Contour Tracing by Piecewise Linear Approximations

David P. Dobkin
Silvio V. F. Levy
William P. Thurston[1]

Princeton University

Allan R. Wilks

AT&T Bell Laboratories
Murray Hill, New Jersey 07974

July 22, 1988

Abstract. We present a method for tracing a curve that is represented as the contour of a function in Euclidean space of any dimension. The method proceeds locally by following the intersections of the contour with the facets of a triangulation of space.

The algorithm does not fail in the presence of high curvature of the contour; it accumulates essentially no round-off error, and has a well-defined integer test for detecting a loop. In developing the algorithm we explore the nature of a particular class of triangulations of Euclidean space, namely, those generated by reflections.

Keywords: contour tracing, simplicial continuation, Coxeter triangulations.

1. Problem

We consider the problem of tracing the contours of a map between Euclidean spaces. If $f : \mathbf{R}^n \to \mathbf{R}^k$ $(k < n)$ is, say, smooth, the implicit function theorem [Hoffman 1975] (together with Sard's theorem [Guillemin and Pollack 1974]) says that, for most points y in the image of f, the *contour* $C = f^{-1}(y)$ is a smooth manifold of dimension $n - k$. If $k = n$, again for most points y in the image of f, C is a collection of isolated points, and finding C is the same as finding the *roots* of $y = f(x)$; although the solution to this problem has a vast literature, it seems to be of little help in solving the problem for $k < n$.

The problem of finding the contour C breaks into two parts: locating the various components of C (that is, finding at least one point in each component), then tracing around each component, using the fact that the components are connected. Later we state more precisely our definition of what it means to trace C; the general idea is that we want to compute a collection of "representative" points in C. In this paper we will focus on the tracing problem, although we discuss very briefly

[1]Research of the first three authors was supported in part by the National Science Foundation

is correspondingly easier to implement. A fairly general implementation of it is the subject of a paper currently under preparation.

Many interesting problems can be stated in the context of contour tracing. A few of them, drawn from computer graphics, computational geometry, pure mathematics and statistics, are discussed in the next section. These examples of applications are not meant to be exhaustive, but rather to illustrate some directions in which the general method can be applied. In section 3 we outline two possible approaches to the tracing problem, and indicate why we chose one over the other.

Section 4 is devoted to an easily understood instance of the algorithm, and to setting the stage for the more general case. In section 5 we generalize the triangulation, and in section 6 we present further computational details, including a theoretical analysis of the relative performance of the various triangulations. Section 7 contains the full algorithm, adapted to degenerate cases. Finally, in section 8 we give examples of the algorithm in action, while in section 9 we suggest several possible extensions of the method.

2. Some applications

We begin by describing a broad range of sample problems to which contour tracing has been applied.

Example 1: curve tracing

Many simple curves may be represented as the contour of some suitable function. For example, let $f : \mathbf{R}^2 \to \mathbf{R}^1$ be the function defined by $f(x,y) = x^2 + y^2$. The graph of f is a paraboloid and contours of f, for y positive, are circles centered at the origin. The contour of f containing the starting point $(r,0)$ is just the circle $f(x,y) = r^2$.

Example 2: shadow computation

Let Σ_1 and Σ_2 be spheres with centers $\mathbf{c}_1 = (x_1^0, y_1^0, z_1^0)$ and $\mathbf{c}_2 = (x_2^0, y_2^0, z_2^0)$, and radii r_1 and r_2. Assume that a light source is located at (lx, ly, lz) and that it is known that Σ_1 casts a shadow on Σ_2. We can study the outline of this shadow by considering a contour of the function $f : \mathbf{R}^6 \to \mathbf{R}^5$ with components $f = (f_1, f_2, f_3, f_4, f_5)$, where

$$f_i(x_1, y_1, z_1, x_2, y_2, z_2) = (x_i - x_i^0)^2 + (y_i - y_i^0)^2 + (z_i - z_i^0)^2, \quad \text{for } i = 1,2,$$

is the square of the distance from the point $\mathbf{p}_i = (x_i, y_i, z_i)$ to the center of sphere Σ_i;

$$f_3(\mathbf{p}_1, \mathbf{p}_2) = (x_1 - x_1^0)(x_1 - lx) + (y_1 - y_1^0)(y_1 - ly) + (z_1 - z_1^0)(z_1 - lz)$$

vanishes if and only if $\mathbf{p}_1$ lies on the horizon of the first sphere (the vector from the $\mathbf{p}_1$ to $\mathbf{c}_1$ is perpendicular to the vector from $\mathbf{p}_1$ to the light source); and

$$f_4(\mathbf{p}_1, \mathbf{p}_2) = (x_1 - lx)(y_2 - ly) - (x_2 - lx)(y_1 - ly),$$
$$f_5(\mathbf{p}_1, \mathbf{p}_2) = (x_1 - lx)(z_2 - lz) - (x_2 - lx)(z_1 - lz)$$

display of torus knots. The first of these problems is discussed in section 8. For the latter, other considerations are more significant than the contour tracing and the methodology will be described elsewhere.

3. A Framework for the Algorithm

We now analyze the problem of contour tracing in more detail. Let $f : U \to \mathbf{R}^{n-1}$ be a C^1 function on an open, connected subset U of $\mathbf{R}^n$, and assume that we are given x_0 and y_0 such that $y_0 = f(x_0)$. The *contour* of f passing through x_0 is the inverse image $C = f^{-1}(y_0)$, or sometimes the connected component of it containing x_0. Assuming for the moment that y_0 is a *regular value* of f, that is, $df(x)$ is of rank $n-1$ whenever $f(x) = y_0$, the implicit function theorem says that C is a disjoint union of smooth curves in U, each homeomorphic to either the reals $\mathbf{R}$ or to the circle S^1.

Our problem is to find the connected component C_0 of C that contains x_0. More specifically, we would like to generate a finite sequence of points in C_0 by starting at x_0, and stepping along the curve until we either come back to x_0 or we reach the boundary of U. In the latter case, we start again at x_0 and trace in the opposite direction until we again hit the boundary. These are the only possibilities.

The first decision to be made is, how big should the steps along the curve be? A natural criterion for the step size is that every point of the curve should be within ε of one of the computed points, for some preassigned $\varepsilon > 0$. In addition, we might like the computed points to be closer together when the curve is changing more rapidly, in order that the details of the changes can be followed. In fact, we might vary the step size at each step, according to some estimate of the local curvature of C_0 at the current position. In what follows, we will not take this dynamic approach; we will assume that points that are at most ε apart on the curve are sufficient. This is certainly a reasonable assumption if C_0 is being graphically presented as, say, the linear spline connecting the computed points, and if ε is taken to be equal to the resolution of the plotting device; in this case the true picture of C_0 would be indistinguishable from the computed picture. We will talk a bit more about variable step sizes in section 9.

The basic problem is then to find the next point $x_{i+1} \in C_0$, given the current point $x_i \in C_0$, and perhaps previous points. Two general approaches can be broadly characterized as "infinitesimal" and "local" (but see [Geisow 1983] for a classification of methods that is somewhat different from ours); *grosso modo*, we can say that the local approach consists in tracing the exact contour of an approximation to f, while the infinitesimal approach traces an approximate contour of the exact f (figure 1).

The infinitesimal approach is easiest to describe for $n = 2$ (figure 1(a)). Our current position is x_i. on the contour $C = f^{-1}(y_0)$. To find x_{i+1}, we use the derivative of f at x_i to determine the tangent line to the curve. We take a small step away from x_i along this line, to the point $x_i + \varepsilon f'(x_i)$, then do a one-dimensional search in the direction orthogonal to the tangent, along the dotted line, to rediscover

in $\tilde{C}$ (figure 2(c)). It may even happen that a component disappears entirely, when the open set bounded by it does not enclose any points of the grid (figure 2(d)).

Figure 2

In each of these situations the detail ignored or misconstrued in passing to the approximation is geometrically "negligible" (in the sense that it's occurring in a scale smaller than the chosen resolution), although, as we have seen, it may be of topological significance. At any rate it is unreasonable to expect any algorithm to yield the topology of $f^{-1}(y_0)$ for all differentiable f, since this set can be an arbitrary compact (that is, bounded and topologically closed) subset of $\mathbf{R}^n$.

The one situation when the difference between C and $\tilde{C}$ can have large-scale consequences is when one component of C splits into two or more components of $\tilde{C}$, as in figure 2(c). This case is examined in more detail in figure 3, where we show $\tilde{C}$ for $f(x,y) = xy$ and various positions of the triangulation. Cases (a) and (b) present no difficulty because $\tilde{C}$ still has just one component, but in (c) the algorithm would only draw the piece of $\tilde{C}$ containing the starting point, and thus would give an unsatisfactory approximation for C.

If f is differentiable one way to deal with this problem is to incorporate in $\tilde{f}$ a bias toward zero: one replaces values of components of f at the vertices of the triangulation by 0 whenever their absolute value is less than a constant ε. As ε increases so does the connectivity of $\tilde{C}$, ensuring that C is entirely represented by $\tilde{C}$: figure 3(d). On the other hand, if ε is too large too many simplices get "blacked out". The right value of ε is essentially proportional to the second derivative of f and the square of the resolution (since the first derivative is relatively small at the places where things go wrong), but it's often easier to choose ε by trial and error.

