

Robust Ray Intersection with Interval Arithmetic

Don P. Mitchell

AT&T Bell Laboratories
Murray Hill, NJ 07974

1. Abstract

This paper discusses a very general root-isolation algorithm, based on interval arithmetic, which can find real roots in a large class of analytic functions. This algorithm has been used successfully to generate ray-traced images of a variety of implicit surfaces.

2. Introduction

One of the fundamental operations in ray tracing is the calculation of intersections between rays and primitive solids or surfaces. This generally reduces to the problem of solving for roots of an equation or a system of equations, and often the problem is nonlinear.

The details of the numerical problem depends to a large extent on how the primitive solid or surface is represented. An indirect definition of the points in a solid or surface is given in an *implicit surface*, defined by a scalar-valued function F :

$$\mathbf{S} = \{(x, y, z) | F(x, y, z) = 0\} \quad (1)$$

The other major type of representation gives an explicit formula for generating the points of the solid or surface, such as a patch of a *parametric surface*.

$$\mathbf{S} = \{(x(u, v), y(u, v), z(u, v)) \quad (2)$$

$$| u_0 \leq u \leq u_1, v_0 \leq v \leq v_1 \}$$

This paper is concerned with the case of implicit surfaces. In this case, the ray/solid intersection problem reduces to the problem of finding roots of a single equation in one variable. Given a ray represented parametrically by a starting point S and a direction vector \vec{D} , a simple substitution gives an equation in

the ray parameter t :

$$\begin{aligned} F(x, y, z) &= F(S_x + tD_x, S_y + tD_y, S_z + tD_z) \\ &= f(t) \end{aligned} \quad (3)$$

The surface of this solid is represented by points where F is zero, and intersections of the ray with the surface correspond to parameter values where f is zero. If f is a polynomial of degree less than five, then closed-form expressions for the roots exist (alho it may not be a sound numerical method to use the formulae of Cardan or Ferrari to solve cubic and quartic equations).

For more general types of equations $f(t) = 0$, the problem of finding roots can be divided into two steps. First the roots must be *isolated* by finding intervals $[t_i, t_{i+1}]$ which are known to contain one and only one root of the function. Secondly, the intervals about each root can be *refined* by reducing the size of the isolating interval until the root is located as accurately as possible using machine arithmetic.

The problem of root isolation is the more difficult problem. The problem of root refinement is well understood, and efficient, stable algorithms exist for locating a single root within an interval. A refinement algorithm as simple as bisection could be used, but faster methods exist which are equally reliable [Press88].

If $f(t)$ is a polynomial, there are a number of root finding methods. To find all of the complex roots of a polynomial, the Madsen-Reid algorithm is a current favorite [Madsen75].

For finding real roots of polynomials, root isolation methods based on Rolle's theorem, Budan's theorem, Descartes' rule of signs, and Sturm's theorem have all

been demonstrated [Collins82]. Some of these algorithms have been applied to ray tracing algebraic surfaces. Hanrahan has used a method based on Descartes' rule [Hanrahan83], and Sturm's theorem has been used by two others [Wijk84, Duff88]. Duff reports that his implementation of a Sturm-sequence root finder was much faster than the Madsen-Reid algorithm and also faster than the method used by Hanrahan.

For functions more general than polynomials, there are fewer results. An heuristic for root isolation has been proposed which estimates bounds on the value of a function in an interval from samples [Jones78]. This heuristic can fail, but other methods are based on guaranteed upper and lower bounds on a function.

Very few attempts have been made to ray trace non-algebraic surfaces. Blinn demonstrated an heuristic method for ray tracing Gaussian density distributions [Blinn82]. Kalra and Barr have demonstrated a robust root isolation algorithm that works on functions for which a Lipschitz condition for f and its derivative can be found within given intervals of the range [Kalra89].

3. The Interval Root Isolation Algorithm

Interval analysis has proven successful for finding real roots of systems of nonlinear equations [Kearfott87]. Toth used such an algorithm to ray trace parametric surfaces [Toth85]. The algorithm he used was based on the idea of subdividing parameter space until safe starting regions were found for an iterative root-refinement method.

