C^1 Spline Implicitization of Planar Curves

M. Shalaby*, B. Jüttler**, J. Schicho*

* Research Institute for Symbolic Computation
** Institute of Analysis, Dept. of Applied Geometry

Johannes Kepler University, Linz, Austria

Abstract

We present a new method for constructing a low degree C^1 implicit spline representation of a given parametric planar curve. To ensure the low degree condition, quadratic B-splines are used to approximate the given curve via orthogonal projection in Sobolev spaces. Adaptive knot removal, which is based on spline wavelets, is used to reduce the number of segments. The B-spline segments are implicitized. After multiplying the implicit B–spline segments by suitable polynomial factors the resulting bivariate functions are joined along suitable transversal lines. This yields to a globally C^1 bivariate function.

Keywords: implicitization, B-spline, approximation, knot removal

1 Introduction

Planar curves in Computer Aided Geometric Design can be defined in two different ways. In most applications, they are described by a *parametric representation*, $x = x(t)/w(t)$ and $y = y(t)/w(t)$ where $x(t), y(t)$, and $w(t)$ are often polynomials, or piecewise polynomials. Alternatively, the *implicit form* $f(x, y) = 0$ can be used. Both the parametric and implicit representation have its advantages. The availability of both often results in simpler and more efficient computations. For example, if both representations are available, the intersection of two curves can be found by solving a one–dimensional root finding problem.

Any rational parametric curve has an implicit representation, while the converse is not true. The process of converting the parametric equation into implicit form is called *implicitization.* A number of established methods for *exact* implicitization exists: resultants [3], Gröbner bases [1], and moving curves and surfaces [12]. However, *exact* implicitization has not found widespread use in CAGD. This is – among other reasons – due to the following facts:

- Exact implicitization often produces large data volumes, as the resulting implicit polynomials may have a huge number of coefficients.

- The exact implicitization process is relatively complicated, especially, in the case of high polynomial degrees. For instance, most resultant–based methods need the symbolic evaluation of large determinants.

- Even for regular parametric curves, the exact implicitization may have unwanted branches or self–intersections in the region of interest.

For these reasons, *approximate implicitization* has been proposed. Several methods are available: Montaudouin and Tiller [10] use power series to obtain local explicit approximation (about a regular point) to polynomial parametric curves and surfaces. Chuang and Hoffmann [2] extend this method using what they called "implicit approximation". Dokken [5] proposes a new way to approximate the parametric curve or surface globally; the approximation is valid within the whole domain of the curve segment or surface patch. Sederberg et al. [13] use monoid curves and surfaces to find an approximate implicit equation and approximate inversion map of a planar rational parametric curve or a rational parametric surface.

As a well–known fact, the parametric and implicit representations of a planar curve have the same polynomial degree. However, the number of the coefficients in the parametric case is $2(n+1)$ while it is $(n+1)(n+2)/2$ in the implicit case. In the implicit case, high polynomial degree will lead to expensive computations. This is even more dramatic for surfaces. We restrict ourselves to *low degree* implicitization.

In [7], we used quadratic B–splines for constructing a low degree spline implicit representation of a given parametric planar curve of degree n. A *spline implicitization* is a partition of the plane into polygonal segments, and a bivariate polynomial for each segment, such that the collection of the zero contours approximately describes the given curve. On the boundaries, these polynomial pieces are joined to form a globally C^m spline function, for a suitable choice of m. In [7], we restricted ourselves to continuous functions, i.e. $m = 0$.

Clearly, differentiability (C^1) is needed for many applications. For example, in foot point generation, one has to compute a point such that $[(\mathbf{p} - \mathbf{X}), \nabla f(\mathbf{X})] = 0$. As the computation of the gradient is needed, the curve should be C^1. Another example is the computation of distance bounds between two planar curves [8].

The main goal of this paper is to find a low degree C^1 spline implicit representation of a given parametric planar curve of degree n. To ensure the low degree condition, quadratic B-splines are used to approximate the given parametric curve (section 2). Adaptive knot removal, which is based on spline wavelets, is used to reduce the number of segments (section 3). The resulting quadratic B-spline segments are implicitized (section 4). Finally, by multiplying with suitable polynomial factors, these implicitized segments are joined together with C^1 continuity (sections 5, 6).

2 Quadratic B–spline Approximation

Following the idea proposed in [11], we generate a quadratic B–spline approximation via orthogonal projection in Sobolev spaces. The quadratic B–splines B_j on $[0,1]$ with uniform knots (stepsize $h = 2^{-i}$) and 3-fold boundary knots form an orthonormal sequence in a suitably weighted Sobolev space. In the interior of the segment, the inner product is defined by

$$\langle f, g \rangle = \frac{1}{h}(f,g) + \frac{1}{4}h(f',g') + \frac{1}{30}h^3(f'',g'') \tag{1}$$

where $(.,.)$ is the usual L^2 inner product. In order to achieve orthogonality at the boundary, additional terms have to be used. These weights and weight matrices (which are used near the boundary), have been specified in [11].

The B-spline approximation $\mathbf{g}^*$ of a given curve $\mathbf{g} = (g_1(t), g_2(t))$, with respect to the norm which is induced by the inner product (1), can then be written as:

$$\mathbf{g}^* = \sum_j \mathbf{d}_j \; B_j(t) \qquad \text{with} \qquad \mathbf{d}_j = \begin{pmatrix} \langle g_1, B_j \rangle \\ \langle g_2, B_j \rangle \end{pmatrix}$$

The control points $\mathbf{d}_j$ of the approximating B–spline curve can be generated by simple and efficient computations, as only (possibly numerical) integrations are needed. Also, no assumption about the given parametric representation have to be made, except that it should be at least in the underlying Sobolev space $H^{2,2}$. By using sufficiently many segments, an arbitrarily accurate approximation can be generated; the approximation order is 3.

Example

Consider a polynomial parametric curve $\mathbf{g}$ of degree 15 which is shown in Fig. 1. First, we approximate $\mathbf{g}$ using quadratic B–splines. Fig. 1 shows the error between $\mathbf{g}$ (black) and the quadratic B–spline approximation $\mathbf{g}^*$ (gray) for stepsize $h = 1/128$. Note that the error had to be exaggerated by a factor $\delta = 50000$ to make it visible.