Figure 3

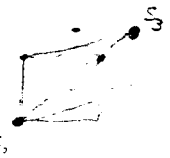
Conventions

To simplify the notation we will, in sections 4 through 7, use f to denote the affine (linear) approximation of some original, not necessarily differentiable, function from a bounded domain of interest $U \subset \mathbf{R}^n$ into $\mathbf{R}^{n-1}$. Without loss of generality, we assume that the contour that interests us is the inverse image $f^{-1}(0,\ldots,0)$ of the origin. We assume further that we know a point x_0 in this contour, and we denote by C the connected component of the contour that contains x_0. Our goal is to describe the piecewise linear set C.

for every permutation $\pi = (i_1, i_2, \ldots, i_n)$ of the integers $\{1, 2, ..., n\}$, the set of points $(x^1, \ldots, x^n)$ that satisfy the constraints

$$x^{i_1} \leq x^{i_2} \leq \cdots \leq x^{i_n}$$

is an n-simplex, and that these $n!$ simplices triangulate the unit hypercube. The vertices of such a simplex are



$$\begin{aligned} s_0 &= (0, \ldots, 0) \\ s_1 &= (0, \ldots, 1, \ldots, 0) && \text{(1 in the } i_n\text{-th position)}, \\ s_2 &= (0, \ldots, 1, \ldots, 1, \ldots, 0) && \text{(1 in the } i_n\text{-th and } i_{n-1}\text{-th position)} \\ &\cdots \\ s_n &= (1, \ldots, 1); \end{aligned}$$

notice that s_0 and s_n are shared by all simplices, and the edge they span is a main diagonal of the unit hypercube.

We complete the triangulation of $\mathbf{R}^n$ by translating the triangulation of the unit hypercube to all other hypercubes. Each simplex in this triangulation can be uniquely referenced by the lowest-coordinate corner of the hypercube it's contained in, together with a permutation of $\{1, \ldots, n\}$. There is, however, a simpler way to reference each simplex, namely, by the coordinates of its center of mass (which, when multiplied by $n + 1$, are integers). It is important to have a simple way to address simplices because, as we shall see later, we must keep track of already-visited ones.

With these examples in mind, we now describe in detail a simple version of the algorithm. Imagine that we have an arbitrary triangulation of $\mathbf{R}^n$. (There is no need to compute the positions of all the simplices of this triangulation; all we will need is a means of finding the simplices that abut a given simplex. Finding a triangulation for which this is easy turns out to be an interesting problem in its own right, and we take it up in the next section.) We will denote by C^σ the (non-empty) intersection of the contour C with a simplex σ of this triangulation.

For this simple version we'll want to avoid two types of situations. The first occurs when C^σ, which normally is a line segment, has dimension greater than one. The second is when C^σ, which normally goes in and out of σ through the interior of facets, happens to intersect also the interior of a face of dimension less than $n - 1$. The first type of degeneracy implies the second; both are ruled out for the nonce.

As in the two-dimensional example above, we work by keeping track of three pieces of information, which together constitute the *current state* $(\sigma, k, \mathbf{b})$ and reflect the situation as we are passing into a new simplex:

(1) the *current simplex* σ with vertices $s_0, \ldots, s_n \in \mathbf{R}^n$;
(2) the index k of the *current facet* through which we're entering σ (remember that facets are indexed by the subscript of the opposite vertex); and
(3) the *barycentric coordinates* $\mathbf{b}_{\text{in}} = (b^0_{\text{in}}, \ldots, b^n_{\text{in}})^t$, of the *current position*.

Here the equality has at least one solution $\mathbf{b}_{\text{in}}$, so the dimension of its set of solutions is $n+1$ minus the rank of F_σ. (Thus the first type of degeneracy mentioned above occurs when F_σ does not have full rank n.) The general solution of the equality can be written $\mathbf{b}_{\text{in}} + t\mathbf{v}$, where $\mathbf{v}$ is a non-trivial solution of the homogeneous equation

$$(*) \qquad F_\sigma \mathbf{v} = 0.$$

Finding $\mathbf{v}$ is therefore just a standard linear algebra problem.

To ensure that a barycentric vector anchored at $\mathbf{b}_{\text{in}}$ and pointing in the direction $\mathbf{v}$ points into σ, we simply look at the k-th coordinate v^k of $\mathbf{v}$ and multiply $\mathbf{v}$ by -1 if this coordinate is negative. We observe for later use that a vector anchored in σ, and in the direction $\mathbf{v}$, points away from facet i if v^i is positive, points towards facet i if v^i is negative and is parallel to facet i if v^i is zero. If v^k were zero C would run along facet k and would intersect a lower-dimensional face, so we would have the second kind of degeneracy mentioned above. Furthermore, since facet k is shared with the previous simplex (unless we're just starting the process), this would imply that in that simplex there was a degeneracy too.

It is now a simple matter to find the exiting facet in which $\mathbf{b}_{\text{out}}$ lies. In general, $\mathbf{b}_\mathbf{v}(t)$ lies on an extended facet of σ whenever it has at least one zero coordinate. For t slightly larger than 0, $\mathbf{b}_\mathbf{v}(t)$ lies in the interior of σ (because $\mathbf{v}$ points into σ), so τ is the smallest positive number t for which one of the coordinates of $\mathbf{b}_\mathbf{v}(t)$ is 0. In symbols,

$$(**) \qquad \tau = \min\{-b^i_{\text{in}}/v^i : v^i \neq 0, -b^i_{\text{in}}/v^i > 0\}.$$

The index k' that achieves the minimum gives the exiting facet, that is, $-b^{k'}_{\text{in}}/v^{k'} = \tau$. (If the minimum were achieved for more than one k' the contour would pass out of σ through one of its faces of dimension less that $n-1$, and a degeneracy of the second kind would occur.)

The simplex σ' into which C now passes shares all but one of its vertices with σ, namely, $s_{k'}$. Thus to find σ' we just need to compute the new $s_{k'}$. In the next section we show how our triangulation of $\mathbf{R}^n$ can be chosen in such a way that this is very easy to do.

5. Triangulations

So far we have identified two properties that we would like our triangulation to have: (a) it should be easy to find the simplex that shares a facet with a given simplex; and (b) it should be possible to label the vertices of all the simplices at the same time with indices $0, \ldots, n$, in such a way that each of the $n+1$ vertices of an n-simplex has a different label. Two more conditions are intuitively desirable: (c) all simplices should have more or less the same size, and (d) their dimensions should be roughly the same in all directions. The reasoning here is that, having normalized the largest dimension of a simplex to the pixel size, say, we don't want to cross too

looks like an equilateral triangle, and in fact that is precisely what it represents, but only by accident: the correct way to read the diagram is by observing that it stands for a simplex with three edges (nodes), each pair of which have a 60° angle between them (one edge in the diagram).

A sample of each of the three two-dimensional Coxeter triangulations, P_3, R_3 and V_3 is given in figure 5.

Figure 5

Notice that there are four families that have representatives in all but a few low dimensions, namely, P_m, Q_m, R_m and S_m. These are the natural candidates for an algorithm that is meant to work in any dimension, although the remaining entries in the list might be useful in their specific dimensions.

To wrap up the description of the iteration step of our algorithm, in section 4, we needed to display a rule to compute the simplex that shares a given facet with the current simplex. Here this boils down to finding the reflection of a vertex s_i of the simplex in the opposite facet. The reflection of a point p in a hyperplane H is reached by going from p to its orthogonal projection $\pi_H(p)$ on H, then once again as far in the same direction: in symbols, $2\pi_H(p) - p$. Finding $\pi_H(p)$ is straightforward given the equation of H and the coordinates of p, but it turns out that we can avoid all of that because of the geometry of the simplices. Our treatment loosely follows Coxeter's [1973, p. 182–184].

Let $\pi_i(s_i)$ be the projection of s_i in the hyperplane containing facet i. We first write $\pi_i(s_i)$ in barycentric coordinates with respect to the vertices of facet i: $\pi_i(s_i) = \sum_{j \neq i} w_i^j s_j$, with $\sum w_i^j = 1$. To figure out the w_i^j, consider the plane containing the edge that joins s_i and s_j and orthogonal to the opposite face $\Phi_{\{i,j\}}$, which it meets at u. The intersection of the simplex with this plane is shown in figure 6. In this figure, w_i^j is, by definition, the ratio between the length of the segment $u\pi_i(s_i)$ and the side us_j. Now the length of $u\pi_i(s_i)$ equals the length of side us_i times the cosine of the angle at u, which is just the dihedral angle $\theta_{ij} = \pi/p_{ij}$; since the sides of a triangle are inversely proportional to the corresponding altitudes, we can write $w_i^j = \cos\theta_{ij} \cdot (h_i/h_j)$, where h_i and h_j are the altitudes of the simplex corresponding to vertices s_i and s_j.

Figure 6

To calculate the altitudes h_i, we first show that $\sum_{i=0}^{n} \frac{\mathbf{n}_i}{h_i} = 0$, where $\mathbf{n}_i$ is the unit vector normal to facet i and pointing out. This equality can be verified by considering its inner product with each vector $s_j - s_0$: we have $\mathbf{n}_i \cdot (s_j - s_0) = 0$ unless $i = 0$ or j, in which cases the inner product is h_0 or $-h_j$, respectively. Since the n vectors $s_j - s_0$ form a basis for $\mathbf{R}^n$, this implies the equality.