Ray tracing implicit surfaces presents an easier, one-dimensional problem. For this case, Moore gives a simple and general algorithm for root isolation which can be applied to rational functions and also to functions involving familiar transcendental functions [Moore66].

An *interval number* is represented by a lower and upper bound, $[a, b]$ and corresponds to a range of real values. An ordinary real number can be represented by a degenerate interval $[a, a]$. It is straightforward to define basic arithmetic operations on interval numbers:

$$[a, b] + [c, d] = [a + c, b + d] \quad (4)$$

$$[a, b] - [c, d] = [a - d, b - c]$$

$$[a, b] * [c, d] = [\min(ac, ad, bc, bd),$$

$$\max(ac, ad, bc, bd)]$$

and if $0 \notin [c, d]$

$$[a, b] / [c, d] = [a, b] * [1/d, 1/c]$$

Using the above rules, a rational expression $r(x, y, z)$ can be evaluated with interval values $[x_0, x_1], [y_0, y_1], [z_0, z_1]$ for its variables. The resulting value may be an interval that is much wider than the actual range of the corresponding real-valued expression, but it is guaranteed to bound that range. That is, for intervals X, Y, Z :

$$r(X, Y, Z) \supseteq \{r(x, y, z) \mid x \in X, y \in Y, z \in Z\} \quad (5)$$

As the intervals X, Y, Z become narrower, a rational expression converges toward its corresponding real restriction:

$$X' \subset X, \text{ implies } r(X') \subseteq r(X) \quad (6)$$

and

$$X = [x, x] \text{ implies } r(X) = [r(x), r(x)]$$

Using these concepts, a simple recursive algorithm can be described for isolating the roots of a function of one variable. We start with a rational function r and an initial interval $[a, b]$.

Step 1. Evaluate $r([a, b])$. If the resulting interval value does not contain zero, then there cannot be a root in $[a, b]$, and we are finished with this interval.

Step 2. Evaluate the derivative $r'([a, b])$. If the resulting interval value does not contain zero, then the function must be monotonic in the interval. If the function is monotonic and $r(a)r(b) \leq 0$, then there is a root in the interval which can be refined by some standard method.

Step 3. If $r([a, b])$ and $r'([a, b])$ both contain zero, then subdivide the interval at its midpoint and recursively process $[a, (a+b)/2]$ and $[(a+b)/2, b]$.

Step 4. The process of subdivision should be stopped when the width of an interval approaches the machine accuracy. For example, if the midpoint tests equal to either endpoint on the machine or if the width of the interval is less than some minimum allowed value.

We see that this algorithm is based on the existence of a *root inclusion test*, which checks for the presence of a single root in an interval. The root inclusion test can return a value of "yes", "no" or "maybe", and bisection is performed when the result is "maybe". Many root-isolation algorithms conform to this paradigm, whether they use Descartes's Rule, Lipschitz conditions, or interval arithmetic to test for root inclusion.

The algorithm above is slightly different than the one described by Moore which assumes that root refinement will also be performed by an interval algorithm.

A discussion of Moore's algorithm should include the very important issue of machine arithmetic and round-off error. I have found that ordinary rounded floating point arithmetic (in single precision) and ordinary root-refinement algorithms are sufficient to produce the images presented below. However, strict bounds on function variation can be computed even with finite precision floating point arithmetic if it is done with outward rounding. This is safer but more computationally expensive.

4. Application of the Algorithm to Ray Tracing

In the context of ray tracing an implicit surface, the interval root isolation algorithm is well suited to finding zeros of the function $f(t)$ in (3).

Given the three-dimensional surface $F(x, y, z) = 0$, it may be straightforward to derive a closed-form expression for $f(t)$ and its derivative $f'(t)$. If so, the root finding algorithm can be applied directly to the interval extension $f([t_0, t_1])$.