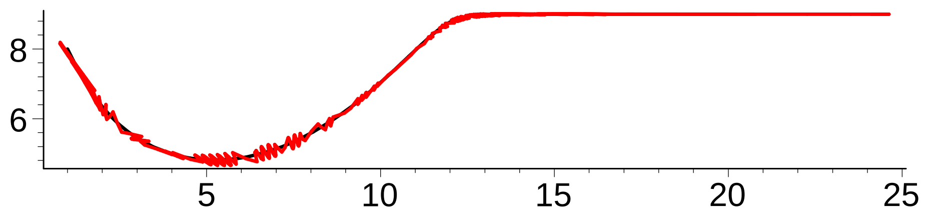


Figure 1 The original curve (black) and the error introduced by approximating it with a quadratic B–spline curve (gray). The error has been exaggerated.

3 Data Reduction via Spline Wavelets

After computing the initial B–spline approximation, we apply a knot removal (or data reduction) procedure, in order to reduce the number of segments. Such a reduction means that we approximate the given B–spline in a space S by a B-spline in linear subspace of S.

Methods for knot removal have been discussed by several authors, see e.g. [6, 9] and the references cited therein. In [6], the authors propose an optimal technique, by treating the knot removal procedure as a reverse approximative knot insertion process. It is based on a so–called "ranking list", which is used to compare the evaluate the error introduced by removing a specific knot.

In [7], a special method for our special situation has been proposed, which is based on the use of spline wavelets [4]. The method is not optimal, but it is cheaper than all other methods since no sorting or ranking lists are required. For the convenience of the reader, we give an outline of the method.

1. First, the wavelet transform of the given B-spline curve is computed.
2. Then, by setting all wavelets coefficients vectors with norm less than the threshold to be zero, we can remove blocks of wavelets with zero coefficients vector. For each block, one of the two common knots can be removed from the knot sequence. The length of these blocks varies between 2 and 5 wavelets, depending on the location of the removed knot in the knot sequence (that is, if the knot is an inner knot or close to the boundary).
3. Finally, the B–spline final representation $\mathbf{g}^{**}$ of the given B–spline curve $\mathbf{g}^{*}$ is computed over the reduced knot sequence K_{final}.

The error can be bounded simply by applying the wavelet synthesis to the set of removed wavelets. Due to the convex hull property of the B–splines, the error is bounded by the maximum absolute value of the resulting Bézier control points.

Example Continued

We apply the procedure to the quadratic B–spline curve $\mathbf{g}^{*}$. The number of knots is reduced from 133 to 13, where the threshold is equal to 10^{-3}. Fig. 2 shows the error between the original curve $\mathbf{g}$ (black) and the final B–spline representation curve $\mathbf{g}^{**}$ (gray) over K_{final}. The knots are plotted as circles. The knots at the boundary have multiplicity 3. The error is exaggerated by a factor $\delta = 5$ to make it visible.

4 Segment-wise Implicitization

After the data reduction process, we have a quadratic B–spline approximation $\mathbf{g}^{**}$ defined over the reduced non–uniform knot sequence K_{final}. In order to implicitize this curve,

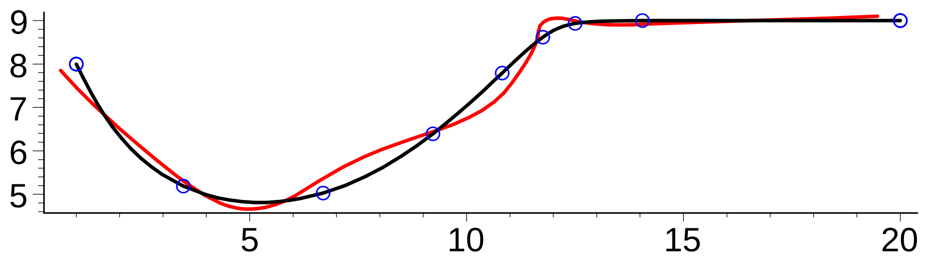


Figure 2 The original curve (black) and the B–spline approximation curve after the data reduction (gray). The error is exaggerated (amplified) by a factor $\delta = 5$.

we split the B–spline representation of this curve into the Bézier segments. Then, each quadratic Bézier segment is implicitized.

The conversion from B–spline representation of the curve to Bézier representation can easily be achieved via knot insertion. By increasing the knot multiplicity at each knot to be equal to the degree of the curve (in our case to be 2), the B–spline representation is converted to Bézier representation.

Each quadratic parametric Bézier segment has three control points. Let (p_0, q_0), (p_1, q_1) and (p_2, q_2) be the control points of one of these segments. Then the implicit form of this segment can be shown to be equal to

$$G(x,y) \;=\; \det \begin{pmatrix} Q_0(y)P_2(x) - P_0(x)Q_2(y) & Q_0(y)P_1(x) - P_0(x)Q_1(y) \\ Q_1(y)P_2(x) - P_1(x)Q_2(y) & Q_0(y)P_2(x) - P_0(x)Q_2(y) \end{pmatrix}$$

where

$$P_i(x) = \binom{2}{i}(p_i - x), \qquad Q_i(y) = \binom{2}{i}(q_i - y) \qquad \text{for} \qquad i = 0, 1, 2.$$

5 Joining two segments

In order to generate a C^1 continuous function, we modify the bivariate polynomials, which have been produced by the implicitization process, by multiplying them with suitable quadratic polynomial factors.

We shall use the following abbreviation:

Definition 1. *For a given polynomial $f(x,y)$, let $\mathcal{Z}(f)$ be the zero set $\{(x,y) \mid f(x,y) = 0\}$.*

In this section, we consider two neighboring Bézier segments of $\mathbf{g}^{**}$, with implicit representations $G_i(x,y)$, $i = 1,2$. These segments are parabolas which have a common tangent $T_{\mathbf{g}}$ at their common point $\mathbf{p}$ (see Fig. 3). Clearly, this point is non–singular. We choose the transversal line L as an arbitrary line passing through $\mathbf{p}$, which is different from $T_{\mathbf{g}}$. Let $\mathbf{q}$ and $\mathbf{r}$ be the second points of intersection of $\mathcal{Z}(G_1)$ and $\mathcal{Z}(G_2)$, respectively, with the chosen transversal line L. Let $\mathbf{s}$ be a point on L, distinct from $\mathbf{p}$, $\mathbf{q}$ and $\mathbf{r}$. We multiply $G_1(x,y)$ and $G_2(x,y)$ by quadratic polynomial factors $f_{2,1}(x,y)$ and $f_{1,2}(x,y)$ respectively such that:

(i) Neither $f_{2,1}(x,y)$ nor $f_{1,2}(x,y))$ contains the factor $\hat{L} = \hat{L}(x,y)$, where $\hat{L}$ is the linear polynomial that vanishes on the line L.

