Spline Implicitization of Planar Shapes and Applications

M. Shalaby
Research Institute for Symbolic Computation

Johannes Kepler University, Linz, Austria

July 21, 2003

Zusammenfassung

Wir entwickeln eine neue Methode zur Konstruktion algebraischer Spline-Kurven durch näherungsweise Implizitisierung. Unser Ziel ist die Erzeugung impliziter Spline-Darstellungen niedrigen Grades (in C^m für $m = 0, 1$) von parametrisch gegebenen ebenen Kurven. Um einen niedrigen Grad zu erhalten, werden quadratische B-Splines verwendet und die gegebene Kurve wird durch Orthogonalprojektionen in Sobolev- Räumen angenähert. Auf Spline-Wavelets basierende, adaptive Methoden zur Entfernung von Kontrollpunkten verringern die Anzahl der benötigten Segmente. Die resultierenden Spline-Segmente werden implizitisiert.

Um eine stetige bivariate Funktion (in C^0) zu erhalten, werden die implizitisierten Segmente entlang geeigneter Linien zusammengesetzt. So erhält man eine global stetige bivariate Funktion. Durch Analyse des asymptotischen Verhaltens für Schrittweite $h \to 0$ wird gezeigt, daß die Kurve mit beliebiger Genauigkeit implizitisiert werden kann.

Um eine stetig differenzierbare bivariate Funktion (in C^1) zu erhalten, multiplizieren wir die implizitisierten Segmente mit passenden polynomialen Faktoren. Die resultierenden Segmente werden wiederum entlang geeigneter Linien zusammengesetzt. So erhält man eine global stetig differenzierbare bivariate Funktion.

Abschließend erwähnen wir kurz einige Anwendungen der vorgestellten Methode. Sie gibt uns ein einfaches und mächtiges Werkzeug zur Implizitisierung ebener Kurven in die Hand.

Abstract

A new method for constructing algebraic spline curves by approximate implicitization is proposed. Our research aims at generating a low degree C^m implicit spline representation of a given parametric planar curve for $m = 0, 1$. 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 resulting spline segments are implicitized.

In order to generate a continuous bivariate function, the implicitized segments are joined along suitable transversal lines. This yields a globally continuous bivariate function. As shown by analyzing the asymptotic behavior of these transversal lines for step size $h \to 0$, the given curve can be implicitized with any desired accuracy.

In order to generate a C^1 bivariate function, we multiply the implicitized segments with suitable polynomial factors. The resulting bivariate functions are joined along suitable transversal lines. This yields a globally C^1 bivariate function.

We conclude the thesis by discussing some applications for the proposed method. Our method yields a simple and powerful tool for the spline implicitization of planar curves.

Introduction

A planar algebraic curve over a field K is a curve which can be defined implicitly as $F(x, y) = 0$, where $F(x, y)$ is a polynomial in x and y with coefficients in K. It can be seen as the zero contour[1] of the bivariate function $F(x, y)$.

A planar polynomial parametric curve is a curve which can be defined parametrically as $x = p(t)/w(t)$ and $y = q(t)/w(t)$ where $p(t), q(t)$, and $w(t)$ are polynomials, or piecewise polynomials.

Both implicitly and parametrically defined curves play an essential role in Computer Aided Geometric Design "CAGD". Parametric curves are better suited for certain applications than algebraic curves. For example, in display purposes, it is easier to generate a large numbers of points on such curve if the parametric form is available. Also, finite segments of curves are easily defined by imposing parameter bounds. Another advantage, parametric curves can be pieced together with various degrees of derivative continuity. In addition, certain parametric curve forms exhibit highly desirable properties such as the variation diminishing property and convex hull containment.

In other hand, it is well known that algebraic curves (curves defined by an implicit equation) are better suited for certain applications than are parametrically defined curves. For example, an implicit equation makes it simple to determine whether a given point lies on the curve, or if not, to determine which side of the curve it lies on. As a theoretical advantage, the set of all parametric curves are not closed under the offsetting operation[2], but the set of algebraic curves is, see [35, 39]. In other word, the offset curve of a parametric curve is not in general a polynomial parametric curve, while it is always an algebraic curve.

[1] The zero contour of a function $F(x, y) = 0$ is the zero set$\{(x, y) \mid F(x, y) = 0\}$

[2] An offset curve is a curve generated by sweeping along a fixed perpendicular distance from a given curve.

It is well known that for any polynomial parametric curve there exists an implicit equation which defines precisely the same curve [38]. In addition, the availability of both implicit and parametric representation 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.

For these reasons, it is clear that algebraic curves are important entities in CAGD environment. The process of converting the parametric equation into implicit form is called *implicitization*. A number of established methods for *exact implicitization* exists:

- resultants [6],
- Gröbner bases [2],
- moving curves and surfaces [40].