We have just shown that the expression $\left\| \sum_{i=0}^{n} \mathbf{n}_i x^i \right\|^2$ vanishes when the x^i are inversely proportional to the altitudes h_i. Now this expression is a quadratic form

with 0 and n at the two ends.) The vertices of a P_m are $s_i = (m-i)\big((m(m-1))^{-1/2}, ((m-1)(m-2))^{-1/2}, \ldots, ((m-i+1)(m-i))^{-1/2}, 0, \ldots, 0\big)$, where the first i coordinates are non-zero. (Here the nodes of the Coxeter diagram are numbered in order around the circle, with 0 and n next to one another.) The reflection rules for P_m are extremely simple: to reflect vertex s_i in the opposite facet,

$$(*) \qquad \text{replace } s_i \text{ by } s_{i-1} + s_{i+1} - s_i,$$

where we agree to take the indices modulo m whenever we talk about P_m simplices. The next simplest rules occur for R_m:

$$(**) \qquad \text{replace } s_i \text{ by } \begin{cases} 2s_1 - s_0 & \text{if } i = 0, \\ s_{i-1} + s_{i+1} - s_i & \text{if } 0 < i < n, \\ 2s_{n-1} - s_n & \text{if } i = n. \end{cases}$$

Once we have expressed a particular set of reflection rules in the form $s_i \leftarrow \sum_{j \neq i} w_i^j s_j$, we can apply them to an arbitrary simplex, having nothing to do with the Coxeter triangulation. The rules will no longer describe reflections, but they will generate a triangulation of $\mathbf{R}^n$, because the rules are preserved by affine transformations and we can always find an affine transformation to transform the Coxeter simplex into our newfangled simplex. Thus it is that, if we apply the P_m reflection rules to an R_m simplex, we get the example triangulation in section 4. Like P_m, but unlike R_m, it has the same number of simplices incident at each vertex, as illustrated by figure 7 in the case $m = 3$: in (a), R_3 with R_3 rules, and in (b), R_3 with P_3 rules. (These two triangulations are referred to as K and H in the fixed-point literature; see, for example, [Todd 1976].) Notice that, in general, triangulations obtained in this way from arbitrary simplices are not monohedral, and so may be inferior from the point of view of criterion (c) at beginning of this section; but such is not the case with the ones in figure 7.

Figure 7

6. Some Implementation Details

To complete the non-degenerate algorithm, we must choose a triangulation of the appropriate dimension from figure 4, then find an appropriate instance of the prototype simplex containing the given starting point. We also need a test to determine when we have found the entire contour. We take up each of these points in turn.

ular simplex of the same dimension. Table 2 shows this value up to dimension 8, and again the pattern is already clear: the Q_m, R_m and S_m entries are virtually indistinguishable from one another (asymptotically they are the same) and more than 50% bigger than those for P_m (the asymptotic ratio is $\sqrt{3}$).

n	P_m	Q_m	R_m	S_m	T_m	U_m	V_m
2	1.00		1.21				1.37
3	1.05		1.39	1.61			
4	1.12	1.50	1.56	1.61		1.81	
5	1.18	1.70	1.71	1.76			
6	1.25	1.86	1.85	1.90	2.00		
7	1.31	2.01	1.98	2.03	2.32		
8	1.37	2.14	2.10	2.15	2.72		

Table 2. Normalized ratios of circumscribed to inscribed spheres

The third, and subtlest, measure of nonregularity is the condition number of the covariance matrix of a random point, uniformly distributed inside the simplex. (The *condition number* of a matrix (for the Euclidean norm) is the ratio between its largest and smallest eigenvalues. The condition number for regular simplices is always 1, so no normalization is necessary here.) The logs of these numbers are given in Table 3, again up to dimension 8 only because the relative behavior after that is the same:

n	P_m	Q_m	R_m	S_m	T_m	U_m	V_m
2	0.00		2.20				3.18
3	1.39		3.53	4.29			
4	1.92	3.22	4.50	4.09		5.70	
5	2.77	4.64	5.27	5.03			
6	3.24	5.54	5.91	5.76	5.74		
7	3.84	6.22	6.46	6.36	7.33		
8	4.23	6.78	6.94	6.87	8.72		

Table 3. Logs of the condition numbers of Coxeter simplices

All measures seem to indicate that P_m has a somewhat better geometry for our purposes than all other Coxeter triangulations. It also has the simplest reflection rules. However, it has one slight disadvantage over R_m, in that it requires messier arithmetic to pass from real-world coordinates to simplex coordinates.

to the resolution, or perhaps half the resolution; the expected step size can be determined as explained at the beginning of subsection 6.1.

Finally, we have to find an instance of the scaled simplex that contains the starting point. This can be done in two ways, depending on the application. If only one contour of the function is to be traced, we can translate our scaled prototype simplex so that x_0 equals, say, the barycenter of its facet n. Then we initialize the state accordingly ($k = n$, $\mathbf{b}_{\text{in}} = (1/n, \ldots, 1/n, 0)^t$ and σ as appropriate). If, on the other hand, we are computing several contours, we may desire them all to use the same triangulation, rather than computing a separate translation of the triangulation for each contour. In this case, we must fix one particular instance of our triangulation and find, for each contour to be traced, the particular simplex that contains the starting point. This can be a little tricky; a nice treatment of essentially the same problem is given in [Conway and Sloane 1982]. Unless x_0 happens to lie on a simplex face already, we also have to find our first $\mathbf{b}_{\text{in}}$ and k; as the reader can easily verify, this is accomplished by solving for $\mathbf{v}$ and τ as in section 4 (formulas (*) and (**), with $\mathbf{b}_{\text{in}}$ replaced by the barycentric coordinates of x_0 in the latter), then adding $\tau\mathbf{v}$ to x_0.

6.3. Termination

Termination of the algorithm occurs in two cases: when the contour exits the region U of interest, and when it loops. An out-of-bounds check is straightforward; we stop, say, when one vertex of σ falls outside the region of interest (f may not be defined there anyway). Then we restart the algorithm from x_0 in the opposite direction; having saved the initial state $(\sigma, k, \mathbf{b}_{\text{in}})$ we just have to replace σ by its reflection in facet k.

There is a simple integer test for loops as well. In a basis given by n edges $\bar{s}_i - \bar{s}_0$ of a *fixed* simplex of the triangulation, the coordinates of the barycenter of any simplex (minus $\bar{s}_0$) are integral multiples of $1/m$, so we can multiply these coordinates by m to obtain an integer vector that uniquely identifies a simplex. As a matter of fact, it can be advantageous to work with coordinates in the basis $\{\bar{s}_i - \bar{s}_0\}$ for other computations as well, including applications of the reflection rules; this avoids possible accumulation of roundoff errors.

Checking if we're back to the start, then, is trivial: we just compare the integer coordinates of the current simplex with those of the starting simplex, and stop if they are equal. It may also be useful to keep a hash table of the simplices that have been visited; for example, if we have a method of generating starting points on different components of the same contour, we will probably want a test for whether two starting points lie on the same component.

7. The Degenerate Case

Having completed the description of the non-degenerate algorithm, we now tackle the degenerate case. Recall first that two sorts of degeneracies can occur: the first

7.1. Crossing a Simplex

Our task after entering a simplex σ through a face Φ_A is to find at least one not previously visited link starting at the current point $\mathbf{b}_{\text{in}}$ and contained in σ. Let's first consider how to characterize segments in C_σ that start at the current point, then how to characterize the condition that such a segment is a link.

A direction $\mathbf{v} \in \ker F_\sigma$, anchored at the current point, points into σ if and only if $v^i \geq 0$ for all $i \in A$. In addition, not all of these v^i can be zero, otherwise $\mathbf{v}$ is parallel to Φ_A, which implies that C intersects Φ_A in more than one point. Since we're only interested in $\mathbf{v}$ up to a factor, we can express this last condition by saying that $\sum_{i\in A} v^i = 1$. Thus finding a segment in C_σ starting at the current point reduces to finding $\mathbf{v} = (v^0, \ldots, v^n) \in \mathbf{R}^{n+1}$ such that

(i) $F_\sigma \mathbf{v} = 0$,
(ii) $\sum_{i\in A} v^i = 1$, and
(iii) $v^i \geq 0$ for $i \in A$.

By definition, such a segment $S_\mathbf{v} = \{\mathbf{b}_{\text{in}} + t\mathbf{v} : t \in [0, \tau]\}$ determines a link—an edge of C^σ—if and only if it does not intersect any segment whose endpoints $\mathbf{b}_1, \mathbf{b}_2$ are in C^σ but not in $S_\mathbf{v}$. We claim that, if $B_\mathbf{v} = \{i \in A : v^i = 0\}$ is the set of A coordinates of $\mathbf{v}$ that vanish, this condition is equivalent to saying that there is no solution $\mathbf{v}' \neq \mathbf{v}$ to (i) and (ii) above with $B_\mathbf{v} \subseteq B_{\mathbf{v}'}$. Geometrically, $B_\mathbf{v}$ is the label of the smallest face of σ that contains $S_\mathbf{v}$, and the claim is that $S_\mathbf{v}$ determines a link if and only if $\Phi_{B_\mathbf{v}} \cap C^\sigma$ has dimension one.