(ii) $\mathcal{Z}(f_{2,1})$ and $\mathcal{Z}(G_2)$ have a common tangent at $\mathbf{r}$.

(iii) $\mathcal{Z}(f_{1,2})$ and $\mathcal{Z}(G_1)$ have a common tangent at $\mathbf{q}$.

(iv) $\mathcal{Z}(f_{1,2})$ and $\mathcal{Z}(f_{2,1})$ have a common tangent at $\mathbf{s}$.

We consider the bivariate polynomials

$$F_1(x,y) = G_1(x,y)f_{2,1}(x,y) \qquad \text{and} \qquad F_2(x,y) = G_2(x,y)f_{1,2}(x,y).$$

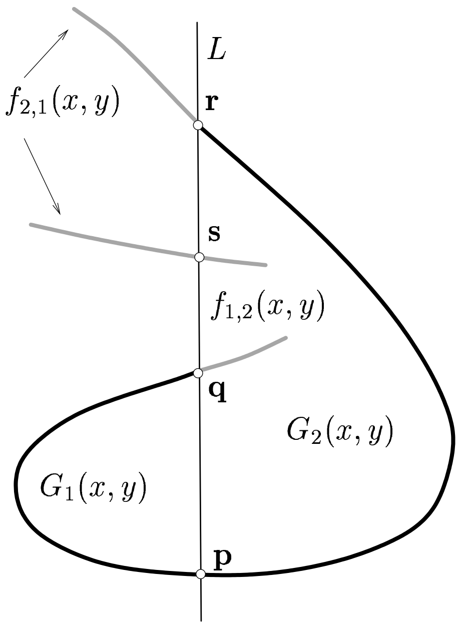


Figure 3 Multiplying by quadratic polynomial factors.

Theorem 1. *If the conditions $(i)-(iv)$ are satisfied, then, after multiplying F_2 by a suitable constant, the two bivariate polynomials $F_1(x,y)$ and $F_2(x,y)$ are C^1 along the transversal line L.*

Proof. Let

$$G_1(x,y) = \sum_{\substack{i+j\le 2\\ i,j\ge 0}} a_{i,j}\, x^i\, y^j, \qquad G_2(x,y) = \sum_{\substack{i+j\le 2\\ i,j\ge 0}} b_{i,j}\, x^i\, y^j,$$

$$f_{2,1}(x,y) = \sum_{\substack{i+j\le 2\\ i,j\ge 0}} c_{i,j}\, x^i\, y^j, \qquad f_{1,2}(x,y) = \sum_{\substack{i+j\le 2\\ i,j\ge 0}} d_{i,j}\, x^i\, y^j.$$

By normalization and simple change of coordinates, we may achieve that $\mathbf{p} = (0,0)$, $\nabla G_1(0,0) = \nabla G_2(0,0)$, and L is the y–axis. Let the coordinates of the points $\mathbf{q}$, $\mathbf{r}$, $\mathbf{s}$ be $(0,k_1)$, $(0,k_2)$ and $(0,k_3)$, respectively, with certain constants k_1, k_2 and k_3.

The zero contours of G_1 and G_2 pass through $\mathbf{p}$ and they have the same tangent at this point. Thus, $a_{0,0} = b_{0,0} = 0$, $a_{1,0} = b_{1,0}$ and $a_{0,1} = b_{0,1}$.

The conditions $(ii) - (iv)$ give 9 linear equations. By solving this system of equation, it is easy to show that after multiplying F_2 by $(k_2 c_{0,2})/(k_1 d_{0,2})$, the coefficients $A_{i,j}, B_{i,j}$ of $F_1(x,y)$ and $F_2(x,y)$ respectively are equal whenever $i \leq 1$. Hence, $F_1(x,y)$ and $F_2(x,y)$ meet with C^1 along L.

The cases $k_1 = 0$ and $k_2 = 0$ are excluded since $\mathbf{p}$ is nonsingular. The case $c_{0,2} = 0$ ($d_{0,2} = 0$) is the case where $f_{2,1}$ (resp. $f_{1,2}$) contains $\hat{L}$; and this is also excluded from the assumption (condition (i)). $\square$

Remark 1.
1. A similar analysis shows that the theorem is also valid if $\mathbf{s} = \mathbf{q}$, $\mathbf{s} = \mathbf{r}$, or $\mathbf{s} = \mathbf{q} = \mathbf{r}$.

2. In practice we choose $\mathbf{s}$ far away from $\mathbf{p}$, in order to avoid singular points in the area of interest. Note that the above construction has one degree of freedom.

3. It should be noted that a C^1 joint along L can be achieved also (see Fig. 4) if:

   (a) $f_{2,1} = f_{1,2} = \hat{L}^2$ or

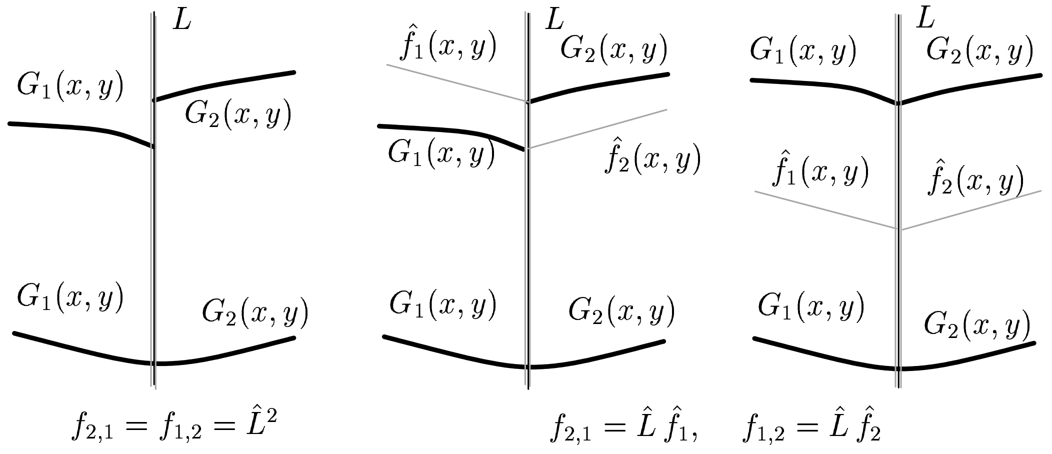
   (b) $f_{2,1} = \hat{L}\,\hat{f}_1$, $f_{1,2} = \hat{L}\,\hat{f}_2$ and $\hat{f}_1\,G_1$, $\hat{f}_2\,G_2$ are joined with continuously along L, where $\hat{f}_1$, $\hat{f}_2$ are two linear factors.