If $f(t)$ cannot be easily derived in closed form, it is possible to work directly with the interval extension of $F(x, y, z)$. Given a ray defined by a starting point S and a direction \bar{D} , an interval $[t_0, t_1]$ can be substituted into (3) to evaluate the resulting $F([x_0, x_1], [y_0, y_1], [z_0, z_1])$.

Similarly, $f'(t)$ can be derived by taking the interval extension of the directional derivative of F :

$$f'(t) = \bar{D} \cdot \nabla F(x, y, z) \quad (7)$$

Given an expression $F(x, y, z)$ in symbolic postfix form, a simple interpreter can compute an interval evaluation. By application of the chain rule for differentiation, the value of $\nabla F(x, y, z)$ can be computed concurrently.

The interval root isolation algorithm was described for rational functions, but it is straightforward to extend this to include most of the familiar transcendental functions. For monotonic functions, the interval extension is trivial:

$$e^{[a, b]} = [e^a, e^b] \quad (8)$$

$$[a, b]^3 = [a^3, b^3]$$

For modeling superquadrics, the absolute-value function is needed, and its interval extension is simply:

$$|[a, b]| = [0, \max(|a|, |b|)] \quad (9)$$

Many commonly-used transcendental functions like sine and cosine are made up of monotonic segments with minima and maxima at known locations. That information is sufficient to compute exact upper and lower bounds of an interval extension of a function.

When a ray grazes the surface $F(x, y, z) = 0$ at a tangent, the corresponding root of $f(t) = 0$ will be a multiple root (i.e., the value and some number of the derivatives of the function will all be zero at the same point). As the interval root isolation process converges on a multiple root, the derivative will always be zero in the interval, so the algorithm will not terminate until it reaches a minimum-sized interval (in Step 4 of the algorithm). The algorithm will succeed, but like many root-finding methods, it is slower in finding multiple roots.

In order to ray trace concave superquadrics [Barr81], it is also necessary to deal with singularities in the derivative. This is because a function such as

$$r(x) = |x|^{0.75} \quad (10)$$

has a singularity in its derivative at $x = 0$. The singularity results from dividing by zero, and the interval extension of the division operation must be modified

to return some representation of $[-\infty, \infty]$ in this case.

5. Results

In Plate 1, a fourth-degree algebraic surface is rendered with this method. The equation for this surface is:

$$4(x^4 + (y^2 + z^2)^2) + 17x^2(y^2 + z^2) - 20(x^2 + y^2 + z^2) + 17 = 0 \quad (11)$$

Plate 2 shows an example of a non-algebraic analytic surface—a sum of five Gaussians representing an arrangement of atoms. Plate 3 is a concave superquadric of the form:

$$|x|^{0.75} + |y|^{0.75} + |z|^{0.75} = 1 \quad (12)$$

This illustrates a surface with singular gradients at some points. Concave super quadrics are difficult objects to render by direct intersection alone. With even smaller exponents, the "webbing" between the corners becomes so thin that even double precision arithmetic may not be sufficient to isolate the roots correctly or to compute the gradient accurately (in such close proximity to a singularity). It would be interesting to see if a careful application of round-out interval arithmetic could handle such pathological cases.

Plate 4 is the same surface as in Plate 3, with a twist deformation [Barr84]:

$$x' = x \cos(4y) - z \sin(4y) \quad (13)$$

$$z' = x \sin(4y) + z \cos(4y)$$

In all of these figures, intersection with a simple bounding box provides a starting interval for the root-isolation algorithm. Better starting intervals might be obtained by the octree spatial subdivision described by Kalra and Barr [Kalra89] which could be easily modified to use interval analysis.

Test images were generated with an experimental ray-tracing system running on a SPARCstation 330. Using this algorithm, a simple unit sphere was rendered with 2.7 msec/ray. That compares to 0.78 msec/ray to render a sphere using the usual methods of solving quadratic equations.