For suppose there is such a $\mathbf{v}'$: then there are solutions to (i)–(iii) on both sides of $\mathbf{v}$, say $\mathbf{v}_1 = (1-\varepsilon)\mathbf{v} + \varepsilon\mathbf{v}'$ and $\mathbf{v}_2 = (1+\varepsilon)\mathbf{v} - \varepsilon\mathbf{v}'$, for $\varepsilon > 0$ sufficiently small. For t sufficiently small, $\mathbf{b}_1 = \mathbf{b}_{\text{in}} + t\mathbf{v}_1$ and $\mathbf{b}_2 = \mathbf{b}_{\text{in}} + t\mathbf{v}_2$ belong to C^σ, and the segment $\mathbf{b}_1\mathbf{b}_2$ intersects $S_\mathbf{v}$, so $S_\mathbf{v}$ is not a link. Conversely, assume there is a segment in C^σ whose endpoints $\mathbf{b}_1$ and $\mathbf{b}_2$ do not lie in $S_\mathbf{v}$; then $\mathbf{v}$ is a convex linear combination of the directions $\mathbf{v}_1$ and $\mathbf{v}_2$ corresponding to $\mathbf{b}_1$ and $\mathbf{b}_2$ and since, by assumption, $v_1^i \geq 0$ and $v_2^i \geq 0$ for $i \in A$, this implies $B_\mathbf{v} \subseteq B_{\mathbf{v}_1}$ and $B_\mathbf{v} \subseteq B_{\mathbf{v}_2}$.

In order to identify directions $\mathbf{v}$ such that $\Phi_{B_\mathbf{v}} \cap C^\sigma$ has dimension one, we define the following routine:

```
solve(F, A, B)
{
  ⟨ try to find v = (v^0, ..., v^n) ∈ R^{n+1} such that Fv = 0, Σ_{i∈A} v^i = 1, and
      v^i = 0 for i ∈ B ⟩:
  if (v exists) {
    if (v is unique) {
      if (v^i ≥ 0 for all i ∈ A) return v;
      else if (v^i ≤ 0 for all i ∈ A) return −v;
        else return FAIL;
    }
    else return ∞;
  }
```

Skipping supersets of B is also easy:

$$next\_non\_superset(B) \leftarrow \begin{cases} \{b_0, \ldots, b_{i-2}, b_{i-1}, b_i + 1\} & \text{if } b_i < k, \\ \{b_0, \ldots, b_{i-2}, b_{i-1} + 1\} & \text{if } b_i = k \text{ and } i > 0, \\ FAIL & \text{if } b_i = k \text{ and } i = 0. \end{cases}$$

We also need to keep a list of values of B for which *solve* returned ∞ or a vector, called the *hit list*, so we can skip all subsets of sets in this list. Whenever *solve* returns ∞ or a vector for some B, we add B to the hit list. We also delete any existing subsets of the newly added B from the hit list, as they are now redundant: in future tests, any subset of such a subset will also be a subset of B.

When describing the non-degenerate algorithm, we mentioned that we might want to keep a hash table of simplices already visited. Here such a table becomes a necessity, in a slightly different form: we must store each pair (σ, A), together with a pointer to a structure $\mathcal{B}_{(\sigma,A)}$ that contains the hit list and the value of B at the end of the last call to *find_link*(σ, A). This structure is read every time *find_link*(σ, A) is invoked—these invocations being non-consecutive, since we're conducting a depth-first search—and destroyed when *find_link* knows there are no more vectors to be returned. (Actually, $\mathcal{B}_{(\sigma,A)}$ may not be created at all if we only have to call *solve* once, with $B = \emptyset$.) A pair (σ, A) in the hash table is called *dead* if $\mathcal{B}_{(\sigma,A)}$ no longer exists; otherwise it is called *alive*.

```
find_link(σ, A)
{
  if (B_(σ,A) exists)    /* we have seen these arguments before */
    ⟨get B and the hit list from B_(σ,A)⟩;
  else ⟨set B to ∅ and the hit list to the empty list⟩;
  while (B ≠ FAIL) {    /* there are still faces to examine */
    if ⟨B ⊂ B' for some B' in the hit list⟩ {
      B ← successor(B);    /* skip this B */
      continue;    /* go back to beginning of while loop */
    }
    v ← solve(F, A, B);    /* look for solutions in Φ_B */
    if (v = FAIL) {    /* no solution */
      B ← next_non_superset(B);    /* skip this B and all its supersets */
      continue;
    }
      /* at least one solution */
    ⟨delete subsets of B from the hit list⟩;
    ⟨add B to the hit list⟩;
    if (v = ∞) {    /* too many solutions */
      B ← successor(B);
      continue;
    }
```

geometric sense, because this reflection is not always a simplex in the triangulation (think of the 60°-vertices in figure 5(c)); if it is, it coincides with σ_{opp}.

As seen from a point in Φ_Z, the set of directions that point into σ_{opp} is opposite the set of directions that point into σ, which means that when the direction of C changes little at Φ_Z chances are good that C goes into σ_{opp}. But this is by no means always the case, because, in principle, f can take any value at the remaining vertices of σ_{opp}. For a concrete example, consider $f = (x^2 + y^2) - n^2$ on $\mathbf{R}^2$, for n an integer, and the R_3 triangulation whose vertices form the lattice $\mathbf{Z}^2 \subset \mathbf{R}^2$. The lattice point $(0, n)$ (together with $(n, 0)$, $(0, -n)$ and $(-n, 0)$) lies in C, and it is easily seen that the two triangles which intersect C and have $(0, n)$ as a vertex are not opposite with respect to $(0, n)$.

This situation tends to occur when, as in this example, the intersection C^σ is very close to one of the facets of σ. It also occurs more often as n or q increases, simply because the measure of the set of directions that point into σ_{opp} from a point in a codimension-q face decreases as a fraction of the measure of the total set of directions.

Our problem, then, is the following: as we're exiting a simplex σ through a face of codimension ≥ 1, we want to find σ_{opp} quickly, and, if it turns out that $C^{\sigma_{\text{opp}}}$ is empty, we want to examine as few as possible of the other simplices containing the face before we find one where the contour continues.

To do this we will again consider Coxeter triangulations. Readers who skipped the discussion in section 5 are encouraged to go back and read it now, or they can skip this one too, going down to the paragraph containing formula (*).

Imagine applying to σ all the symmetries of the triangulation that leave the face Φ_Z fixed. These symmetries evidently form a group G, which is finite because its elements are in one-to-one correspondence with the simplices in the star of Φ_Z. Furthermore, G is generated by reflections in the q facets of σ that contain Φ_Z. The identity 1 of G represents σ, and some element $\neq 1$ of G of order two, which we call -1, represents σ_{opp}.

One extreme case of this procedure occurs when $q = 1$, and then the group reduces to $\{1, -1\}$. The other extreme is $q = n$, that is, Φ_Z is a vertex: then G can be seen as a group acting by reflections on an $(n-1)$-dimensional sphere around Φ_Z, and determining there a *spherical triangulation*. Spherical reflection groups and spherical triangulations can be assigned Coxeter diagrams, following the same conventions as their Euclidean counterparts. By the very nature of this construction, the Coxeter diagram for the group that leaves fixed a vertex $s_i = \Phi_{\{0,\ldots,i-1,i+1,\ldots,n\}}$ of σ can be obtained by removing from the Coxeter diagram of σ the node that represents facet i, plus any edges that end at that node. Observe that this can yield a disconnected diagram; in this case the face group G is a direct product of smaller groups corresponding to the pieces of the diagram, because the absence of an edge between two nodes means that the angle between the respective facets is $\pi/2$. so reflections in these two facets commute. A list of connected spherical Coxeter diagrams can be found on page 297 of [Coxeter 1973].

(But first, a word of caution. If we insist on tracking all of the 1-skeleton of C, we do have to examine the star of Φ_Z exhaustively—even if C intersects $\sigma_{\mathbf{opp}}$. This is because C can actually intersect any number of simplices in the star, and there is no way to tell by looking at some of them what C will do in the others: for example, think of f defined in one simplex σ in such a way that C^σ is a vertex plus an interior line segment, then extend f to the star of this vertex by applying to σ the group of symmetries that fix the vertex.

Thus, in addition to keeping a list of active pairs (σ, A), one could keep a list of active node faces: all exiting faces of codimension $q > 1$ whose stars have not been completely examined yet. One could define a procedure *find_simplex*(Φ), similar to *find_link*, that would go through all the n-simplices in the star of such a face Φ, in some order, and return all those that intersect C, one at a time. Obviously, it would be advantageous to have things organized in such a way that, given a current node face Φ, all pairs (σ, A) such that $\Phi = \Phi_A$ in simplex σ are easily accessible: by consulting the hit lists of such pairs one could avoid calling *solve* on faces that had been examined previously. Better yet would be to merge *find_simplex* and *find_link* into one procedure that examines all the simplices, of any dimension, that have Φ as a face.)

Here's the alternative method. The *corona* of a simplex Φ in a triangulation is the union of all simplices that are faces of simplices in the star of Φ, but do not have Φ as a face: in some sense, the simplices "opposite" Φ. (The word "link" is sometimes used for this concept, but here it would be terribly confusing.) Thus, if Φ contains a node of C, the links of C starting at this node end at points on the corona of Φ. In particular, the point $\mathbf{b}_{\mathbf{in}}$ where we got into the current simplex σ and from which we drew a link to the exiting face Φ_Z lies on the corona of that face. Since Φ_Z is where we got into trouble, we will try to skirt it altogether by navigating on its corona.

