

# **New approach to point-to-curve raytracing**

**Tijmen Jan Moser**

**Fugro-Jason, The Netherlands**

**29 September 2004**

**Charles University, Prague, Czech Republic**

## Biography

---

1987 - masters degree Geophysics, State University of Utrecht, The Netherlands  
(Nolet)

1992 - PhD, "Shortest path method for seismic ray tracing in complicated media"  
State University of Utrecht, The Netherlands  
(Helbig, Nolet)

1992 - Postdoc Amoco, Tulsa, OK  
(Gray, Treitel)

1992-4 - Institut Français du Pétrole and Institut de Physique du Globe, Paris  
(Lailly, Madariaga, Tarantola)

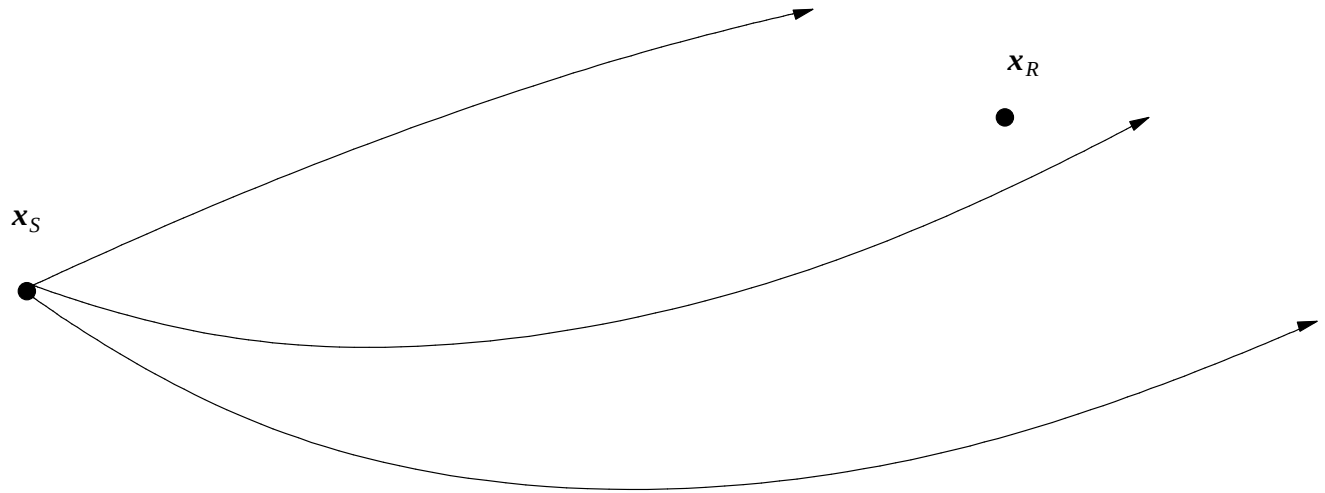
1994-5 - Institute of Solid Earth Physics and Norsk Hydro  
(Hanyga, Pajchel)

1996-7 - Alexander von Humboldt fellow - Karlsruhe  
(Hubral, Karrenbach, Shapiro)

1997-2000 - Geophysical Institute of Israel/Norsk Hydro  
(Landa, Keydar, Gelchinsky, Pajchel)

2001-present - Fugro-Jason

## Two-point raytracing



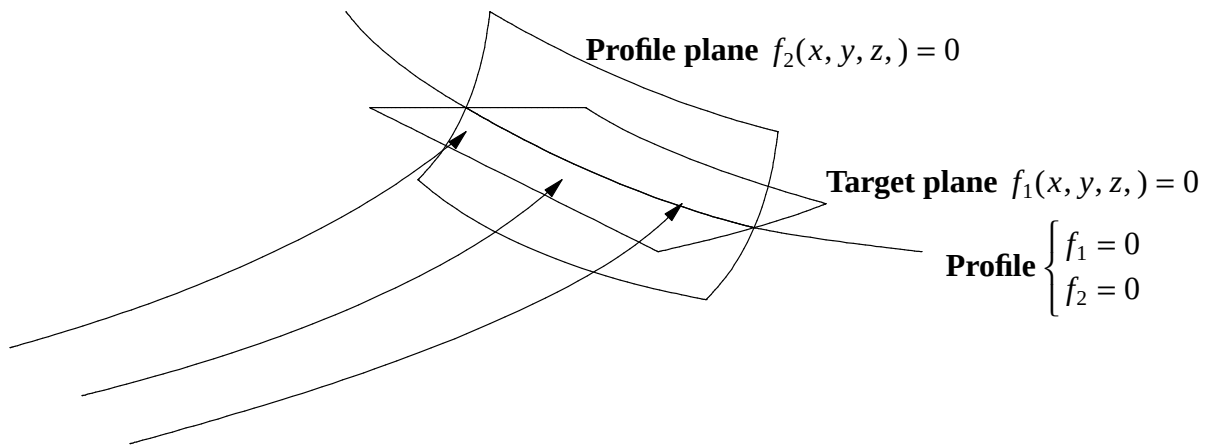
## 3D Ray field

$$\mathbf{x}(\gamma_1, \gamma_2, \tau) = \mathbf{x}_R$$

$\gamma_1, \gamma_2$  initial angles,  $\tau$  independent monotonic parameter along the ray

nonlinear system of 3 equations with 3 unknowns

Point-to-curve raytracing (Hanyga and Pajchel, GP, 1995)



3D Ray field up to target plane

$$f_1(\mathbf{x}(\gamma_1, \gamma_2, \tau)) = 0$$

(solve for  $\tau_0 = \tau$ )