However, *exact implicitization* has not found widespread use in CAGD. This is – among other reasons – due to the following facts:

1. 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.
2. Exact implicitization often produces large data volumes, as the resulting bivariate polynomials may have a huge number of coefficients.
3. Even for regular parametric curves, the exact implicitization may have unwanted branches or self–intersections in the region of interest.

For a planar curve of degree n, it is well known that the number of the coefficients in the implicit case is $(n+1)(n+2)/2$. Hence, the 2^{nd} problem is some how is a high polynomial degree problem. In order to avoid the 1^{st} and 2^{nd} problems, one may hope to get a low degree implicit form of a given parametric curve of degree n. Clearly, the quadratic curves are singularity free curves. In order to avoid the above 3 problems, one may hope to find an implicit representation of degree 2 of a given parametric curve of degree n.

Unfortunately this can not be done by *exact implicitization*. This is due to the fact: the parametric and implicit forms of a planar curve have the same degree. In order to have a low degree implicit form for a parametric curve of degree n. One may use what we called *low degree spline implicitization.*

Spline implicitization is an *approximate implicitization* method. In [19, 42], we discusses the problem of constructing *spline implicitization* for planar curves: a partition of the plane into polygonal segments, and a bivariate polynomial for each segment, such that the collection of the zero countors approximately describes the given curve. The polynomial pieces form a global C^m spline function, for certain choice of m. We restrict ourselves to the cases $m = 0, 1$.

To ensure the low degree condition, quadratic B–spline are used to approximate the given parametric curve. The quadratic spline segments are implicitized and joined together in order to generate a C^m bivariate function, see Fig. 1.

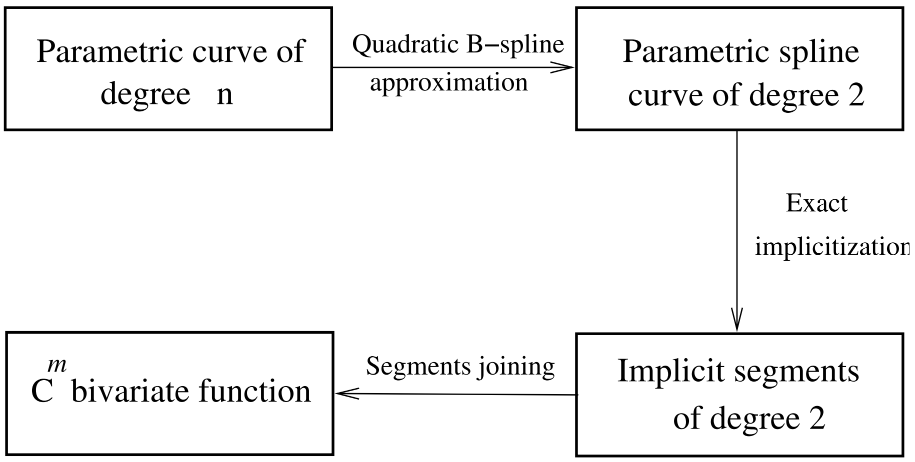

Figure 1 Low degree spline implicitization

In addition to avoiding the 3 problems of the exact implicitization. The spline implicitization 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 quadratic polynomial (for C^0 case) or quartic polynomial (for C^1 case).

The term *approximate implicitization* has been discussed by several authors: Montaudouin and Tiller employed a power series method to obtain local explicit approximation (about a regular point) to polynomial parametric curves and surfaces [26]. This approach is asymmetric as one variable is made an explicit function of the other(s).

Chuang and Hoffmann [3] extend this method using what they called “implicit approximation”. They propose a method for computing an implicit approximant at a point to a parametric curve or surface. The method works for both polynomially and rationally parameterized curves and surfaces and achieves an order of contact that can be prescribed. In the case of nonsingular curve points, the approximant must be irreducible, but in the surface case additional safeguards are incorporated into the algorithm to ensure irreducibility. The method also yields meaningful results at most singularities. In principle, the method is capable of exact implicitization and has a theoretical relationship with certain resultant–based elimination methods.

Dokken [8, 9] proposed a new way to approximate the parametric curve and surface “globally” in the sense that the approximation is valid within whole domain of the curve segment or surface patch. He presented a method for the approximation of piecewise rational parametric manifold of chosen degree with an algebraic hybersurface. This method provides good convergence rates, uses singular value decomposition to find a set of alternative approximations and enables the selection of an approximation to be based on required behavior of the gradient of the resulting approximation algebraic surface.

Sederberg et al. [41] employed 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 . High accuracy of the approximation is achieved with a relatively small number of low–degree curve segments or surface patches. By using monoid curves and surfaces, the method eliminates the undesirable singularities and unwanted branches normally associated with implicit representation. The monoids are expressed in exact implicit and parametric equations simultaneously, and upper bounds are derived for the approximate errors of implicitization and inversion equations.