To do this, we relax momentarily the condition that the last component of f vanishes, and follow the mini-contour defined by the first $n-2$ components of f on the corona of Φ_Z, starting at $\mathbf{b}_{\mathbf{in}}$. If all goes well, somewhere along this mini-contour we will find another zero for the last component of f. This is a point where C exits the star of Φ_Z, and we can resume our previous contour tracing, leaving Φ_Z behind. Conceptually, then, we're reducing the dimension of the problem, by solving for the contour of a function having $n-2$ components and defined on an $(n-1)$-dimensional space, the corona of Φ_Z.

This may sound complicated, but algorithmically it is not, because we can use the whole machinery developed so far with very little change. First observe that a point in the star of Φ_Z is on the corona if and only if its k-th barycentric coordinate is zero, for some $k \notin Z$. Thus tracing the contour of the first $n-2$ components of f on the corona of Φ_Z is more or less like tracing the contour of $f_{(1)} = (f^1, \ldots, f^{n-2}, b^k)$ in $\mathbf{R}^n$, where $f^1, \ldots, f^{n-2}$ are the components of f and b^k is the function that gives the k-th barycentric coordinate of a point with respect to the current simplex. However, k may change during this process.

8. Examples

We give two sets of examples of the algorithm in action, in dimensions 2 and 4. The first illustrates its behavior in the presence of high curvature of the contour, and the second in the tracing of contours of non-regular values. In each case the R triangulation was used with the P reflection rules, as outlined in section 5.

Consider the complex function $q_c : \mathbf{C} \to \mathbf{C}$ defined by $q_c(z) = z^2 + c$, where c is a fixed complex number. Fix a positive integer $k > 0$ and define $f : \mathbf{C} \to \mathbf{R}$ by $f(z) = \left|q_c^k(z)\right|$, where q_c^r is the r-th iterate of q_c, that is, $q_c^1(z) = q_c(z)$ and $q_c^{r+1}(z) = q_c(q_c^r(z))$ for $r \geq 1$. If we now consider f as a function from $\mathbf{R}^2$ into $\mathbf{R}$, it becomes a candidate for the contour tracing procedure. For the two values of c used in the examples below, any sufficiently large real number is a regular value of f. In addition, the corresponding contour is a single closed curve.

This example is interesting because if we compute a sequence of contours for a fixed value of f, for increasing values of k, they will converge to the so-called *Julia set* of q_c [Blanchard 1984], which is a fractal, or self-similar. Thus, as k increases, the curvature of the contour increases without bound, and the contour would be very difficult to trace with an infinitesimal method. This behavior is not surprising since f is the absolute value of a polynomial of degree 2^k. Self-similarity means we can monitor the interaction of contour and triangulation at various relative scales.

Figure 8 shows $f^{-1}(9)$ for $k = 15$ and for two values of c. Notice that in Figure 8(a) the contour comes very close to itself in many places. At corresponding places on smaller branches the true contour also looks like two strands very close to one another, but the approximate contour unites the two.

Figure 8

In our second set of examples, we are interested in values of c and z for which z is a fixed point of q_c^k, that is, $q_c^k(z) = z$, and for which the derivative of q_c^k evaluated at z has absolute value 1. So we define $g : \mathbf{C} \times \mathbf{C} \to \mathbf{C} \times \mathbf{R}$ to be $g(c, z) = \left(q_c^k(z) - z, |(q_c^k)'(z)|\right)$; thinking of g as a function from $\mathbf{R}^4$ into $\mathbf{R}^3$ our desired points now form the contour $g^{-1}(0, 0, 1)$. Unless $k = 1$, the value $(0, 0, 1)$ is not regular.

The interest in the contours $g^{-1}(0, 0, 1)$ lies in that their union for all $k \geq 1$ is related to the *Mandelbrot set* M [Blanchard 1984]; more exactly, the projection of this union onto the c-coordinate plane is the boundary of the interior of M. This, admittedly, is not a good method to draw M, because $g^{-1}(0, 0, 1)$ is not connected for $k > 2$, so in addition to tracing the contours we must worry about starting points. Still, it is instructive to watch the algorithm drawing the largest component of $g^{-1}(0, 0, 1)$, which contains the point $g(\frac{1}{4}, 0, \frac{1}{2}, 0) = (0, 0, 1)$ for all k: see figure 9, where the projections of this largest component onto each of the two coordinate planes are shown side by side for several choices of k.

For example, the contour tracer begins at $(\frac{1}{4}, 0)$ in Figure 9(c) and at $(\frac{1}{2}, 0)$ in Figure 9(d). As it sweeps out the top half of the cardioid in (c), it traverses the top

thickness: greater thickness means more lattice points would be projected, yielding a denser triangulation. The thickness would be controlled by the desired resolution. The basic drawback with this approach is that the computations involved in passing from one simplex to the next are a good deal trickier.

Finally, there is the matter of finding starting points. There is much literature on this subject (see, for example, [Ostrowski 1966], [Ortega and Rheinboldt 1970] and references therein), and here we limit ourselves to briefly describing a simple method that blends well with the rest of the algorithm; it does not intend to be optimally efficient. The idea is to restrict the search space first to one dimension, until we find a zero of some component of f; then to two dimensions, where we trace the contour of this component until finding a zero of another one; and so on until we find a point where all the components of f vanish. Notice that this is very similar to what we did in 6.2, where we traced the contour of some components of f on a subset of $\mathbf{R}^n$ of appropriate dimension.

10. Bibliography

E. L. Allgower and K. Georg, "Simplicial and continuation methods for approximating fixed points and solutions to systems of equations," *SIAM Review* **22** (1980), 28–85.

E. L. Allgower and P. H. Schmidt, "An algorithm for piecewise linear approximation of an implicitly defined manifold," *SIAM Journal of Numerical Analysis* **22** (1985), 322–346.

D. S. Arnon, "Topologically reliable display of algebraic curves," *ACM SIGGRAPH Computer Graphics* **17** (1983), 219–227.

P. Blanchard, "Complex analytic dynamics on the Riemann sphere," *Bulletin of the American Mathematical Society* **11** (1984), 85–141.

J. H. Conway and N. J. A. Sloane, "Fast quantizing and decoding algorithms for lattice quantizers and codes," *IEEE Transactions on Information Theory* **28** (1982), 227–232.

H. S. M. Coxeter, "Discrete groups generated by reflections," *Annals of Mathematics* **6** (1934), 13–29.

H. S. M. Coxeter, *Regular Polytopes*, third edition, Dover Publications, New York, 1973.

J. J. Dongarra, J. R. Bunch, C. B. Moler, G. W. Stewart, *LINPACK Users' Guide*, SIAM, Philadelphia, 1979.

M. Duneau and A. Katz, "Quasiperiodic patterns and icosahedral symmetry," *J. Physique* **47** (1986), 181–196.

J. D. Foley and A. van Dam, *Fundamentals of Interactive Computer Graphics*, Addison-Wesley, Reading, Mass., 1982.

A. Geisow, "Surface Interrogations," Ph.D. Thesis, University of East Anglia.

CAPTIONS

Figure 1

Two approaches to the problem of tracing contours. In (a) derivative information is used to take a step, while in (b) local values of the function are used to guide the tracing.

Figure 2

Several examples of bad approximations. For each picture, C is on the left and $\tilde{C}$ on the right.

Figure 3

In (a), (b) and (c), we show the interaction of the triangulation with the contour of $xy = 0$ for three different relative positions. In (d) we show how introducing a bias toward 0 can repair the disconnectedness in (c): each vertex with a value less than $\frac{1}{2}$ or less (for a triangle size of 1) is declared to lie on the contour.

Figure 4

Coxeter diagrams for all simplices in all dimensions that triangulate by reflection.

Figure 5

Portions of the three possible triangulations by reflection of $\mathbf{R}^2$.

Figure 6

Finding the barycentric coordinate of the projection of a vertex onto the opposite face.