Distance from profile plane

$$f_2(\mathbf{x}(\gamma_1, \gamma_2, \tau_0))$$

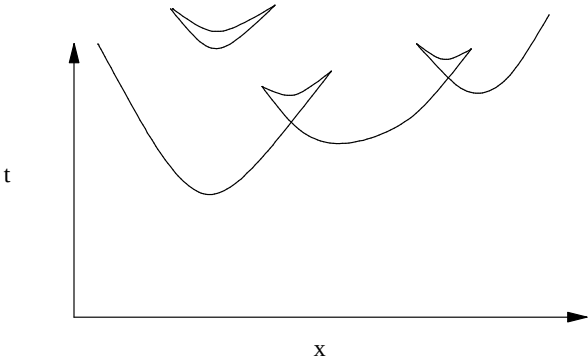
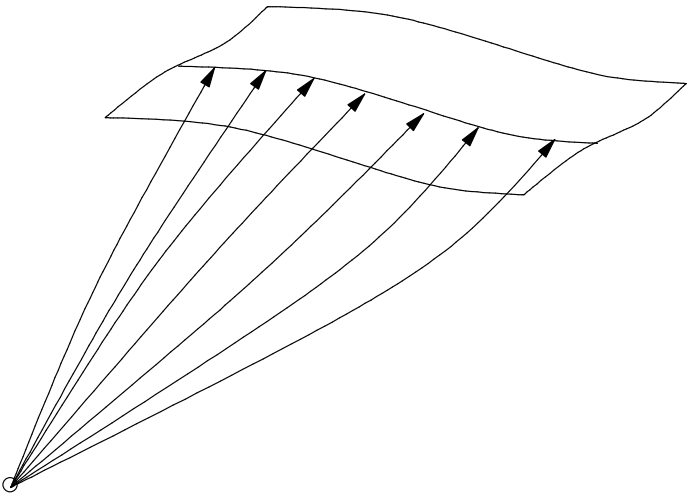
→