The main goal of this work is to present in detailed techniques for finding a low degree spline implicit representation of a given parametric planar curve of degree n. To ensure the low degree condition, quadratic B–spline are used to approximate the given parametric curve (via orthogonal projection in Sobolev space). Adaptive knot removal, which is based on spline wavelets, is used to reduce the number of segments. The resulting quadratic segments are implicitized. Finally, by multiplying with suitable factors, these implicitized segments are joined such that the collection generate a C^m function, for $m = 0, 1$.

The proposed techniques is implemented in Maple. The Maple procedures can be downloaded from www.risc.uni-linz.ac.at/people/shalaby/phdthesis.

The remainder of this thesis is organized as follows:

- Chapter 1 discusses the problem of parametric curve approximation using quadratic B–spline (via orthogonal projection in Sobolev space).
- Chapter 2 shows how the data reduction works. We present a strategy for reducing the number of segments of the quadratic spline curve without perturbing it by more than a given tolerance. The proposed methods is based on spline wavelets.
- Chapter 3 presents a method for a segments–wise implicitization. The resulting quadratic spline segments, from the data reduction process, are implicitized after converting it into Bézier segments.
- Chapter 4 shows how the implicitized segments can be joined such that the collection generate a C^0 function.
- Chapter 5 shows how the implicitized segments can be joined such that the collection generate a C^1 function. In the Section 5.4, we give some applications of the proposed method.
- Chapter 6 as a possible extended research of the C^1 spline approximation technique, we propose a method for generating a hierarchy of planar shapes which it may help in the generating a hierarchy of functions.

Acknowledgments

In the course of my Ph. D, I am highly indebted to pass my heartfelt thanks to the many people who helped me in one way or another.

First of all, I would like to express my gratitude to Josef Schicho and Bert Jüttler, who helped me so much from my first steps to my latest research papers to improve in every aspect a scientist can be judged by. They are with no doubt the teachers who influenced my progress in science the most.

I would also like to thank Bruno Buchberger, Franz Winkler and everybody at RISC for the unforgettable years. I really enjoyed studying and working at RISC.

I want to thanks the Austrian science fund (FWF) for their support in the frame of the Special Research Program (SFB) F013 “Numerical and Symbolic Scientific Computing”, project 15.

I would like to acknowledge my true thanks to the Egyptian government for the financial support to my family during my Ph.D studies. I would also like to express my deep appreciation to the Egyptian Educational office in Vienna for all the help and support they generously offered to me during my stay in Austria.

A lot of people have contributed to my PhD, I would like to thank Alexandria university, the faculty of engineering, all my lecturers, colleagues, staff members, and students for their support during my M.Sc studies especially, Prof. Mahmoud Moustafa, Prof. Ahmed El Sherif, and Prof. Ahmed El Sheik.

Lastly, I want to thank my wife Heba and my children Mahmoud and Fathy for their constant support, and for her valuable assistance with all stages of my study. They have supported me with all the love in the world and patiently endured all the consequences of my being busy.

Contents

Chapter 1

Planar Curves Approximation

Piecewise polynomials are used extensively for the approximation of functions because of their good approximation properties and the ease with which they can be manipulated by the computer. To perform computations with piecewise polynomials, one must choose a suitable representation, i.e. a basis for the selected linear space of piecewise polynomials. A common choice is to use the B–splines as basis. These basis have many nice properties which usually lead to stable and simple algorithms.

In this chapter, we follow the technique proposed in [30], [31]. We generate a quadratic B–spline approximation of a given parametric curve via orthogonal projection in Sobolev space. First, we state some preliminary definitions. Then, we discuss the B–spline quadratic approximation process. Finally, we illustrate the proposed idea by giving some numerical examples.

1.1 Function Approximation

In the real word, functions play a fundamental role as a mathematical formalization for describing and analyzing a variety of physical processes. While in some cases these functions are known explicitly, very frequently it is necessary to construct approximations to them based on limited information about underlying processes. The main two categories of *approximation problem* are:

- *Data fitting problems* where it is required to construct an approximation to a function based on some finite amount of data on the function, see [17] and the references cited therein.

- *Operator–equation problem* where it is required to construct an approximation to a function based on operator equation that determine it, see [23, 31, 32, 33] and the references cited therein.

The most commonly used approach to finding approximations to functions proceeds as follows:

1. Choose a reasonable class of functions $\mathcal{P}$ in which to look for an approximation.

2. Devise an appropriate selection scheme for assigning a specified function to a specified problem.

The class $\mathcal{P}$ should passes at least the following basic properties:

- The functions in $\mathcal{P}$ should be smooth (differentiable);

- The functions in $\mathcal{P}$ should be easy to store and manipulate on the computer;

- The functions in $\mathcal{P}$ should be easy to evaluate on the computer, along with their derivatives and integrals;

- The class $\mathcal{P}$ should be large enough so that arbitrary smooth functions can be well approximated by elements of $\mathcal{P}$.