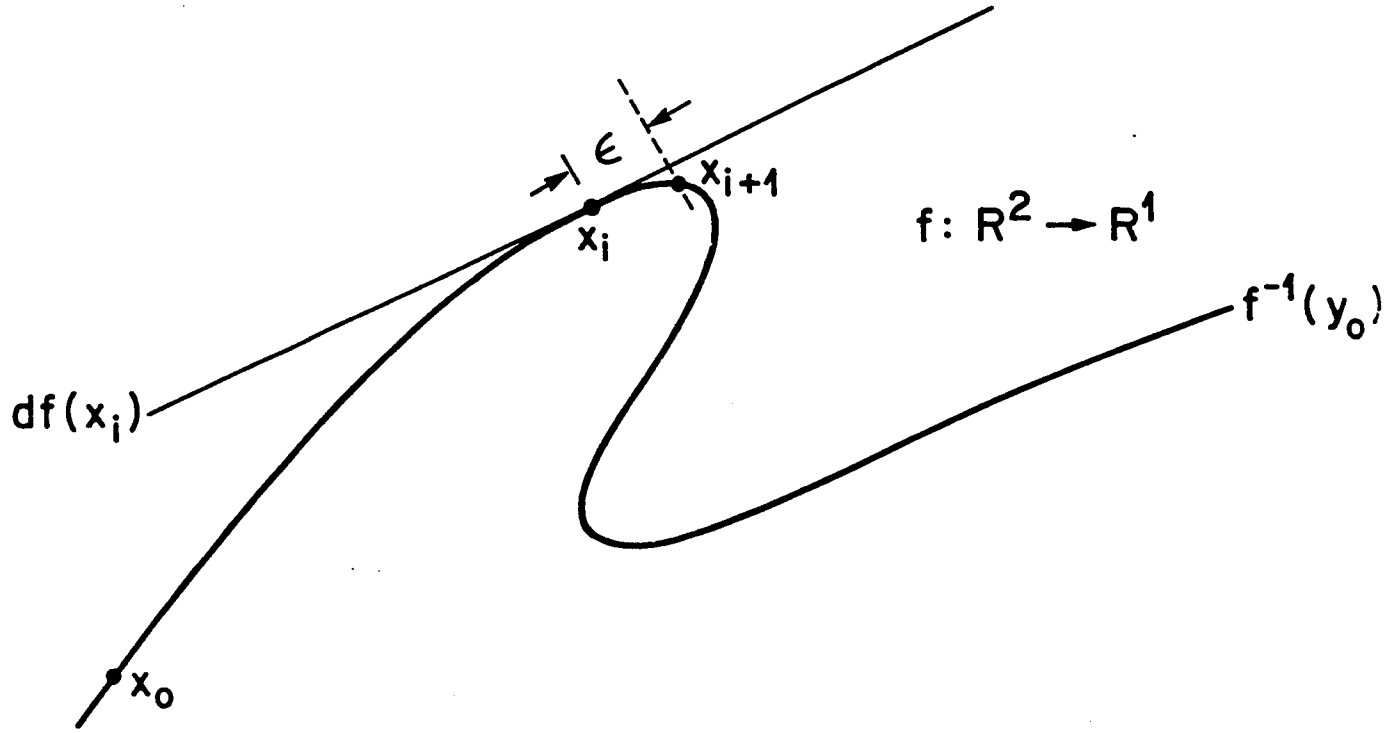
Figure 7

Two triangulations using R_3; (a) shows triangulation by reflection and (b) just replicates the squares. which is equivalent to using P_3 reflection rules on an R_3 triangle.

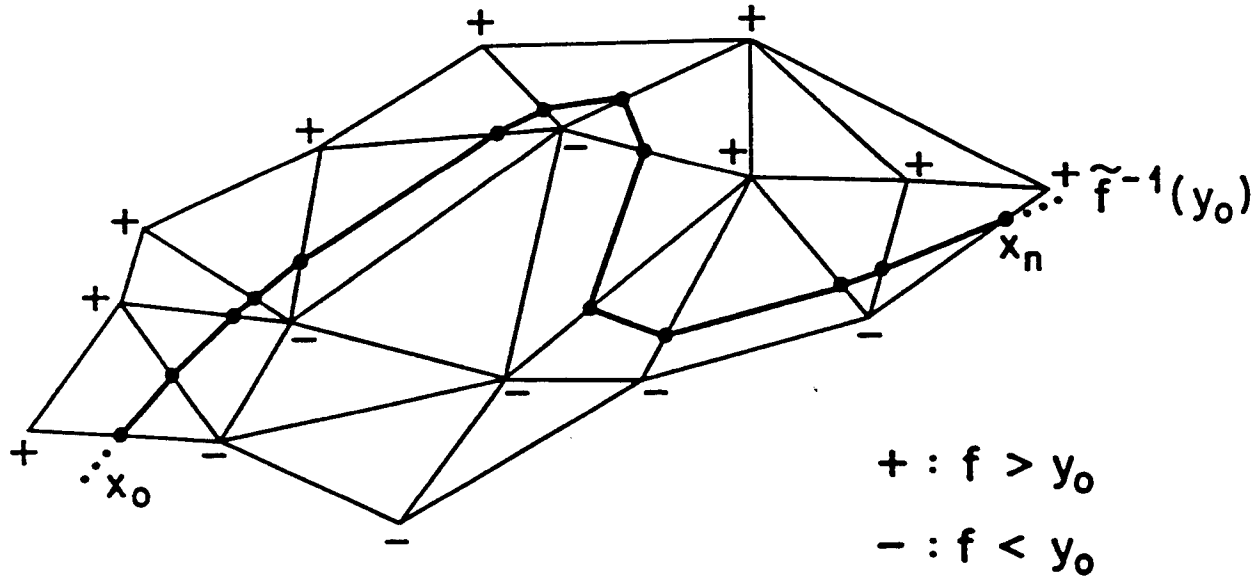
Figure 8

Two contours that approximate Julia sets. The parameter values are approximately (a) $c = -0.156546 + 1.03226i$ and (b) $c = -0.122561 + 0.744862i$. The boxes are 3 units wide in each direction. The R_3 triangulation was used, with step size (short sides of each triangle) equal to .0002.

Figure 1

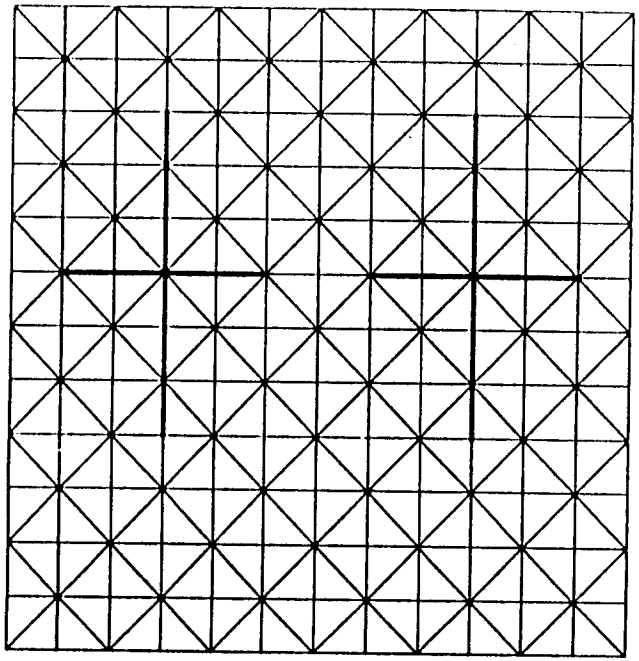

ϵ
x_{i+1}
x_i
$f: R^2 \rightarrow R^1$
$f^{-1}(y_o)$
$df(x_i)$
x_o

(a)

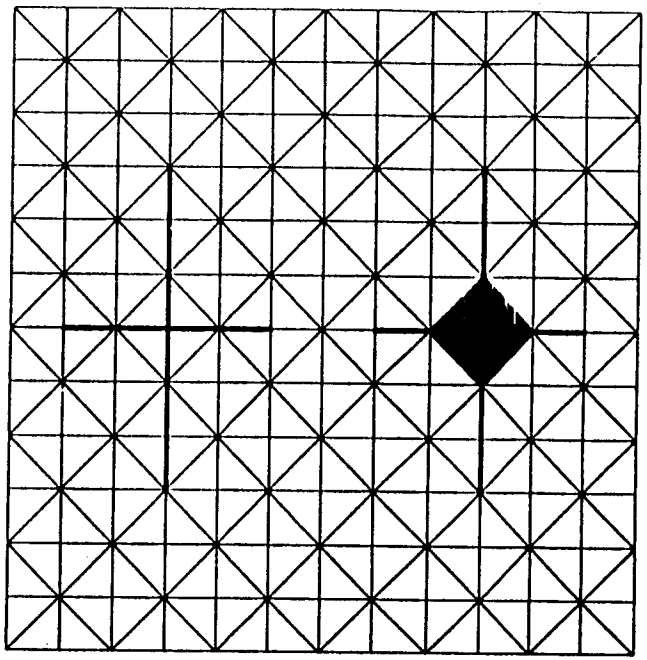

$\tilde{f}^{-1}(y_o)$
x_n
x_o
$+ : f > y_o$
$- : f < y_o$

(b)

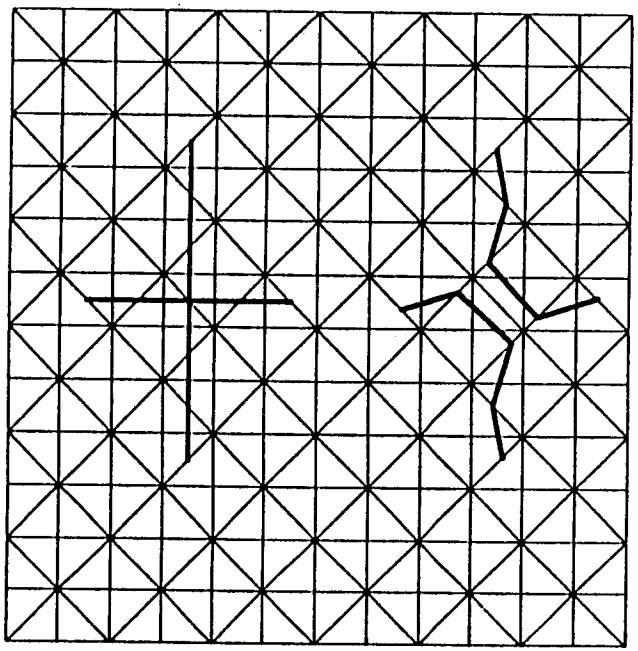
Figure 3.