$$F(\gamma_1, \gamma_2) = f_2(\mathbf{x}(\gamma_1, \gamma_2, \tau_0)) = 0$$

nonlinear system of 1 equation with 2 unknowns

implicit plane curve

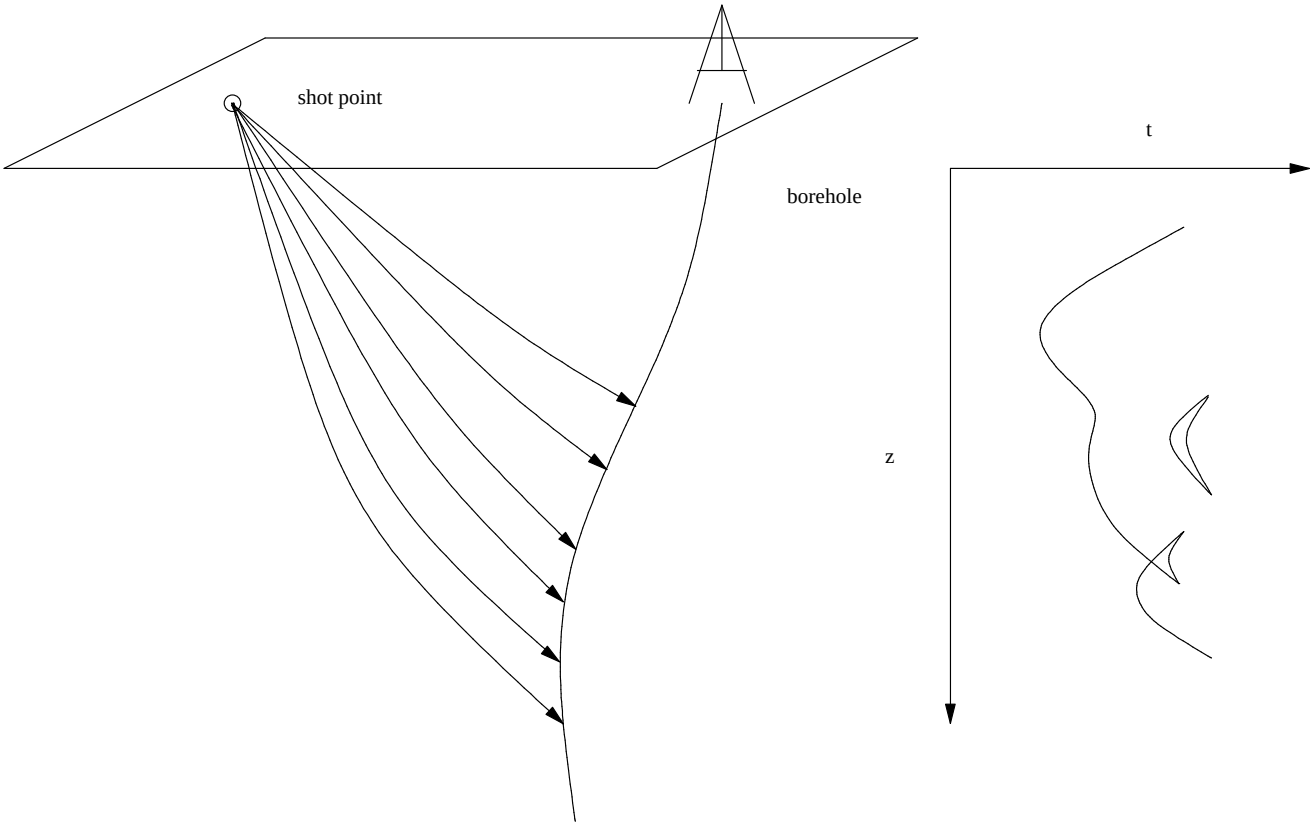
**Horizontal shot profiling**



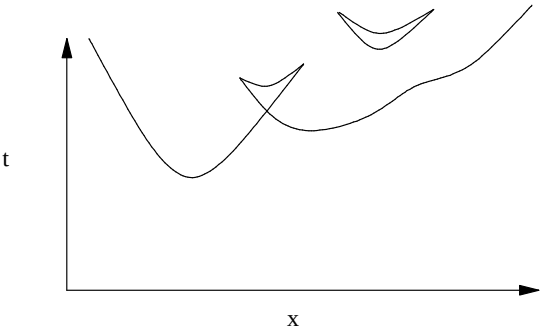
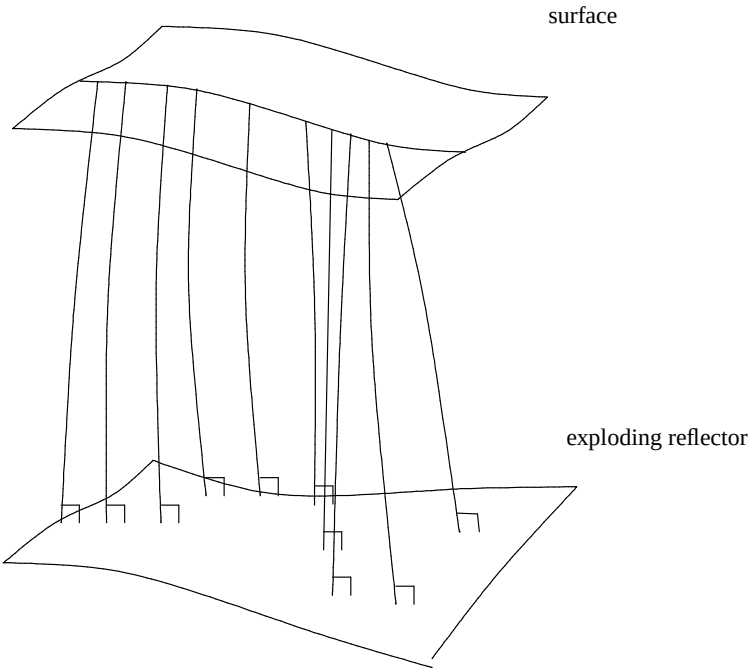
surface topography

shot point

Vertical seismic profiling (VSP)



**Exploding reflector modeling**



Solution of  $F(\gamma_1, \gamma_2) = 0$

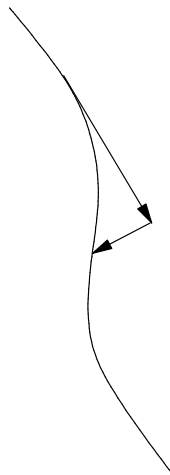
- implicit curve tracing

$$\frac{d}{dq} \begin{pmatrix} \gamma_1 \\ \gamma_2 \end{pmatrix} = \begin{pmatrix} F_{\gamma_1} \\ F_{\gamma_2} \end{pmatrix}$$

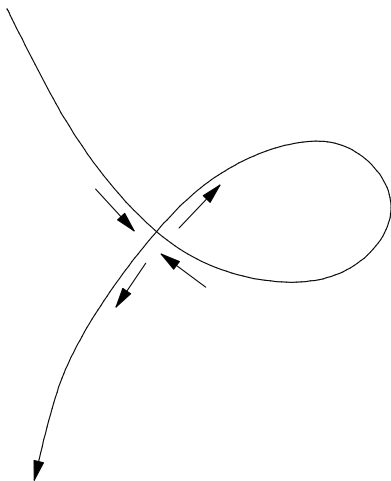
$q$  parameter along curve.

Predictor-corrector, because first integral is known.