For more complex surfaces, the speed of the algorithm varies from function to function. For some expressions, the interval bounds converge more slowly as the intervals are subdivided. Thus a relatively simple algebraic surface may require more time than some non-algebraic surfaces. The convergence of the interval bounds is effected not only by the function, but also the particular form of the expression. In particular, it is useful to keep in mind the *subdistributive* property of interval arithmetic:

$$X(Y + Z) \subset XY + XZ \quad (14)$$

A CSG ray tracer may need to compute all intersections of the ray with the surface in order to perform set operations [Roth82]. In the case of boundary-representation schemes or CSG models using just union set operations, only the closest intersection is needed to find the visible surface. Some representative times are given below for finding all roots or just the root closest to the ray origin:

Rendering Time in Milliseconds per Ray		
Object	Find All	Closest
Sphere	2.7	1.9
Quartic (Plate 1)	33.2	17.9
Gaussians (Plate 2)	23.1	11.0
Superquadric (Plate 3)	6.4	4.4
Twisted SQ (Plate 4)	21.3	13.6

6. Conclusions

Few methods are available for reliably ray tracing non-algebraic implicit surfaces. Root isolation by interval analysis is a simple and general way to find real roots of nonlinear equations. This method has been used to ray trace an interesting variety of algebraic and non-algebraic implicit surfaces. It is effective even for surfaces which have singularities in their gradients.

One advantage of the interval algorithm is that it does not require a mathematical analysis of new surfaces to determine Lipschitz constants—in fact the symbolic expression for a new surface is entered at run time in my implementation.

Several improvements should be made. Preprocessing an object to find a tight-fitting octree boundary (as in [Kalra89]) might improve performance. And in general, the times shown above come from code which is has not been squeezed for performance. Most time is spent in the interval multiply routine which, unnecessarily, performs four multiplications for every call.

7. Acknowledgements

I would like to thank Eric Grosse for urging me to give Moore's algorithm a try. Thanks to John Amanatides who worked with me on the "FX" ray tracing system used to test this algorithm.

8. References

- [Barr81] Barr, Alan H., "Superquadrics and Angle-Preserving Transformations", *IEEE Computer Graphics and Applications*, January, 1981.
- [Barr84] Barr, Alan H., "Global and Local Deformations of Solid Primitives", *SIGGRAPH 84*, July, 1984.
- [Blinn82] Blinn, James F., "A Generalization of Algebraic Surface Drawing", *ACM Transactions on Graphics*, July, 1982, pp 235-256.
- [Collins82] Collins, G. E., R. Loos, "Real Zeros of Polynomials", *Computing Suppl.* Vol. 4, pp 83-94, 1982.
- [Duff88] Duff, Tom, "Using Sturm Sequences for Rendering Algebraic Surfaces", *unpublished report* July 25, 1988.
- [Hanrahan83] Hanrahan, Pat, "Ray Tracing Algebraic Surfaces", *SIGGRAPH 83*, July 1983, pp 83-89.
- [Jones78] Jones, Bush, *et al*, "Root Isolation Using Function Values", *BIT* Vol. 18, 1978, pp. 311-319.
- [Kalra89] Kalra, Devendra, Alan H. Barr, "Guaranteed Ray Intersection with Implicit Surfaces", *SIGGRAPH 89*, July, 1989, pp 297-306.
- [Kearfott87] Kearfott, R. B., "Abstract generalized bisection and a cost bound", *Math. Comput.*, Vol. 49, No. 179, July, 1987, pp. 187-202.
- [Madsen73] Madsen, K., "A root-finding algorithm based on Newton's method", *BIT* Vol. 13, pp 71-75, 1973.
- [Moore66] Moore, Ramon E., *Interval Analysis*, Prentice-Hall, Englewood Cliffs, NJ (1966).
- [Press88] Press, William H., *et al*, *Numerical Recipes in C*, Cambridge University Press, (1988).
- [Toth85] Toth, Daniel L., "On Ray Tracing Parametric Surfaces", *SIGGRAPH 85*, July 1985, pp 171-179.
- [Wijk84] van Wijk, Jarke J., "Ray tracing objects defined by sweeping a sphere", *Eurographics 84*, September 1984, pp 73-82.