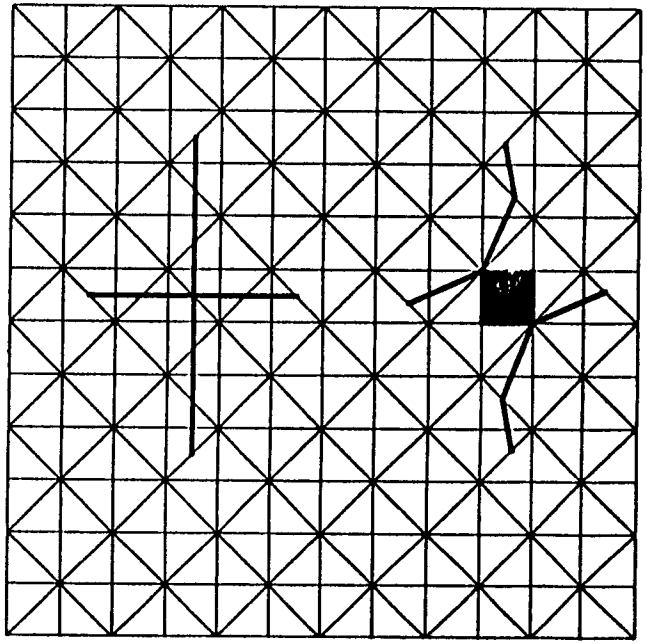
(a)



(c)

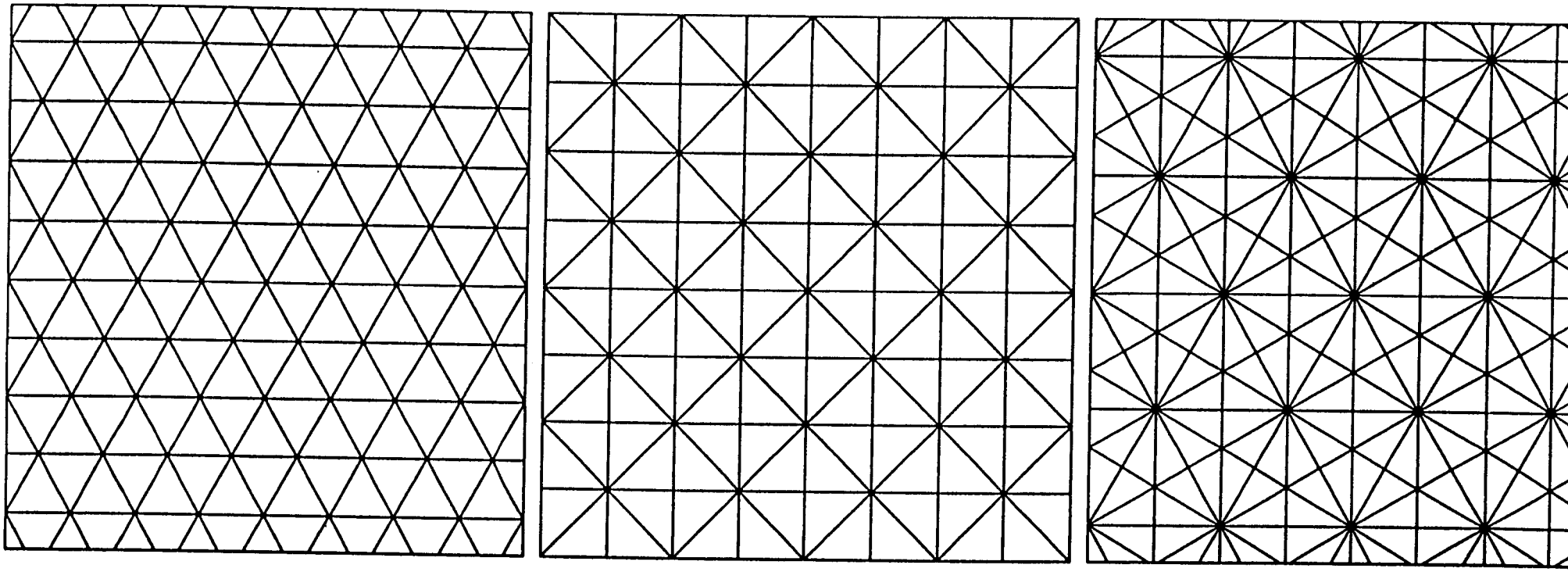


(c)



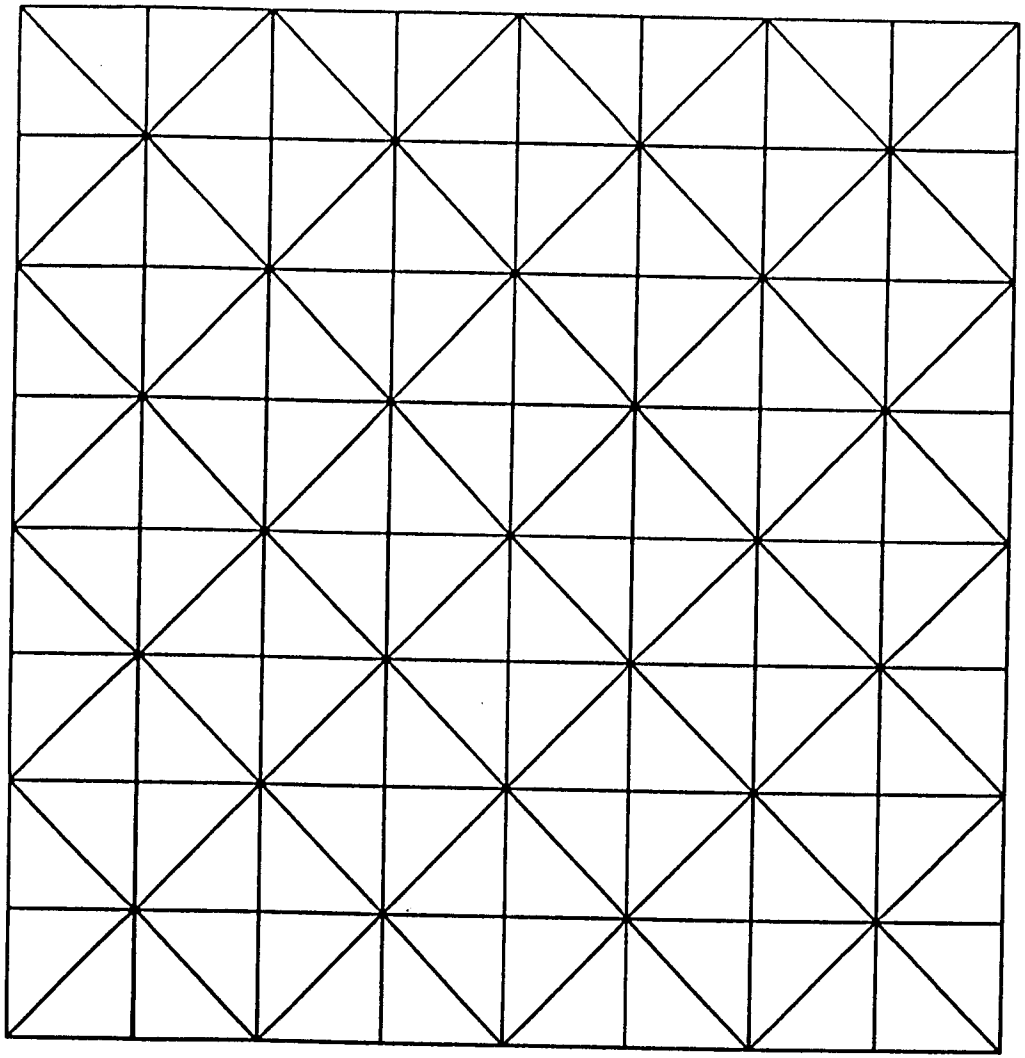
(d)

Figure 5.