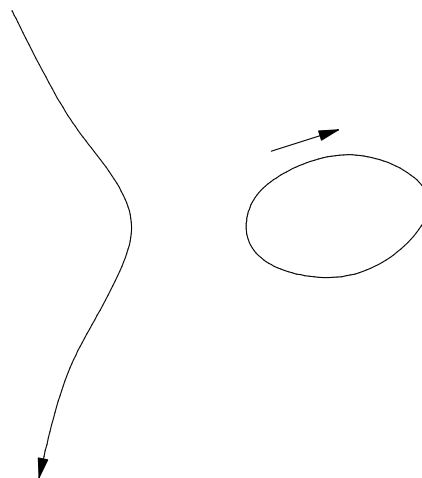
Predictor-corrector



Bifurcation

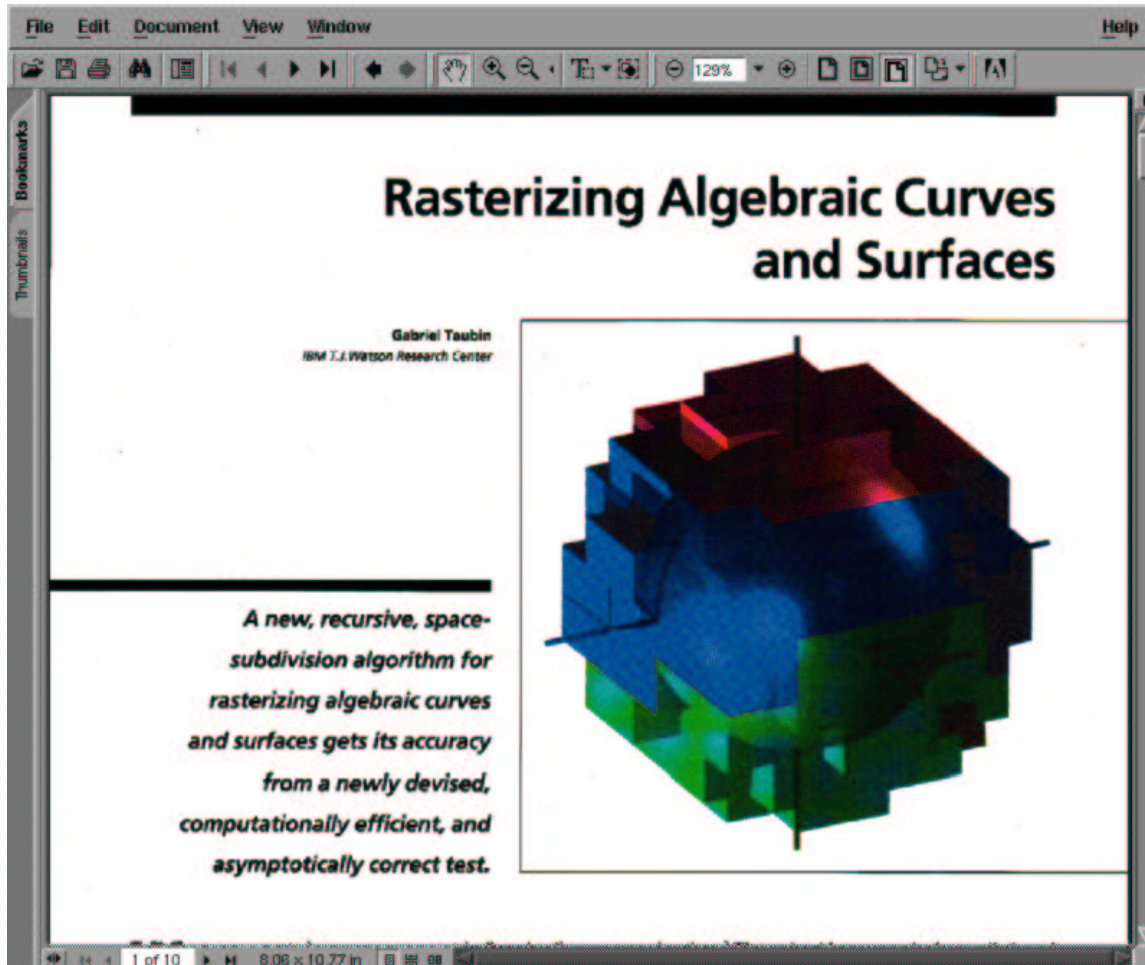


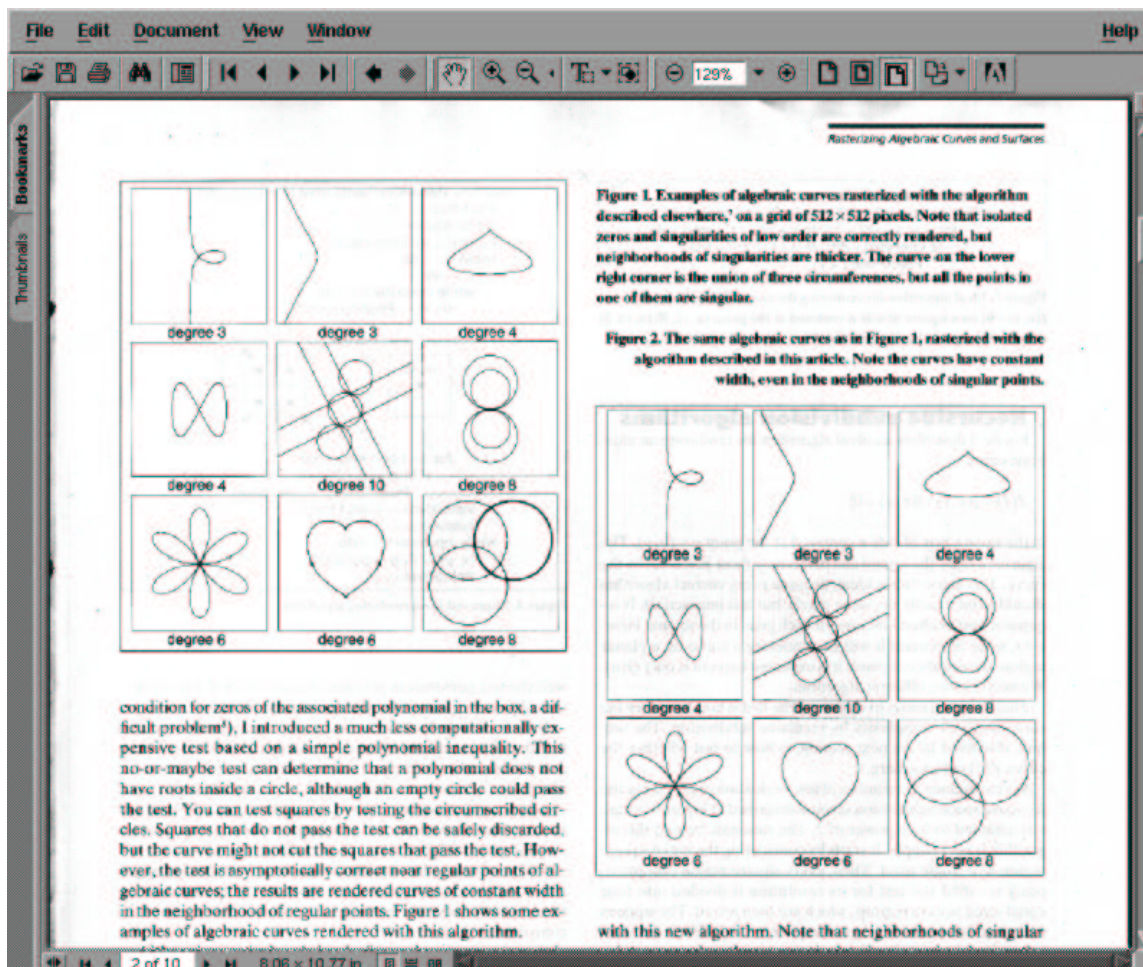
Isolated branches



- Contouring of  $F(\gamma_1, \gamma_2)$
- Algebraic rasterization

(An Accurate Algorithm for Rasterizing Algebraic Curves and Surfaces, by G. Taubin.  
IEEE Computer Graphics & Applications, March 1994.)





Solid Modeling

```

procedure IdealPointCurve (f, B((u,v), n/2))
  for x ← u - n/2 + 1/2 to u + n/2 - 1/2 step 1 do
    for y ← v - n/2 + 1/2 to v + n/2 - 1/2 step 1 do
      if intersect (Z(f), B((x,y), 1/2)) ≠ NO
        PaintPixel (x,y)

```

Figure 3. Ideal algorithm for rendering the curve  $Z(f) = \{(x, y) : f(x, y) = 0\}$  on a square of side  $n$  centered at the point  $(u, v)$ .  $B((u, v), \delta)$  denotes the square box  $\{(x, y) : \max(|x - u|, |y - v|) \leq \delta\}$ .