   These two cases are not interesting for applications, due to the presence of singularities.

4. It is clear that the transversal line passing through $\mathbf{p}$ should be chosen differently from the tangent to the curve at $\mathbf{p}$. In the sequel of this paper, we assume that this assumption is always satisfied.

6 Joining several segments

Now we use the technique from the previous section in order to join more than two segments. As an example, Figure 5 shows three neighboring segments which are described by bivariate polynomials G_{i-1}, G_i and G_{i+1}. In order to join G_{i-1} and G_i along L_{i-1}, we multiply G_{i-1} by $f_{2,i-1}$ and G_i by $f_{1,i}$ where $f_{2,i-1}$, $f_{1,i}$ are quadratic polynomial factors satisfying the conditions $(i) - (iv)$. Similarly, by multiplying G_i, G_{i+1} with $f_{2,i}$, $f_{1,i+1}$, respectively, we join these segments along L_i. In general, the polynomials $f_{1,i}$ and $f_{2,i}$ are different. Thus, we split the i-th patch again into two sub-patches by a line l_i and join the two different factors with C^1 continuity along it.



L
$G_1(x,y)$
$G_2(x,y)$
$\hat{f}_1(x,y)$
$\hat{f}_2(x,y)$
$f_{2,1} = f_{1,2} = \hat{L}^2$
$f_{2,1} = \hat{L}\,\hat{f}_1, \quad f_{1,2} = \hat{L}\,\hat{f}_2$

Figure 4 Multiplying by quadratic polynomial factors contain $\hat{L}$.

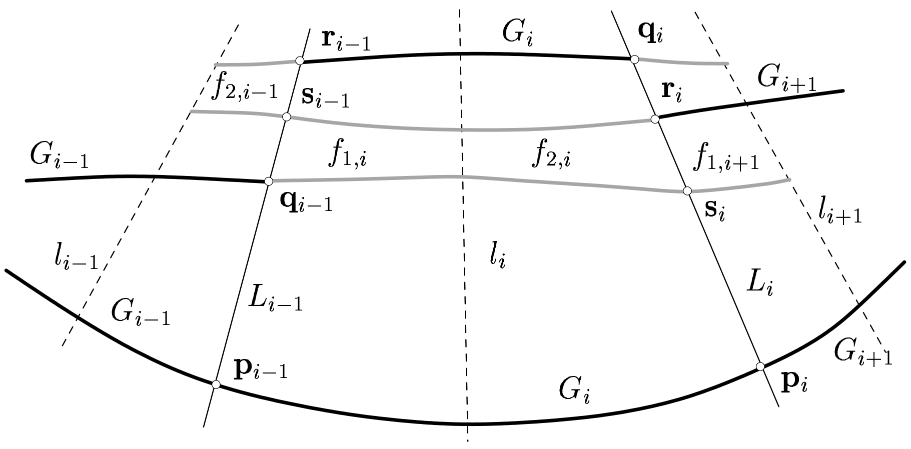


Figure 5 The original curve (black) and the quadratic polynomial factors (gray).

6.1 The algorithm

According to Theorem 1, we may achieve a C^1–joint along the transversal lines passing through the junction points $\mathbf{p}_i$, after multiplying with suitable quadratic polynomial factors. In order to obtain a *globally* C^1 spline function, we propose the following algorithm (see Fig. 5):

1. Divide the plane into patches by arbitrary lines L_i passing through the junction points $\mathbf{p}_i$. For instance, one may choose the normals to the curve.

2. Each internal patch (from the second to the $m-1$th patch, where m is the number of patches) is subdivided into two sub–patches by an arbitrary line l_i. For instance, one may choose the normal through the midpoint.

3. On the first boundary patch, we multiply by a quadratic polynomial factor $f_{2,1}$ such that $\mathcal{Z}(f_{2,1})$ is tangent continuous with the next parabola $\mathcal{Z}(G_2)$ at $\mathbf{r}_1$.

4. The next patch is modified by applying a piecewise quadratic multiplier, defined as a quadratic polynomials $f_{1,i}$ and $f_{2,i}$ on each of the two sub-patches. The multiplier has to satisfy the following conditions:

   - It must be C^1 on the whole patch.
   - $\mathcal{Z}(f_{1,i})$ is tangent continuous with the previous parabola $\mathcal{Z}(G_{i-1})$ at $\mathbf{q}_{i-1}$.
   - $\mathcal{Z}(f_{1,i})$ is tangent continuous with $\mathcal{Z}(f_{2,i-1})$ at $\mathbf{s}_{i-1}$.
   - $\mathcal{Z}(f_{2,i})$ is tangent continuous with the next parabola $\mathcal{Z}(G_{i+1})$ at $\mathbf{r}_i$.

   The first condition is fulfilled by a 7–dimensional linear space of spline functions. The other three conditions give two homogeneous linear conditions each[1]. This results in a homogeneous system of equations, which provides at least one nontrivial solution.

5. The previous step is repeated until one arrives at the last segment.

6. Finally, a similar construction as in Step 3 is used for the last segment.

Remark 2. 1. Theoretically, it could happen that the nontrivial solution computed in step 4 vanishes with multiplicity two along the transversal line. In this case, we cannot form a global C^1 function, because the multiplying constant would be zero. The next Lemma analyzes this degenerate case in more detail, showing that it can be excluded in practice.

2. The above construction provides two degrees of freedom. When joining the first two patches, one may choose the location of the point $\mathbf{s}_1$ and the slope of tangent T_1 (the tangent of $f_{2,1}$ at $\mathbf{s}_1$). In general, the multipliers which are applied to the other patches are then determined up to scalar constants.

Now we analyze the construction which is used in Step 4.