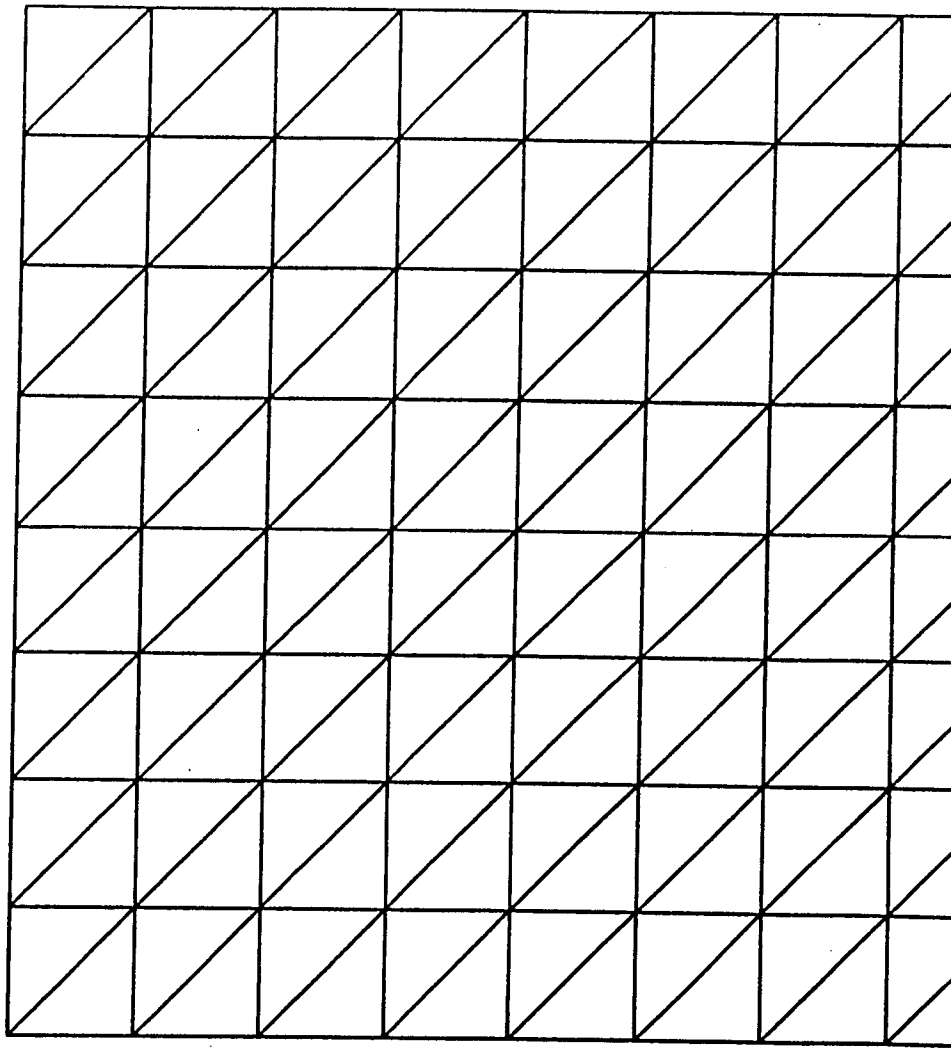


(a) (b) (c)

Figure 7.

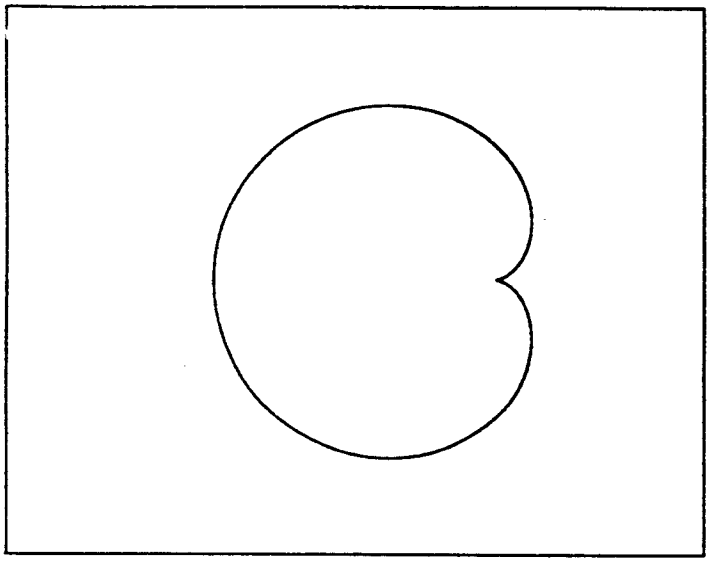


(a)

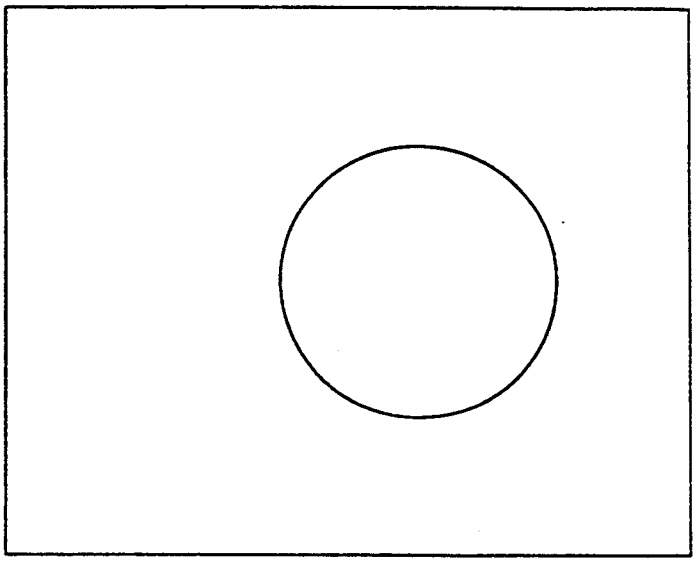


(b)

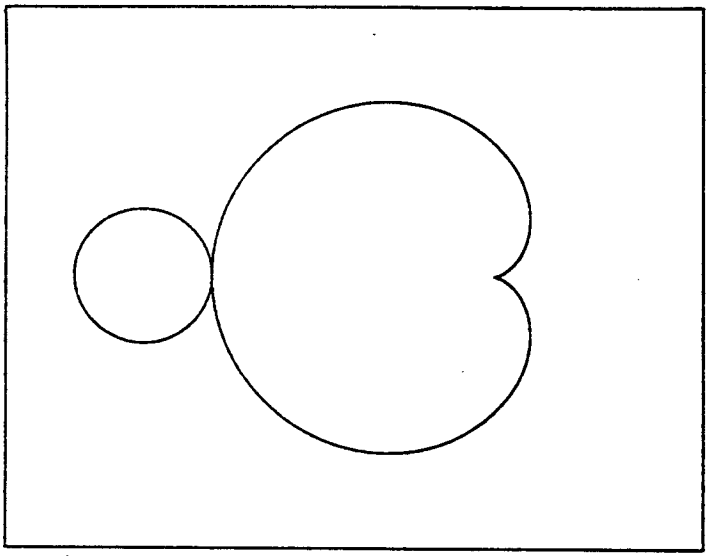
Figure 9.



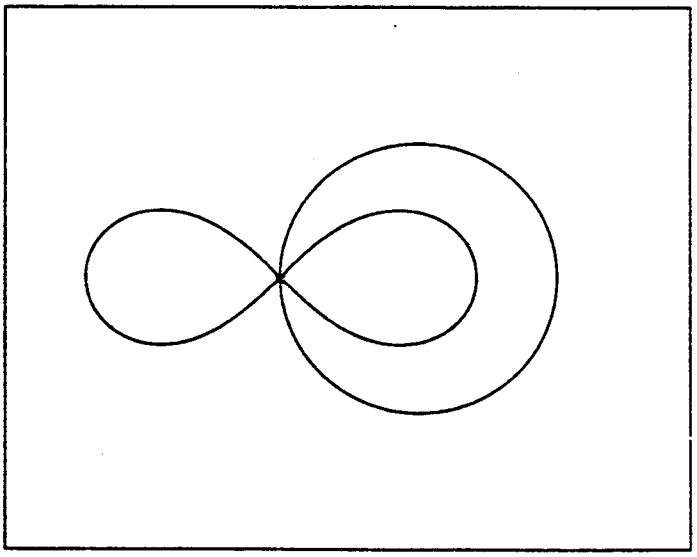
(a)



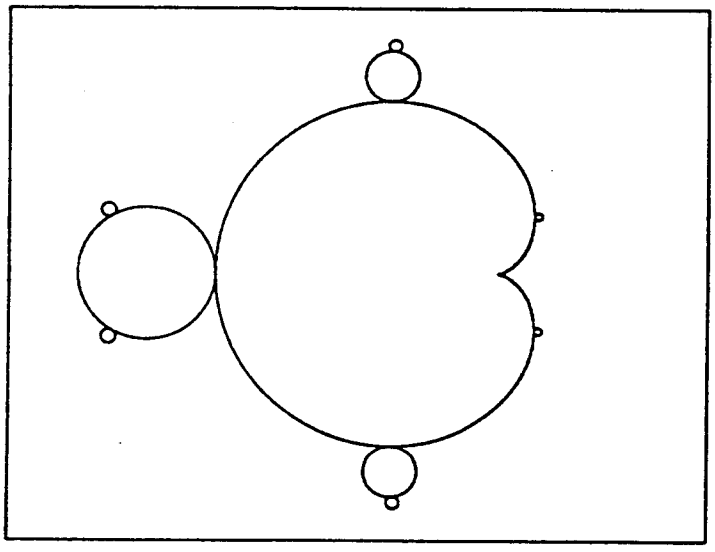
(b)



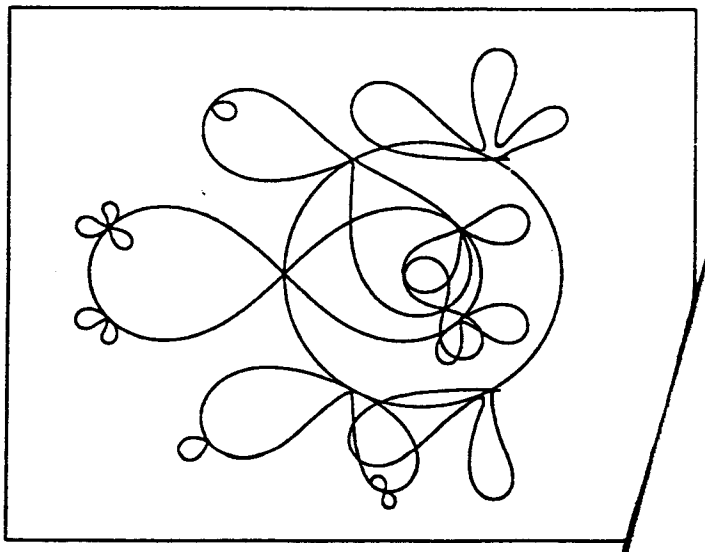
(c)



(d)



(e)



(f)