### Recursive subdivision algorithms

Figure 3 describes an ideal algorithm for rendering an algebraic curve

$$Z(f) = \{(x, y) : f(x, y) = 0\}$$

in the square box of side  $n$  centered at the point  $p = (u, v)$ . The algorithm scans the square and paints only those pixels cut by the curve. This algorithm is ideal, because every correct algorithm should paint exactly the same pixels, but it is impractical. It requires  $n^2$  test evaluations, one for each pixel in the square. However, since in general it will be rendering a curve on a planar region, the number of pixels it is expected to paint is only  $O(n)$ . We need a more efficient algorithm.

There are two issues to examine. The first is how to reduce the computational complexity by recursive subdivision. The second, discussed in the next section, is how to test whether the curve  $Z(f)$  cuts a square.

We can reduce the number of test evaluations easily using the recursive space-subdivision scheme described in Figure 4, where  $n$  is assumed to be a power of 2. The rationale behind this algorithm is very simple. It starts by considering the initial square region as a single pixel. Then, every square region that previously satisfied the test for its resolution is divided into four equal-sized square regions, which are then tested. The squares that pass the distance test are kept in a stack. Each iteration halves the sides of the squares; when the square regions become single pixels, the iteration stops and the pixels are painted.

### Tests for zeros of an algebraic curve

This section tackles the problem of testing whether the algebraic curve  $Z(f)$  defined by a polynomial of two variables  $f(x, y)$  intersects the box  $B((u, v), \delta) = \{(x, y) : \max(|x - u|, |y - v|) \leq \delta\}$ . Tests that give a yes or no answer, that is, tests for necessary and sufficient conditions for the curve to intersect the box, exist.<sup>8</sup> However, they are computationally very expensive, involving elimination techniques, and probably not numerically well behaved. You could use approximate tests instead; the main problem with approximate tests, though, is proving the correctness of the algorithm. An approximate test should satisfy three main properties. First, it should never discard a box that intersects the curve; otherwise, parts of the curve could be missing. That is, the test should be a sufficient condition for a curve not to intersect the box. Second, it should be asymptotically correct; that is, if a box does not intersect the curve and is not rejected by the test, after a number of subdivision steps all the resulting sub-boxes should be rejected. Third, and very much related to the previous property, it should render curves of approximately constant width.

The last two properties are difficult to achieve, in particular when close to singular points. There is also the issue of how

```

procedure RecursivePaintCurve (f, B((u,v), n/2))
  input-stack ← 0
  output-stack ← 0
  Push ((u, v), input-stack)
  while n > 1 do
    n ← n/2
    while input-stack ≠ 0 do
      (u, v) ← Pop(input-stack)
      for i = 1 to 4 step 1 do
        if intersect (Z(f), B((u, v_i), n/2)) ≠ NO
          Push ((u, v_i), output-stack)
      input-stack ← output-stack
      output-stack ← 0
    while input-stack ≠ 0 do
      (x, y) ← Pop(input-stack)
      PaintPixel (x, y)

```

Figure 4. Improved ideal rendering algorithm.

well the test performs in practice, because even if it is asymptotically correct, it could converge very slowly.

Now I will briefly describe a slightly modified version of a test that I recently introduced.<sup>9</sup> It satisfies the first property, and the second and third ones in the neighborhoods of the regular points of the curve. The test fails to discard enough pixels close to singular points, resulting in thicker regions. That is, the width of the rendered curve increases in the neighborhood of singular points. Figure 1 has already shown us examples of algebraic curves rendered with the recursive subdivision algorithm of Figure 4, using this test. I then modify the test further to correct its behavior near singular points. This is the main contribution of this article.

Note that by first translating the origin to  $(u, v)$ , that is, by expanding the polynomial in Taylor series around  $(u, v)$ , the problem reduces to the case  $(u, v) = (0, 0)$ . An efficient algorithm for transforming the polynomial

$$f(x, y) = \sum_{0 \leq i, j \leq d} f_{ij} x^i y^j \quad (1)$$

where  $d$  is the degree of  $f$ , to the form

$$f(x, y) = f'(x - u, y - v) = \sum_{0 \leq i, j \leq d} f'_{ij} (x - u)^i (y - v)^j \quad (2)$$

is Horner's algorithm.<sup>11</sup> Going from Equation 1 to Equation 2 is equivalent to evaluating the polynomial and all its partial derivatives at  $(u, v)$ . Horner's algorithm requires  $O(m^2)$  operations, where  $m$  is the number of coefficients of the polynomial. To evaluate a polynomial in Bernstein form requires the same order of operations.

So, from now on we will concentrate on finding a sufficient condition for the polynomial in Equation 1 not to have zeros in the box  $B((0, 0), \delta) = \{(x, y) : \|(x, y)\|_\infty \leq \delta\}$ ; we can assume that  $f(0, 0) = f_{00} \neq 0$ , because otherwise the curve clearly intersects the box. The approach is as follows: Construct a polynomial of one