Lemma 1. *Consider the situation of step 4, see figure 6. In addition, we assume that the following assumptions are satisfied.*

- *The three points $\mathbf{p_{i-1}}$, $\mathbf{q_{i-1}}$ and $\mathbf{s_{i-1}}$ are distinct.*
- *The three points $\mathbf{p_i}$, $\mathbf{r_i}$ and $\mathbf{s_i}$ are distinct.*
- *L_{i-1} does not pass through $\mathbf{r_i}$.*
- *L_i does not pass through $\mathbf{q_{i-1}}$ or $\mathbf{s_{i-1}}$.*

[1]The zero contours of two bivariate functions h_1 and h_2 are tangent continuous at a point $\mathbf{p} \in \mathbb{R}^2$, if $h_1(\mathbf{p}) = h_2(\mathbf{p})$ and $\det(\nabla h_1(\mathbf{p}), \nabla h_2(\mathbf{p})) = 0$. This leads to two homogeneous linear equations.

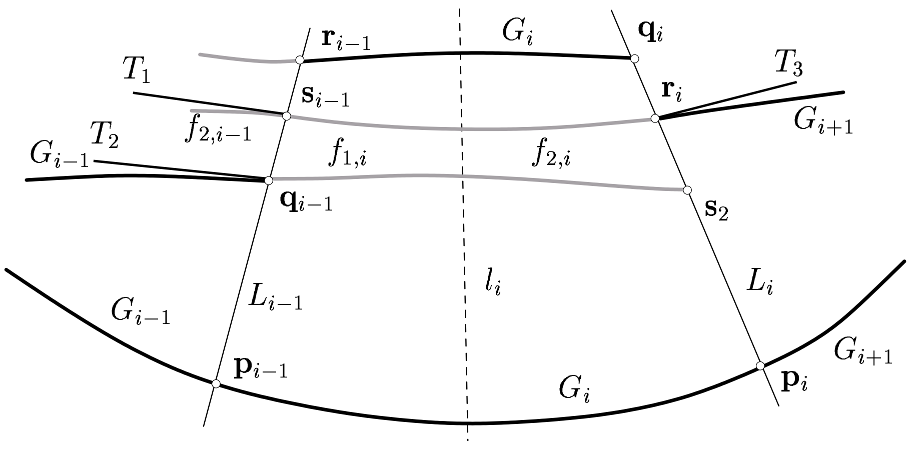


Figure 6 Step 4 of the construction.

- *l_i does not pass through $\mathbf{r_i}$, $\mathbf{q_{i-1}}$ or $\mathbf{s_{i-1}}$.*
- *l_i does not pass through the intersection points of L_{i-1} and T_3, L_i and T_1, or L_i and T_2, where T_1 is the tangent of $f_{1,i}$ at $\mathbf{s_{i-1}}$, T_2 is the tangent of $f_{1,i}$ at $\mathbf{q_{i-1}}$, and T_3 is the tangent of $f_{2,i}$ at $\mathbf{r_i}$.*

Then the nontrivial solution obtained in Step 4 does not vanish along the transversal lines L_{i-1} or L_i.

Proof. Let

$$\begin{aligned} f_{1,i} &= c_{0,0} + c_{1,0}\,x + c_{2,0}\,x^2 + c_{1,1}\,xy + c_{0,1}\,y + c_{0,2}\,y^2 \quad \text{and} \\ f_{2,i} &= d_{0,0} + d_{1,0}\,x + d_{2,0}\,x^2 + d_{1,1}\,xy + d_{0,1}\,y + d_{0,2}\,y^2. \end{aligned}$$

By normalization and suitable choice of coordinates, we may achieve that $\mathbf{p_{i-1}} = (0,0)$ and L_{i-1} is the y–axis. Let l_i be defined by

$$y - y_0 - m_0(x - x_0) = 0$$

If $f_{1,i}$ and $f_{2,i}$ are C^1 along l_i, then

$$\begin{aligned} f_{1,i} &= c_{0,0} + c_{1,0}\,x + c_{2,0}\,x^2 + c_{1,1}\,xy + c_{0,1}\,y + c_{0,2}\,y^2 \quad \text{and} \\ f_{2,i} &= f_{1,2} + (d_{0,2} - c_{0,2})(y - y_0 - m_0(x - x_0))^2. \end{aligned}$$

Let m_1, m_2 and m_3 be the slopes of T_1, T_2 and T_3 respectively. Moreover let $(0, y_1)$, $(0, y_2)$ and (x_3, y_3) be the coordinates of $\mathbf{q_{i-1}}$, $\mathbf{s_{i-1}}$ and $\mathbf{r_i}$, respectively. $\mathcal{Z}(f_{1,i})$ passes through $\mathbf{q_{i-1}}$, $\mathbf{s_{i-1}}$ and has tangents T_2, T_1 at these points. Also, $\mathcal{Z}(f_{2,i})$ passes through $\mathbf{r_i}$ and tangent T_3 there. This gives 6 homogeneous linear conditions for 7 unknowns ($d_{0,2}$ and $c_{k,j}$, $k, j = 0..2, k + j \leq 2$).

Assume that $c_{0,0} = 1$, i.e., $f_{1,i}$ does not vanish at L_{i-1}. Then we get a system of 6 linear equations for 6 unknowns. By factoring the determinant of the coefficient matrix we get

$$2\underbrace{y_1\, y_2\,(y_1 - y_2)}_{1,2,3}\ \underbrace{x_3}_{4}\ \underbrace{(y_3 - y_0 - m_0(x_3 - x_0))}_{5}\ \underbrace{(y_3 - m_3x_3 - y_0 + m_0x_0)}_{6}$$

Clearly, there exist two quadratic factors $f_{1,i}$ and $f_{1,i}$ satisfying the above conditions if the determinant is not equal to zero.

The first three factors vanish if the three points $\mathbf{p_{i-1}}$, $\mathbf{q_{i-1}}$ and $\mathbf{s_{i-1}}$ are not identical. The fourth factor vanishes if L_{i-1} passes through $\mathbf{r_i}$, and the fifth factor vanishes if l_i passes through $\mathbf{r_i}$.

The lines L_1 and T_3 intersect at $(0, y_3 - m_3x_3)$. Consequently, the sixth factor vanishes if l_i passes through the intersection point of L_{i-1} and T_3.