A good candidate for $\mathcal{P}$ is the class of polynomials.

1.1.1 Polynomials

Polynomials have played a central role in approximation theory and numerical analysis for many years. This is – among other reasons – due to the following facts:

- Polynomials are smooth (differentiable) functions;

- Polynomials are easy to store, manipulate, and evaluate on the computer;

- The derivative and integral of a polynomial are again polynomials whose coefficients can be found algebraically;

- The number of zeros of a polynomial of *order* (degree +1) k can not exceed $k - 1$;

- Given any continuous function on an interval $[a, b]$, there exists a polynomial which is uniformly close to it.

For small intervals, polynomials seem to be the most efficient solution but as we go for larger intervals severe oscillations often appear especially when k is more than 3 or 4. For this situation, in order get a greater flexibility for the approximation function we have to work with polynomials of low degree and split the interval into smaller pieces. We are motivated to use piecewise polynomials.

1.1.2 Piecewise Polynomials

DEFINITION 1. *Let* $a = t_0 < t_1 < \cdots < t_\ell < t_{\ell+1} = b$, *and write* $\triangle = \{t_i\}_0^{\ell+1}$. *The set* $\triangle$ *partitions the interval* $[a, b]$ *into* $\ell + 1$ *subintervals,* $I_i = [t_i, t_{i+1})$, $i = 0, 1, \cdots, \ell - 1$, *and* $I_\ell = [t_\ell, t_{\ell+1}]$. *Given a positive integer* k, *let*

$\{$ *f: there exist polynomials* $\mathcal{PP}_k(\triangle) = p_0, p_1, \cdots, p_\ell$ *in* $\mathcal{P}_k$ *with* $f(t) = p_i(t)$ *for* $t \in I_i$, $i = 0, 1, \cdots, \ell$ $\}$.

We call $\mathcal{PP}_k(\triangle)$ *the space of piecewise polynomials of order* k *with knots* $t_1, \cdots, t_\ell$.

By going from polynomials to small piecewise polynomials we gained flexibility but at the cost of losing the diffrentiability (piecewise polynomial functions are not necessarily differentiable). In fact, they can even be discontinuous, see Fig. 1.1.

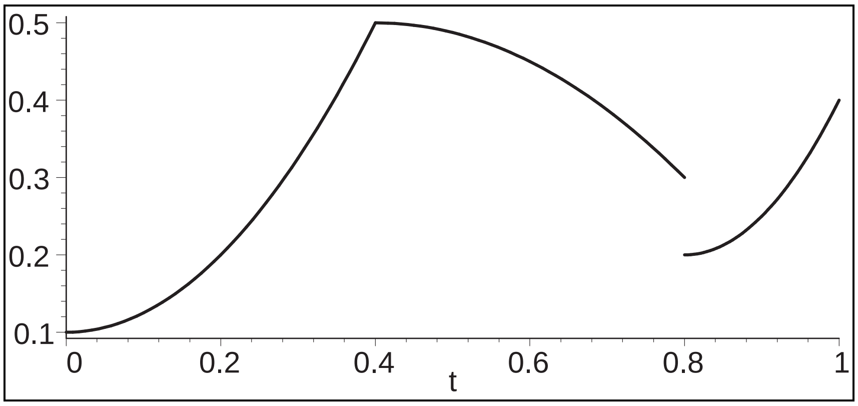

Figure 1.1 A piecewise polynomial with 3 knots

Piecewise polynomials are used extensively for the approximation of curves because of their good approximation properties. To perform computations with piecewise polynomials, one must choose a suitable representation, i.e. a basis for the selected linear space of piecewise polynomials. A common choice is to use the B–splines as basis. These basis have many nice properties which usually lead to stable and simple algorithms; see [7], [37].

1.1.3 B–splines

DEFINITION 2. *Let* $T = (t_0, t_1, t_2, \cdots, t_{n-1}, t_n, t_{n+1}, \cdots, t_{n+k})$ *to be a simple knot vector and* $t_0 < t_1 < t_2 < \cdots < t_{n-1} < t_n < t_{n+1} < \cdots < t_{n+k}$*, the associated normalized B–spline* $B_{j,k,T}$ *of order* k *is defined to be the following function:*

$$B_{j,1,T}(t) = \begin{cases} 1 & for\ t_j \leq t < t_{j+1} \\ 0 & otherwise \end{cases}$$

for $k = 1$*, and*

$$B_{j,k,T}(t) = \frac{(t - t_j)}{(t_{j+k-1} - t_j)} B_{j,k-1,T}(t) + \frac{(t_{j+k} - t)}{(t_{j+k} - t_{j+1})} B_{j+1,k-1,T}(t) \tag{1.1}$$

for $k > 1$ *and* $j = 0, 1, \cdots, n$*.*

These basis functions have the following properties:

- $B_{j,k,T}(t) > 0$ $\quad for\ t_j < t < t_{j+k};$
- $B_{j,k,T}(t) = 0$ $\quad for\ t_0 \leq t \leq t_j, \quad t_{j+k} \leq t \leq t_{n+k};$
- $\sum_{j=0}^{n} B_{j,k,T}(t) = 1$ $\quad t \in [t_{k-1}, t_{n+1}];$
- $B_{j,k,T}(t)$ has continuity C^{k-2} at each knots t_ℓ.