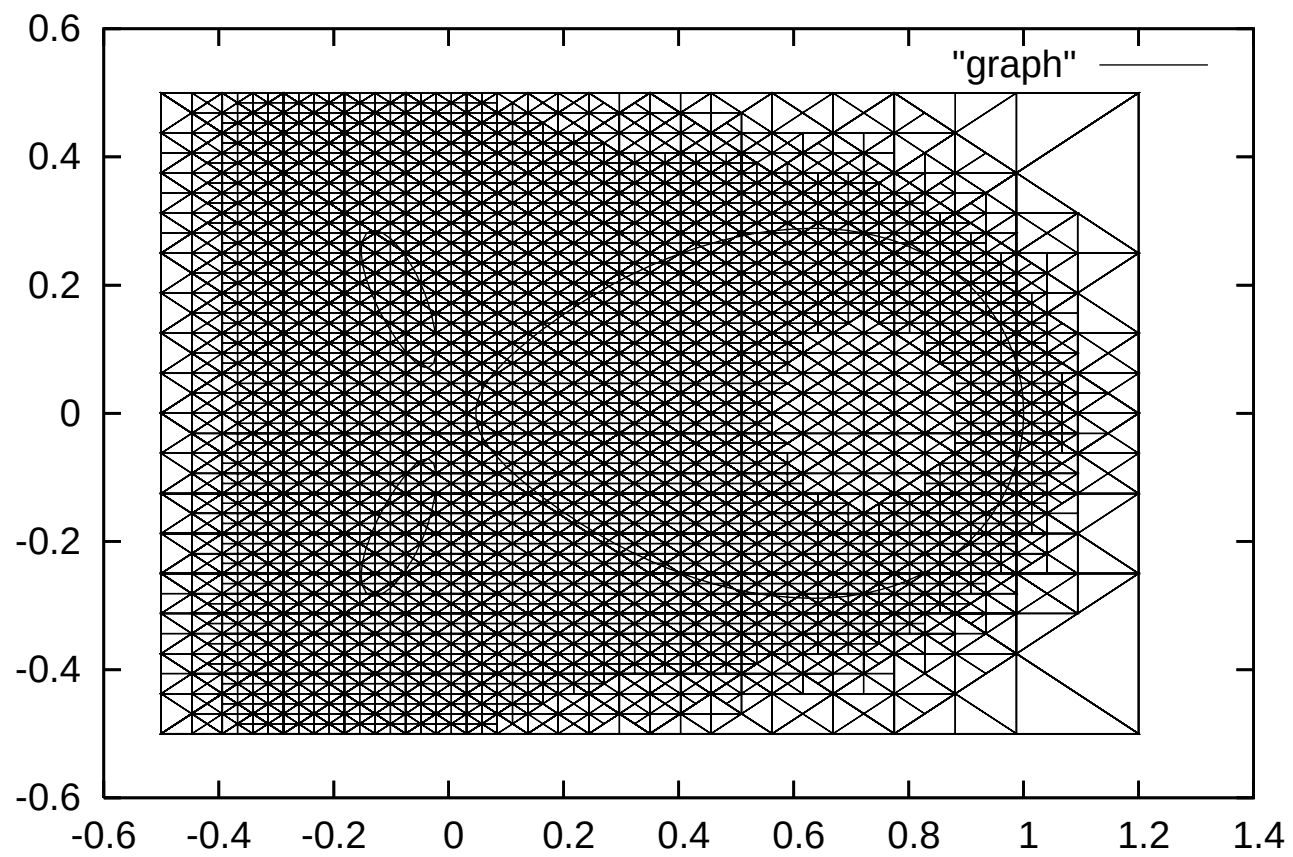
- Lipschitz condition

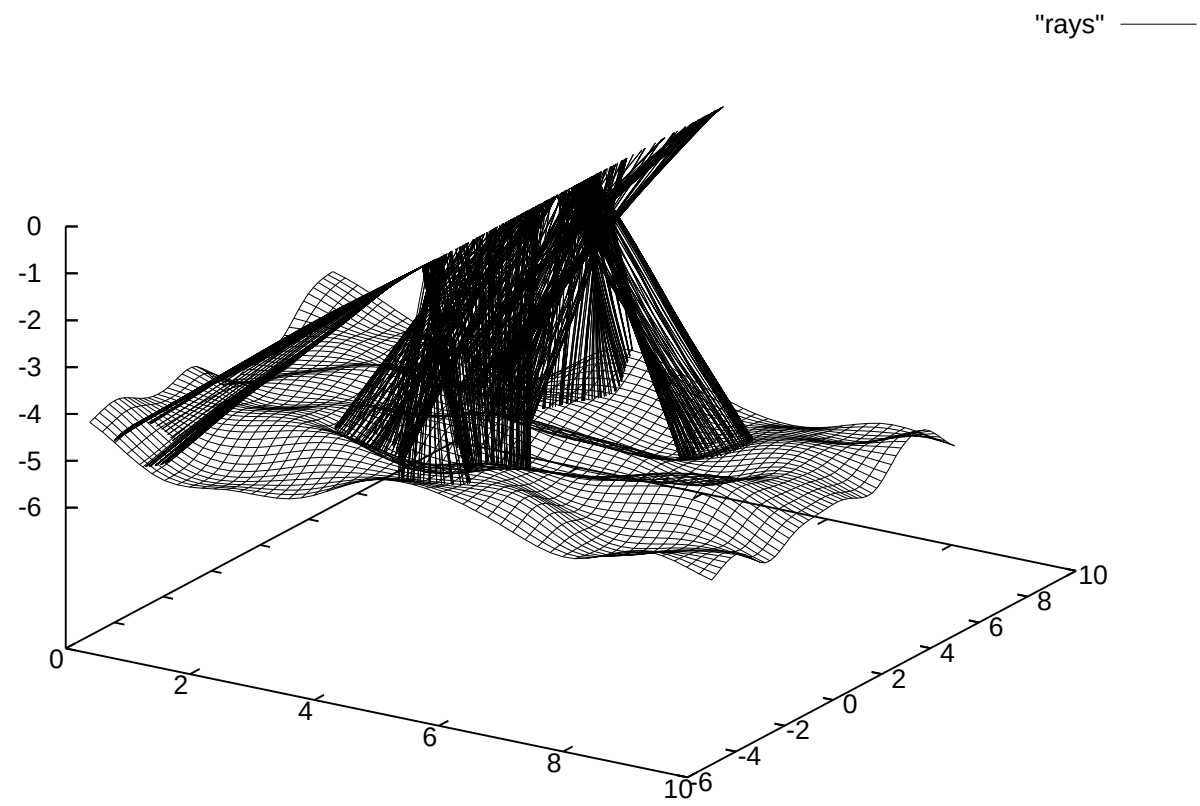
$$|F(\gamma) - F(\gamma')| \leq L \|\gamma - \gamma'\|$$