Analogously, by applying the same technique to $f_{2,i}$ and L_i, it can be shown that the three points $\mathbf{p_i}$, $\mathbf{r_i}$ and $\mathbf{s_i}$ should be distinct, L_{i-1} should not pass through $\mathbf{q_{i-1}}$ or $\mathbf{s_{i-1}}$, l_i should not pass through $\mathbf{q_{i-1}}$ or $\mathbf{s_{i-1}}$, and l_i should not pass through intersection points of L_i and T_1, or L_2 and T_2. □

As the main problem of this construction, the coordinates of the point $\mathbf{s}_i$ depend on the coordinates of the previously generated point $\mathbf{s}_{i-1}$. After the first patch, we do not have any control on the coordinates of the points $\mathbf{s}_l$, $l = 2, \ldots, m-1$ where m is the number of the patches[2]. Hence, $\mathbf{s}_l$ may coincide with $\mathbf{p}_l$, $\mathbf{q}_l$ or $\mathbf{r}_l$, i.e the polynomial factors vanish on the transversal line. Moreover, the polynomial factors may intersect the original curve at the area of interest.

As an example, Fig. 7 shows the original curve segments (black) and the quadratic polynomial factors (gray), these factors intersecting the original curve at points A^* and B^*. Here, The intersection at the point A^* is not so important because this happened far away from our area of interest. The main problem is at the point B^*. In order to avoid this problem, we will localize our approach.

6.2 The modified algorithm

If there is an intersection of the quadratic polynomial factors and the original curve, or $\mathbf{s}_l$ coincides with $\mathbf{p}_l$, $\mathbf{q}_l$ or $\mathbf{r}_l$, then we introduce an unwanted singularity in the region of interest. There is degree of freedom to choose the point s_1 and the tangent T_1, but it may not be sufficient to guarantee that there is a choice avoiding singularities. This problem can be avoided by dividing each patch to 4 sub-patches instead of 2 sub-patches. Below, we give the sketch of the modified algorithm (see Fig. 8).

1. Divide the plane into patches by arbitrary lines L_i passing through the junction points $\mathbf{p}_i$. For instance, one may choose the normals to the curve.

[2]It is the same problem as for C^1 interpolation with quadratic splines. Once we fix the tangent direction for the first quadratic function we do not have freedom to choose any of the tangent directions afterwards

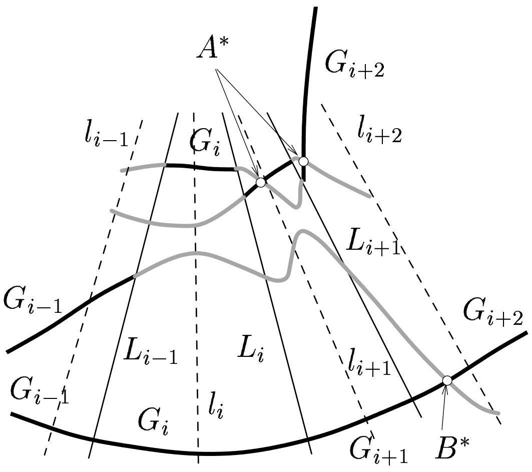


Figure 7 The quadratic factors intersect the original curve at A^*, B^*

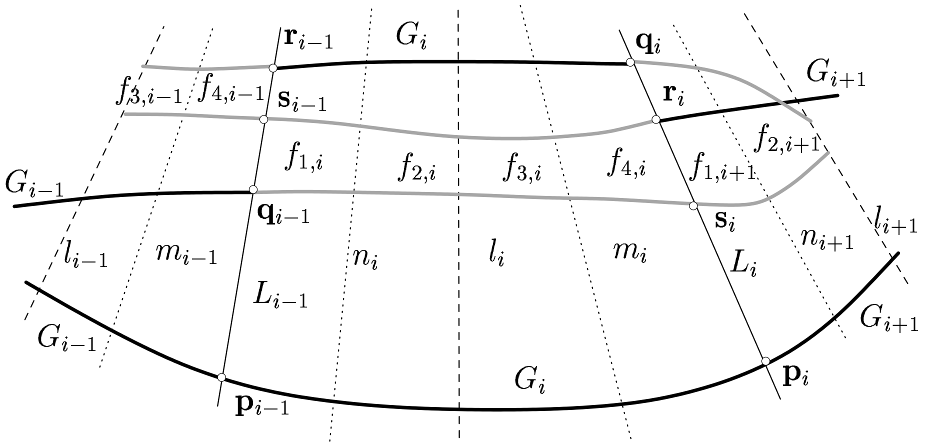


Figure 8 The original curve (black) and the modified quadratic factors (gray)

2. Each internal patch (from patch 2 to patch $m-1$, where m is the number of patches) is subdivided into 4 sub-patches by arbitrary lines l_i, n_i, m_i. For instance, one may choose the normal to the curve through three equally spaced points.

3. On the first boundary patch, we multiply by a factor $f_{4,1}$ of degree 2 such that $\mathcal{Z}(f_{4,1})$ is tangent continuous with the next parabola $\mathcal{Z}(G_2)$ at $\mathbf{r}_1$.

4. On the next patch, we multiply by a piecewise multiplier, defined as a quadratic polynomials $f_{1,i}$, $f_{2,i}$, $f_{3,i}$, and $f_{4,i}$ on each of the four sub-patches. It has to satisfy the following conditions:

   – It must be C^1 within the whole patch.
   – $\mathcal{Z}(f_{1,i})$ is tangent continuous with the previous parabola $\mathcal{Z}(G_{i-1})$ on $\mathbf{q}_{i-1}$.
   – $\mathcal{Z}(f_{1,i})$ is tangent continuous with $\mathcal{Z}(f_{4,i-1})$ at $\mathbf{s}_{i-1}$.

– $\mathcal{Z}(f_{4,i})$ is tangent continuous with the next parabola $\mathcal{Z}(G_{i+1})$ at $\mathbf{r}_i$.

The first condition is fulfilled by a 9-dimensional vector-space of splines. The other three conditions give two linear conditions each, so we gain two additional degree of freedom. After choosing these two degree of freedom (see below) there is a nontrivial solution.

5. The previous step is repeated until one arrives at the last segment.
6. Finally, a similar construction as in Step 3 is used for the last segment.

6.3 Using the additional degree of freedom

Using the above algorithm, for each internal patch, we gain two additional degrees of freedom. They can be used to control the curve locally. This can be done by choosing the intersection point $\mathbf{s}_i$ and the slope of tangent T_i at this point.