REMARK 1.

- *The B–splines form a partition of unity;*
- *The function* $B_{j,k,T}$ *vanishes outside the interval* $[t_j, t_{j+k}]$*;*
- *For equally spaced knots* $T = h\mathbb{Z}$*, the recurrence relation (1.1) becomes*

$$B_{j,k,h}(t) = \frac{(t - jh)}{(k-1)h} B_{j,k-1,h}(t) + \frac{((j+k)h - t)}{(k-1)h} B_{j+1,k-1,h}(t)$$

and we refer to the $B_{j,k,h}$ *as the uniform B–splines. In the case* $h = 1$*, we called* $B_{j,k,1} = B_{j,k}$ *the cardinal B–splines.*

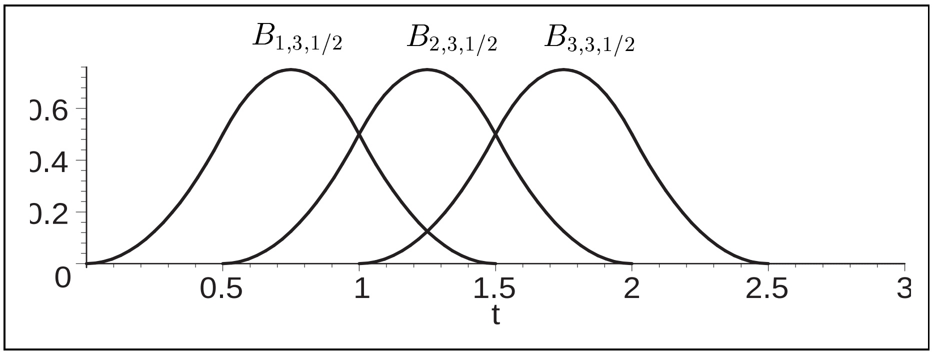

Figure 1.2 Some uniform quadratic B–splines defined over $\mathbb{Z}/2$

1.2 B–spline Approximation of Planar Curves

Function approximation is a process where a given function $\mathbf{f}$ is replaced by a *simpler function* $\mathbf{f}^*$ in such way as to minimize a prescribed error measure. In order to construct an approximation, we look for an *optimal* function $\mathbf{f}^*$ as linear combination of given basis function (B–spline is a good candidate), where optimal means that a prescribed *error norm* should be as small as possible.

Approximation of functions by B–splines is a typical example where the orthonormality is advantage. A basis $B_1, B_2, \cdots$ is orthonormal if:

$$\langle B_i, B_j \rangle = \delta_{i,j}$$

where $\delta_{i,j}$ is called the *Kronecker delta* and is defined to be 1 if $i = j$ and 0 otherwise, and $\langle ., . \rangle$ is an inner product.

In order to approximate a given planar curve $\mathbf{g}$ by B–spline:

- First, we should select an inner product that makes the B–spline basis form an orthonormal sequence.
- Then, the B–spline approximation can be computed easily.

In what follows, we briefly sketch the idea of approximation of planar curves following the technique proposed in [30].

1.2.1 Inner Product Selection

An inner product $\langle ., . \rangle$ on a vector space V is any map from $V \times V$ to $\mathbb{R}$ that is

- symmetric: $\langle u, v\rangle = \langle v, u\rangle$
- bilinear: $\langle au + bv, w.\rangle = a\langle u, w.\rangle + b\langle v, w\rangle$
- positive definite: $\langle u, u\rangle > 0$ for all $u \neq 0$

A Sobolev space $H^{k-1,2}(\mathbb{R})$ is a vector space of functions. As pointed out in [30], the uniform $B_{j,k,h}$, $j \in \mathbb{Z}$, $k \in \mathbb{N}$, $h \in \mathbb{Q}$ of *order* k with knot sequence $h\mathbb{Z}$, where h is the step size, form an orthonormal sequence in Sobolev space $H^{k-1,2}(\mathbb{R})$ endowed with the inner product

$$\langle f, g\rangle = \sum_{\mu=0}^{k-1} w_\mu(k, h)\ (\partial^\mu f, \partial^\mu g) \tag{1.2}$$

and the norm

$$\|f\|^2_{w(k,h)} = \sum_{\mu=1}^{k-1} w_\mu(k, h)\|\partial^\mu f\|^2 \tag{1.3}$$

where $(.,.)$ is the usual L^2 inner product and

$$w_\mu(k, h) = h^{2\mu-1} w_\mu(k)$$

for certain positive weights $w_\mu(k)$. These weights are specified explicitly in [30]. Table 1.1 shows the weights $w_\mu(k)$ for $k \leq 7$.

