surface (see Fig. 36). The possibility of extending the algorithm in this case is under investigation.

Several other extensions to the algorithm are also under investigation. Some results on computing non-zero constant-mean curvature surfaces and the flow of curves on manifolds can be found in [2]. More generally, it is hoped that the combination of the level set idea with boundary conditions will also be capable of solving other highly nonlinear types of partial differential equations with boundary.

ACKNOWLEDGMENTS

All calculations were performed at the University of California, Berkeley and Lawrence Berkeley Laboratory. I thank James Sethian, Ole Hald, Wes Bethel, and L. Craig Evans for helpful discussions.

REFERENCES

1. K. A. Brakke, Research Report GCG 17, the Geometry Supercomputer Project, 1200 Washington Ave. South, Minneapolis, MN 55455, 1990.
2. D. L. Chopp and J. A. Sethian, Technical Report PAM-541, Center for Pure and Applied Mathematics, University of California, Berkeley, November 1991.
3. P. Concus, *Math. Comput.* **21**, 340 (1967).
4. C. Coppin and D. Greenspan, A contribution to the particle modeling of soap films, *Appl. Math. Comput.* **26**, 315 (1988).
5. G. Donnay and D. L. Pawson, *Science* **166**, 1147 (1969).
6. A Elcrat and K. Lancaster, *Arch. Rat. Mech. Anal.* **94** (3), 209 (1986).
7. L. C. Evans and J. Spruck, *J. Differential Geom.* **33**, 635 (1991).
8. H. Graf and H. Thomas, *Math. Z.* **51**, 166 (1947).
9. D. Greenspan, *ICC Bull.* **4**, 91 (1965).
10. S. Hildebrandt and A. Tromba, *Mathematics and Optimal Form, Sci. Am.* Books, New York, 1985.

11. M. Hinata, M. Shimasaki, and T. Kiyono, *Math. Comput.* **28** (125), 45 (1974).
12. R. H. W. Hoppe, *SIAM J. Numer. Anal.* **24** (5) (1987).
13. J. H. Jellett, *Trans. R. Irish Acad.* **22**, 343 (1855).
14. J. C. C. Nitsche, *Lectures on Minimal Surfaces*, Vol. 1, Cambridge Univ. Press, London 1989.
15. S. Osher and J. A. Sethian, *J. Comput. Phys.* **79** (1), 12 (1988).
16. K. Pohlmeyer and K. H. Rehren, *Commun. Math. Phys.* **144**, 177 (1988).
17. R. Schoen and S. T. Yau, *Commun. Math. Phys.* **65**, 45 (1979).
18. R. Schoen and S. T. Yau, *Commun. Math. Phys.* **79**, 231 (1981).
19. R. Schoen and S. T. Yau, *Commun. Math. Phys.* **90**, 575 (1983).
20. J. A. Sethian, *Commun. Math. Phys.* **101**, 487 (1985).
21. J. A. Sethian, *J. Differential Geom.* **31**, 131 (1989).
22. H. J. Wagner, *Computing* **19**, 35 (1977).
23. D. K. Walker, L. N. Scotten, D. E. Hewgill, R. G. Racca, and R. T. Brownlee, *Med. Biol. Eng. Comput.* **21**, 31 (1983).