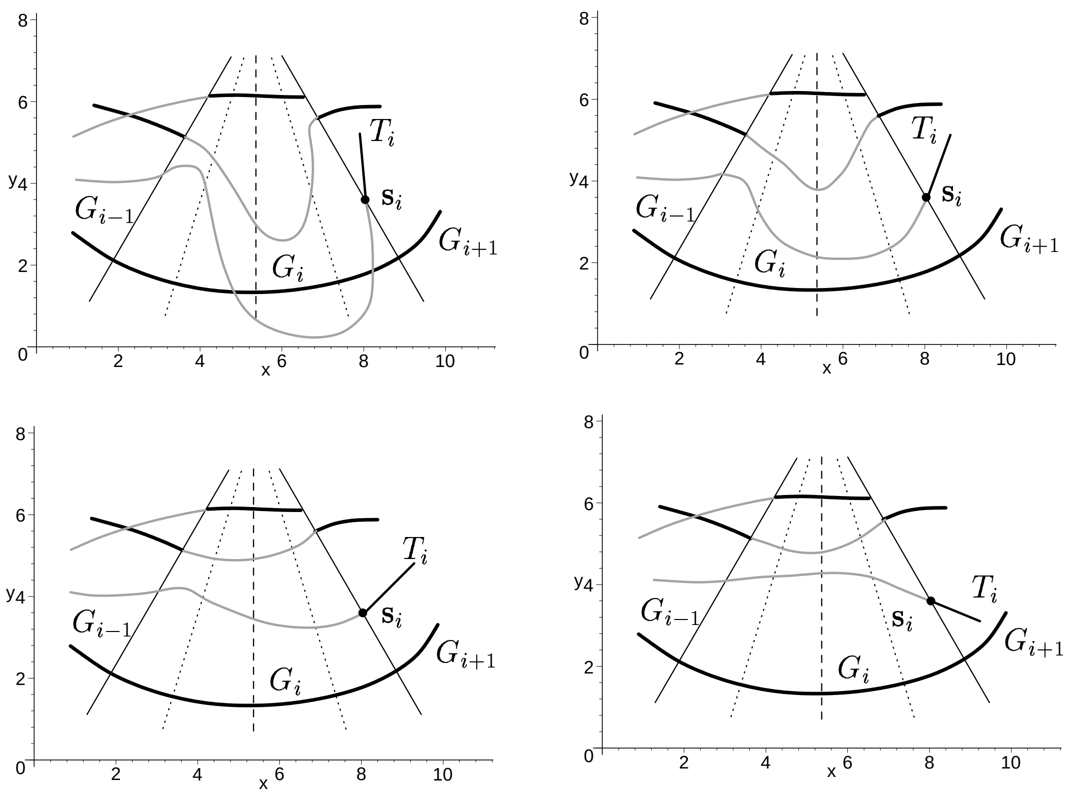


Figure 9 Using the slope of the tangent T_i to control the curve locally

Fig. 9 shows the original curve (black) and the polynomial factors (gray) for different choice of the tangent T_i at $\mathbf{s_i}$. If it here is an intersection between the polynomial factors and the original curve at any patch (for instance patch i) or $\mathbf{s}_i$ coincide with $\mathbf{p}_i$, $\mathbf{q}_i$ or $\mathbf{r}_i$, we modify the coordinate of $\mathbf{s}_i$ or the slope of T_i at this patch to avoid the intersection. Any such modification will act locally and affect only two patches (patch i and patch $i+1$).

Clearly, subdividing into four sub–patches is more expensive than subdividing into two sub–patches, and it leads to a higher data volume. In practice, one may use the following method. First, divide each patch into two sub–patches. Only if a singularity is introduced at patch i, then we discard the two sub-patches and subdivide into four sub–patches.

Fig. 10 shows the same example of Fig. 7. In the left figure, the polynomial factors intersect the original curve at points A^*, B^*. After using the localized method, we avoid the intersection between the quadratic polynomials factors and the original curve at the area of interest (right figure). We kept the intersection at points A^* because it is far away from the area of interest.

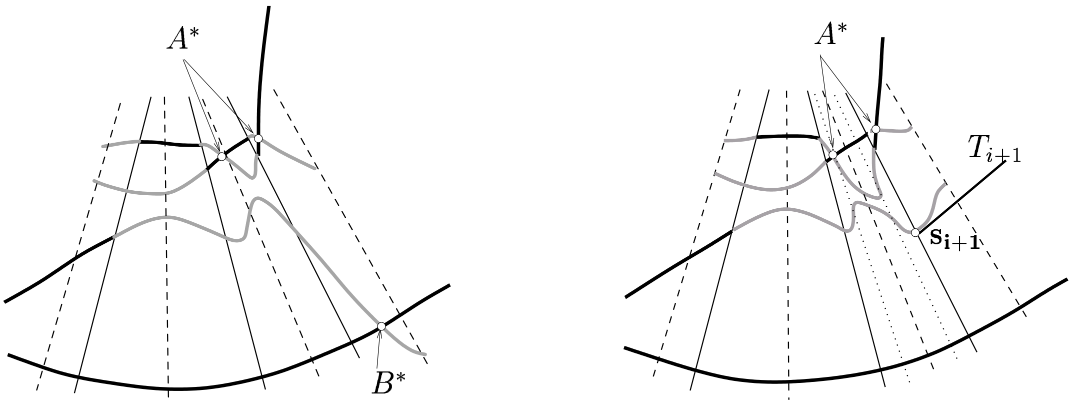


Figure 10 Using the two additional degree of freedom to avoid the intersection between the quadratic factors and the original curve.

Example (finished)

We start with the piecewise quadratic function whose zero contour $G(x,y) = 0$ is the quadratic B–spline curve in Fig. 2. We multiply by suitable quadratic polynomial factors. Fig. 11 shows the algebraic offsets[3] (thin lines) of $\mathcal{Z}(G)$ (thick line) and the transversal lines through the junction points. To make the picture clearer, we enlarge a part of the curve and draw some additional algebraic offsets, see Fig. 12. It can clearly be seen that the algebraic offsets are tangent continuous.

7 Conclusion

We have derived a method for constructing a low degree C^1 implicit spline representation of a given parametric planar curve. The construction consists of four steps: B–spline curve approximation, knot removal, segment implicitization and segment joining.

[3]The algebraic offsets are the curves $\mathcal{Z}(G-c)$, where $c \neq 0$ is a certain constant.