d	$w_0(k)$	$w_1(k)$	$w_2(k)$	$w_3(k)$	$w_4(k)$	$w_5(k)$	$w_6(k)$
1	1						
2	1	$\frac{1}{6}$					
3	1	$\frac{1}{4}$	$\frac{1}{30}$				
4	1	$\frac{1}{3}$	$\frac{7}{120}$	$\frac{1}{140}$			
5	1	$\frac{5}{12}$	$\frac{13}{144}$	$\frac{41}{3024}$	$\frac{1}{630}$		
6	1	$\frac{1}{2}$	$\frac{31}{240}$	$\frac{139}{6048}$	$\frac{479}{151200}$	$\frac{1}{2772}$	
7	1	$\frac{7}{12}$	$\frac{7}{40}$	$\frac{311}{8640}$	$\frac{37}{6480}$	$\frac{59}{79200}$	$\frac{1}{12012}$

Table 1.1 Weights $w_\mu(k)$ for $k \leq 7$

In [31], the orthogonality relations over $\mathbb{R}$ are adapted to the case of *finite intervals* with integer bounds, $r = [hr_1, hr_2]$, $r_i \in \mathbb{N}$, $h(r_2 - r_1) \geq k - 1$, which is of importance for many applications. With $f^{(k)} = (f, \partial f, \cdots, \partial^{k-2} f)$ the inner product is of the type:

$$\langle f, g \rangle_{A_{1,2}(k,h)} = \sum_{i=1}^{2} f^{(k)}(r_i) A_i(k,h) g^{(k)}(r_i)^T + \langle f, g \rangle \tag{1.4}$$

In [31], the matrices $A_i(k,h)$ are specified explicitly and shown to be positive definite for $k \leq 8$.

1.2.2 B–Spline Approximation

The orthogonality relations derived in the preceding subsection can be used for efficiently approximating functions by B-splines. Denote by $S_{k,h}(r)$ the space of uniform splines with knots $h\mathbb{Z}$ over the interval $[hr_1, hr_2]$. Then the B–splines $B_{j,k,h}$, $r_1 - k < j < r_2$ restricted to r form a basis of $S_{k,h}(r)$.

The B–spline approximation of a given curve $\mathbf{g}$ can be written as:

$$\mathbf{g}^* = \sum_{j \in \mathbb{Z}} (Q_h \mathbf{g})_j B_{j,k,h}$$

where $(Q_h \mathbf{g})_j$ are the B–spline coefficients and $B_{j,k,h}$ are the B–spline basis.

REMARK 2. *As pointed out in [30], [31]:*

- *The B–spline coefficients of the solution* $Q_h \mathbf{g} := \sum_{j \in \mathbb{Z}} (Q_h \mathbf{g})_j B_{j,k,h}$ *are given by:*

  $$(Q_h \mathbf{g})_j = \langle \mathbf{g}, B_{j,k,h} \rangle$$

  and the projection $Q_h := H^{k-1,2}(\mathbb{R}) \to S_{k,h}$ *is local in the sense that* $(Q_h \mathbf{g})(x)$ *depends only on the restriction of* $\mathbf{g}$ *to the interval* $[x - kh, x + kh]$.

- *There exists a constant* C *depending only on* k *such that for all* $\ell \in \{0, \ldots, k-1\}$

  $$\|\partial^\ell (\mathbf{g} - Q_h \mathbf{g})\|_\infty \leq C h^{k-\ell} \|\partial^k \mathbf{g}\|_\infty$$

  i.e., the approximation order of Q_h *is optimal as* $h \to 0$.

- *The numerical evaluation of (1.2) can be made efficient by using the identity*

$$(\partial^\mu \mathbf{g},\ \partial^\mu B_{j,k,h}) = \begin{cases} (-1)^\mu (\mathbf{g},\ \partial^{2\mu} B_{j,k,h}) & if \quad 2\mu < k \\ (-1)^\mu h\ k!\ [hj, \cdots, h(j+k)] \partial^{2\mu - k}\ \mathbf{g} & if \quad 2\mu \geq k \end{cases} \tag{1.5}$$

The benefit is that the derivatives of **g** *have to be evaluated solely at the knots and not at the multitude of arguments required by the quadrature scheme. Eq. (1.5) is based on repeated integration by parts. In the second case, the process stops, when the B–spline has transformed to* $\partial^{k-1}B_{j,k,h}$, *which is piecewise constant. Thus, the integral can be evaluated and yields the given divided difference.*