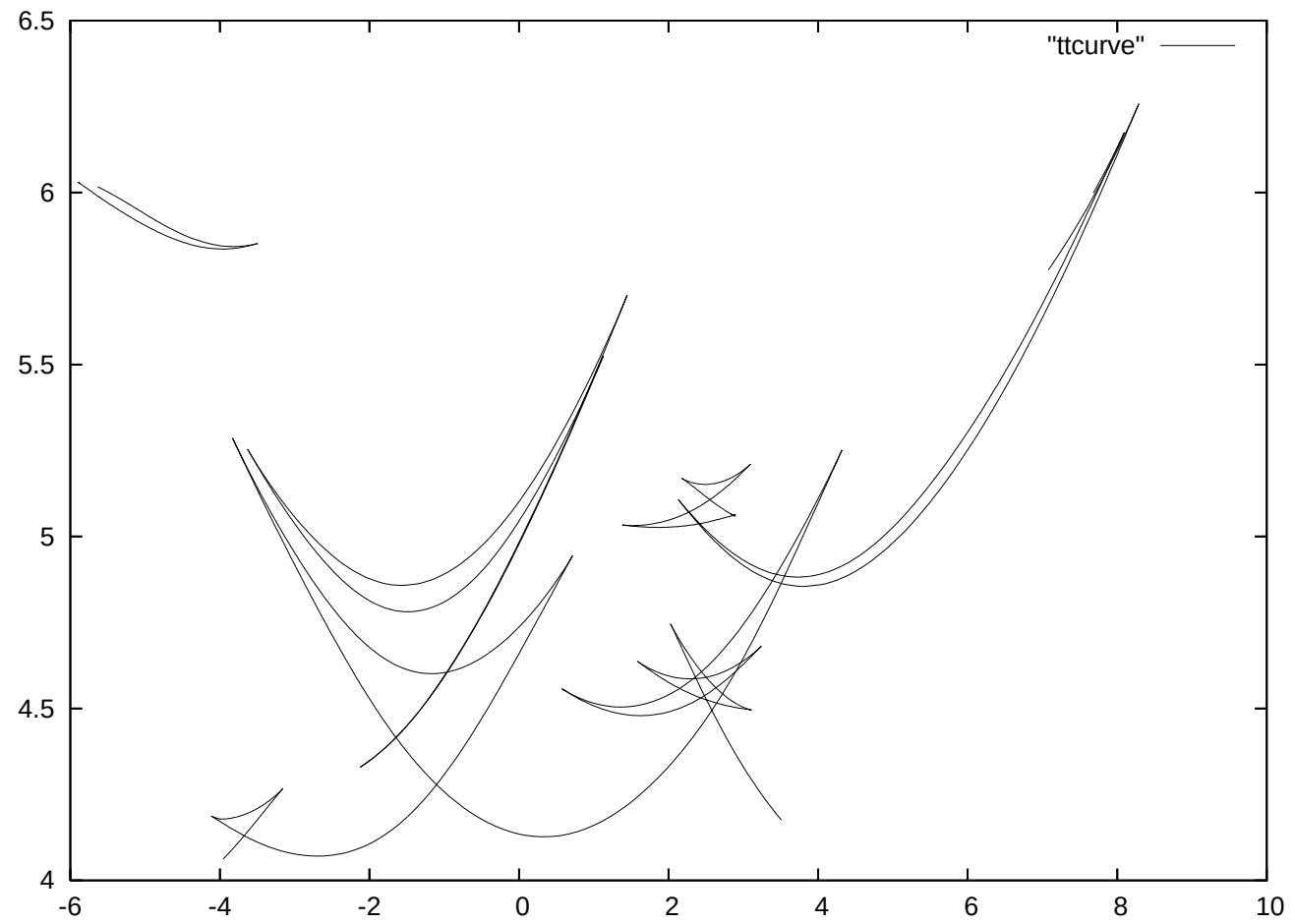
where  $\gamma = (\gamma_1, \gamma_2)$

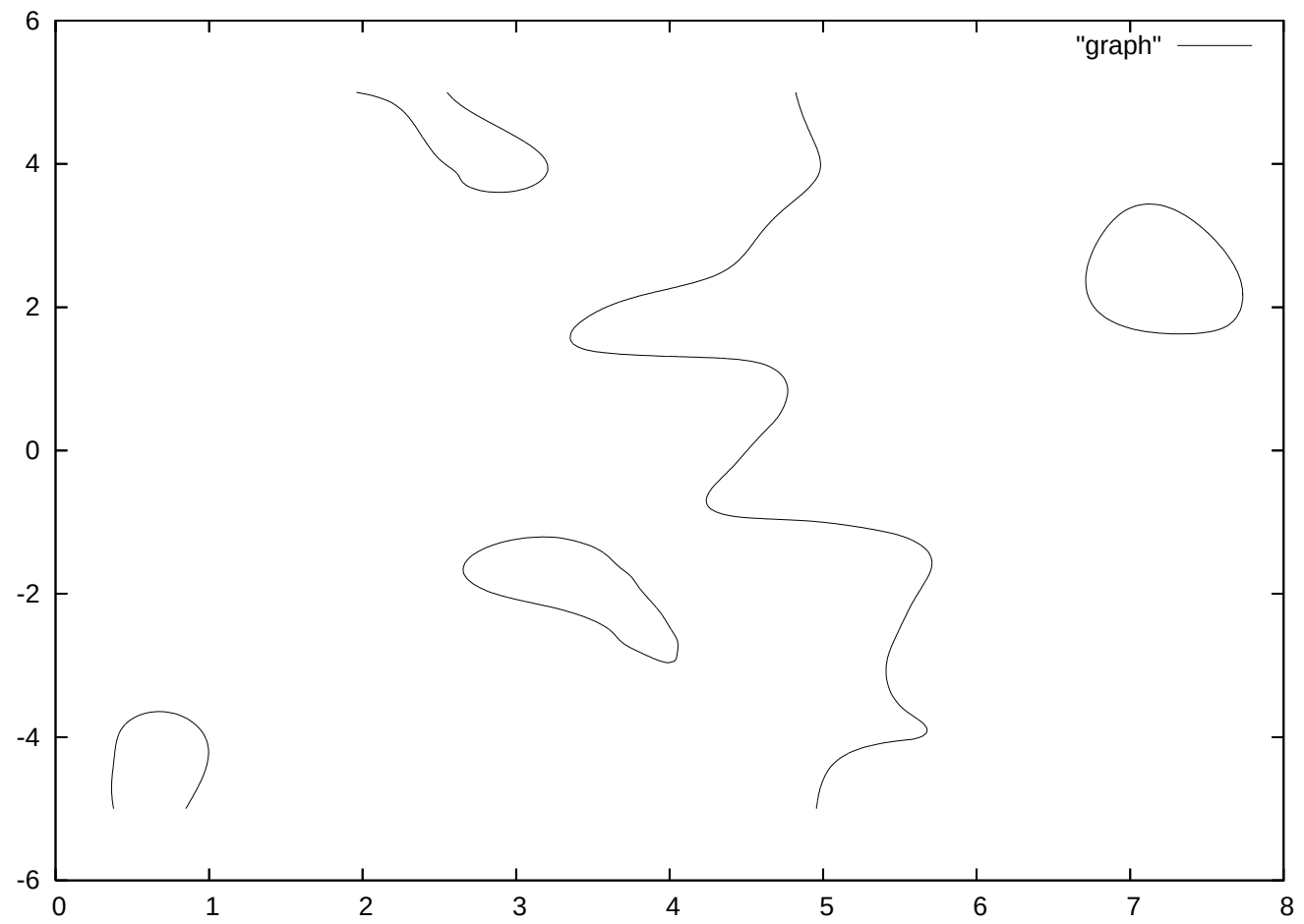
Recursive triangulation of  $\gamma$  plane, rejecting triangles for which Lipschitz condition predicts there is no solution.