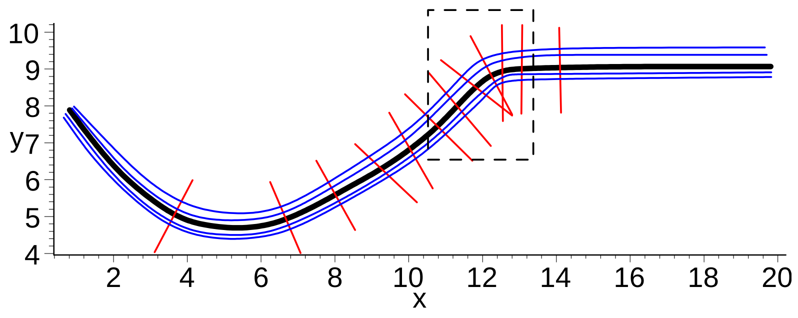


Figure 11 C^1 implicitized curve and its algebraic offsets.

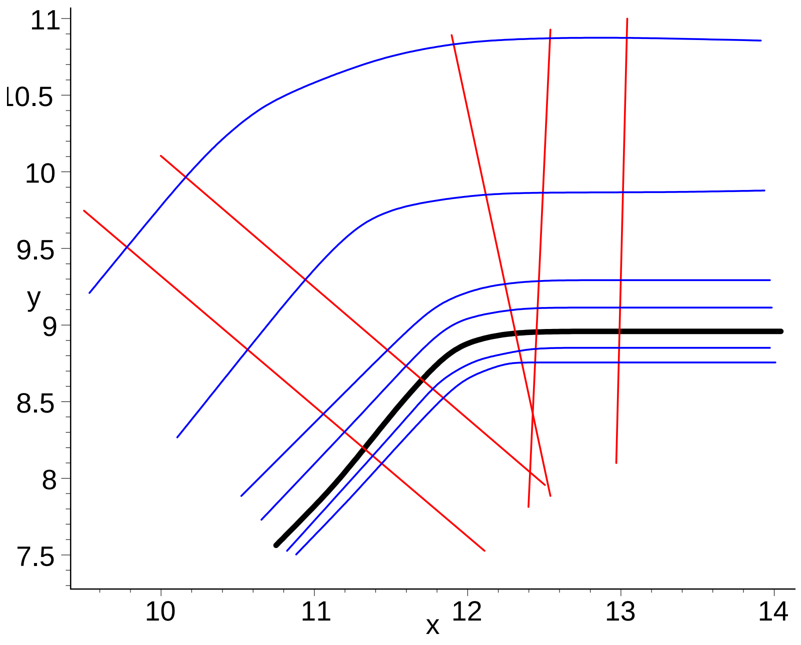


Figure 12 C^1 Implicitized curve and its algebraic offsets(enlarged)

Compared to the existing methods for implicitization, our method has the following advantages.

- The method is computationally simple. In particular, no evaluations (symbolic or numerical) of large determinants are needed.
- It produces a low degree implicit representation. For instance, the intersection of a line with the implicitized curve can be found by computing the roots of a quartic polynomial.
- The methods avoids unwanted branches or singularities, which otherwise could be present in the neighborhood of the given curve.
- The implicit function is globally C^1 continuous.

- The method can be applied to any parametric curve with coordinate functions in the Sobolev space $H^{2,2}$, not just to polynomial or piecewise polynomial representations.

As a matter of future research, we plan to find out an automatic method for adjust the tangents slope (for the modified algorithm 6.2) and generalize this method to surfaces.

Acknowledgments

This research has been supported by Austrian science fund (FWF) in the frame of the Special Research Program (SFB) F013 “Numerical and Symbolic Scientific Computing”, project 15. The second author has also been supported by the European Commission through its Fifth Framework Programme (Project GAIA II entitled “Intersection algorithms for geometry based IT-applications using approximate algebraic methods”, contract no. IST-2002-35512)

References

[1] B. Buchberger, *Application of Gröbner bases in non-linear computational geometry*, in: J. Rice, (ed.), Mathematical Aspects of Scientific Software, Springer, New York/Berlin, 1988, 59–87.

[2] J.H. Chuang and C. M. Hoffmann, *On local implicit approximation and its application*, ACM Trans. Graphics 8, 4 (1989), 298–324.

[3] D. Cox, J. Little, and D. O’Shea, *Ideals, varieties and Algorithms*, Springer-Verlag, New York, 1997.

[4] C.K. Chui and E. Quak, *Wavelets on a bounded interval*, in: D. Braess, L.L. Schumaker (eds.), Numerical Methods in Approximation Theory Vol. 9, Birkhäuser, Basel, 1992, 53–75.

[5] T. Dokken, *Approximate implicitization*, in: T. Lyche, L.L. Schumaker (eds.), Mathematical Method in CAGD, Vanderbilt University Press, Nashville London, 2001, 81–102.

[6] M. Eck, and J. Hadenfeld, *Knot removal for B-spline curves*, Computer Aided Geometric Design 12 (1995), 259–282.

[7] B. Jüttler, J. Schicho and M. Shalaby, *Spline Implicitization of Planar Curves*, submitted.

[8] B. Jüttler, *Bounding the Hausdorff Distance of Implicitly Defined and/or Parametric Curves*, in T. Lyche and L.L. Schumaker (eds.), Mathematical Methods in CAGD: Oslo 2000, Vanderbilt University Press, Nashville 2001, 223–232.

[9] T. Lyche, *Knot removal for spline curves and surfaces*, in E.W. Cheney, C. Chui, and L. Schumaker (eds), Approximation Theory VII, Academic Press, New York, 1992, 207–226.

[10] Y. de Montaudouin, W. Tiller, and H. Vold, *Application of power series in computational geometry*, Computer-Aided Design 18, 10, 1986, 93–108.

[11] U. Reif, *Orthogonality Relations for Cardinal B-Splines over Bounded Intervals*, in: W. Strasser, R. Klein and R. Rau (eds.), Geometric Modeling: Theory and Practice, Springer, (1998), 56-69.

[12] T.W. Sederberg and F. Chen, *Implicitization using moving curves and surfaces*, in: R. Cook, (ed.), Computer Graphics 29 (SIGGRAPH '95 Conference Proceedings), 301–308, Addison-Wesley, Reading, MA.

[13] T.W. Sederberg, J. Zheng, K. Klimaszewski and T. Dokken, *Approximate Implicitization Using Monoid Curves and Surfaces*, Graphical Models and Image Processing 61, 1999, 177–198.