1.2.3 Quadratic B–spline Approximation

Using the idea discussed above, we generate a quadratic B–spline approximation via orthogonal projection in Sobolev spaces. The quadratic B–splines $B_{j,3,h}$ on $[0,1]$, see Fig. 1.3, with uniform knots T, stepsize $(h = 2^{-i})$ and 3-fold boundary knots:

$$T = (t_1, \cdots, t_{2^i+5}) = \frac{1}{2^i}(0,0,0,1,2,\cdots,2^i-2,2^i-1,2^i,2^i,2^i)$$

form an orthonormal sequence in a suitably weighted Sobolev space. From Eq. (1.2) and Table. 1.1, in the interior of the segment, the inner product is defined as:

$$\langle f,g\rangle = \frac{1}{h}(f,g) + \frac{1}{4}h\,(f',g') + \frac{1}{30}h^3\,(f'',g'') \tag{1.6}$$

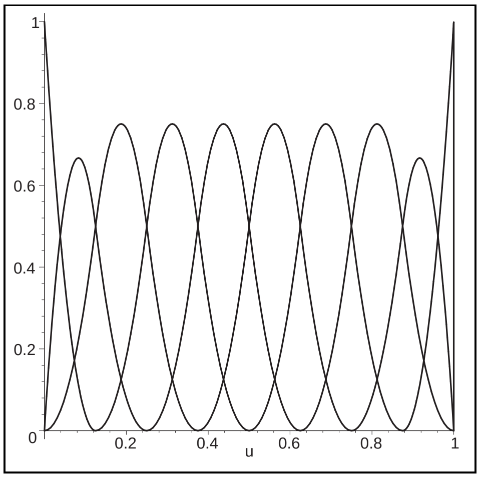

Figure 1.3 The uniform quadratic B–splines defined over T $(i = 3)$

In order to achieve orthogonality at the boundary segments, additional terms have to be used. The inner product, in this case, is defined as:

$$\begin{aligned}\langle f,g\rangle = \;& \tfrac{1}{h}(f,g) + \tfrac{1}{4}h\,(f',g') + \tfrac{1}{30}h^3\,(f'',g'') \\ &+ \textstyle\sum_{i=1}^{2}(f(r_i)\;f'(r_i))\,A_i(3,h)\,(g(r_i)\;g'(r_i))^T\end{aligned} \tag{1.7}$$

These weight matrices (which are used near the boundary), have been specified in [31]. Here we list the weight matrices in the quadratic case:

$$A_1(3,h) = \begin{pmatrix} 1 & \frac{-h}{3} \\ \frac{-h}{3} & \frac{h^2}{4} \end{pmatrix} \qquad\qquad A_2(3,h) = \begin{pmatrix} 1 & \frac{h}{3} \\ \frac{h}{3} & \frac{h^2}{4} \end{pmatrix}$$

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 products (1.6, 1.7), can then be written as:

$$\mathbf{g}^* = \sum_j (Q_h\mathbf{g})_j \; B_{j,3,h}(t) \qquad \text{with} \qquad (Q_h\mathbf{g})_j = \begin{pmatrix} \langle g_1, B_{j,3,h} \rangle \\ \\ \langle g_2, B_{j,3,h} \rangle \end{pmatrix}$$

The control points $(Q_h\mathbf{g})_j$ of the approximating 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.

1.3 Numerical Examples

Throughout the thesis, the following numerical examples will be used to illustrate the idea of the thesis. These examples has been solved by Maple using our self implemented Maple procedures. A complete Maple work sheets for these examples and all necessary Maple procedures can be downloaded from www.risc.uni-linz.ac.at/people/shalaby/phdthesis

EXAMPLE 1.

Given: *A parametric curve* $\mathbf{g}$ *of degree 20 defined over the interval* $u \in [0,1]$

Find: *A quadratic approximation* $\mathbf{g}^*$ *of* $\mathbf{g}$.

Considering the problem as two separate problems of approximation for both X *and* Y *components. Figure 1.4 shows the error between* $\mathbf{g}$ *(black) and* $\mathbf{g}^*$ *(gray) for different step sizes* h. *The error is exaggerated by a factor* δ *to make it visible.*