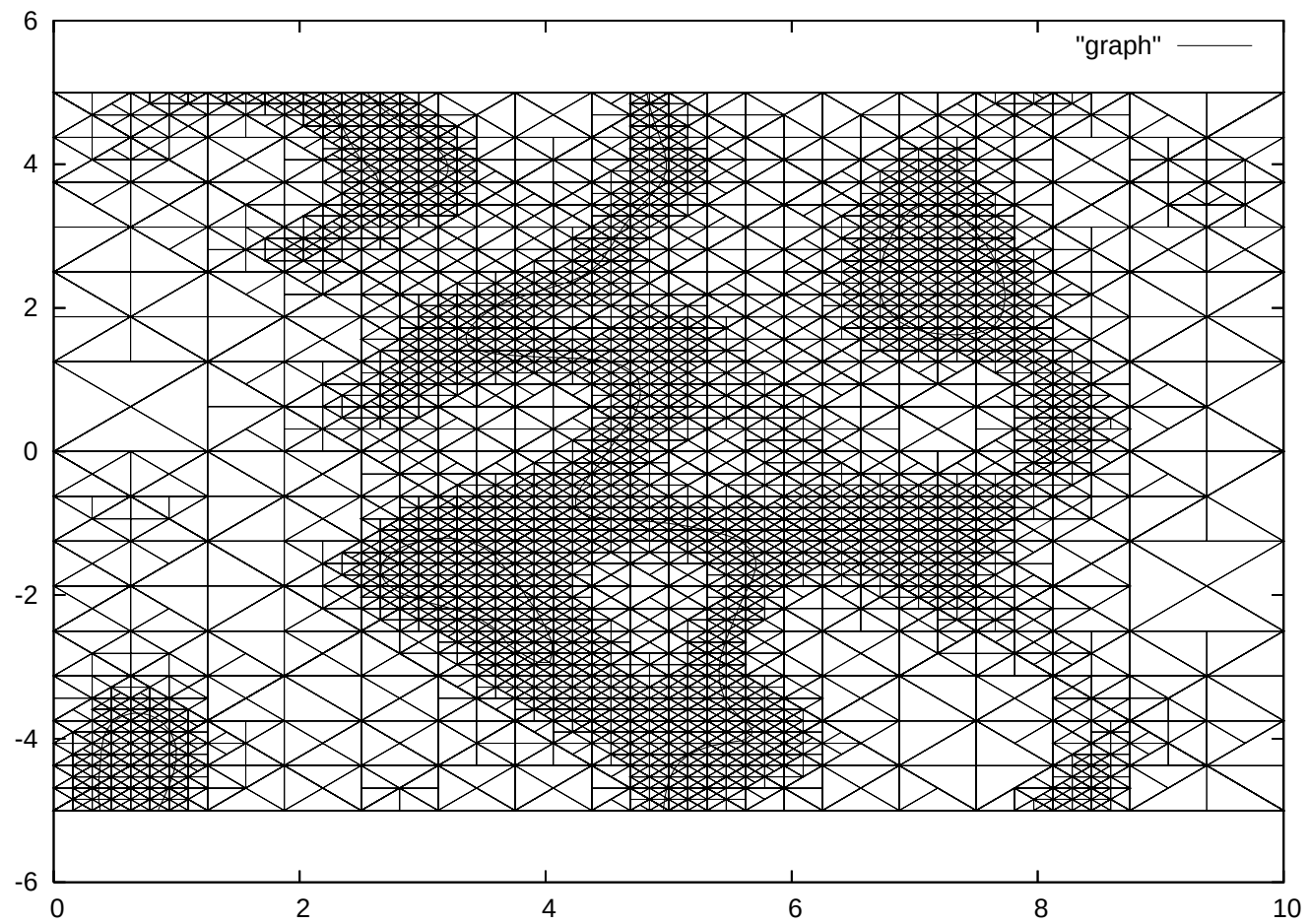
$$x^4 + x^2 y^2 - x (x^2 - y^2) + 0.0001 = 0$$











"rays" ———