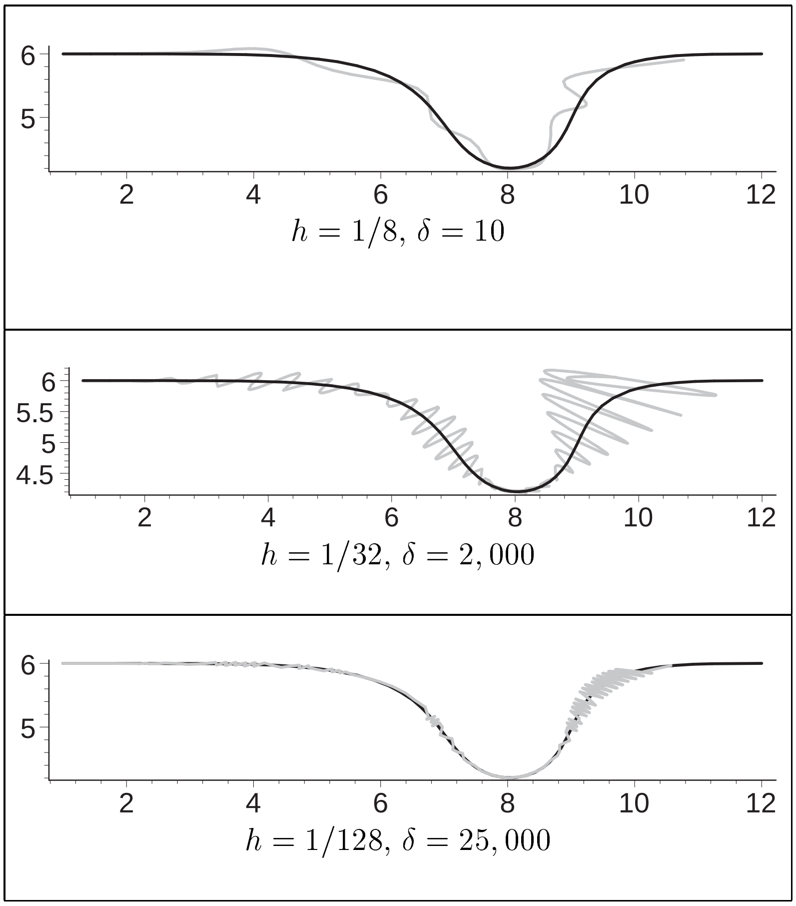

Figure 1.4 The original curve (black) and the error introduced by the quadratic B–spline approximation (gray) for different stepsize h. The error has been exaggerated by factor δ.

EXAMPLE 2.

Given: *A parametric curve* $\mathbf{g}$ *of degree 15.*

Find: *A quadratic approximation* $\mathbf{g}^*$ *of* $\mathbf{g}$.

Considering the problem as two separate problems of approximation for both X and Y components. Figure 1.5 shows the error between $\mathbf{g}$ *(black) and* $\mathbf{g}^*$ *(gray) for different step sizes* h*. The error is exaggerated by a factor* δ *to make it visible.*

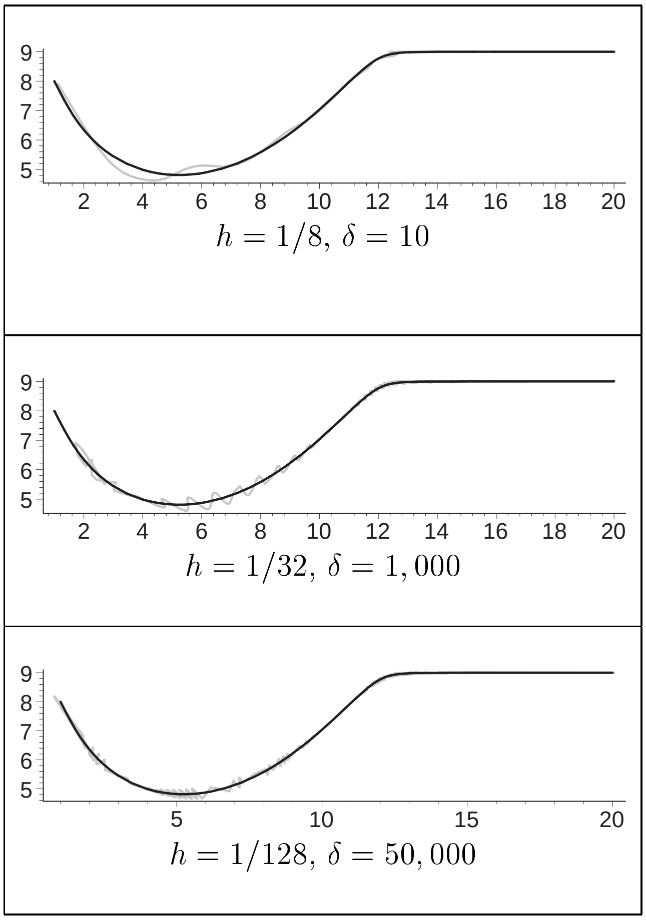

Figure 1.5 The original curve (black) and the error introduced by the quadratic B–spline approximation (gray) for different stepsize h. The error has been exaggerated by factor δ.

EXAMPLE 3.

Given: *A piecewise parametric curve* $\mathbf{g}$ *consist of three pieces of degree 10 each.*

Find: *A quadratic approximation* $\mathbf{g}^*$ *of* $\mathbf{g}$.

Considering the problem as two separate problems of approximation for both X and Y components. Figure 1.6 shows the error between $\mathbf{g}$ *(black) and* $\mathbf{g}^*$ *(gray) for different step sizes* h. *The error is exaggerated by a factor* δ *to make it visible.*

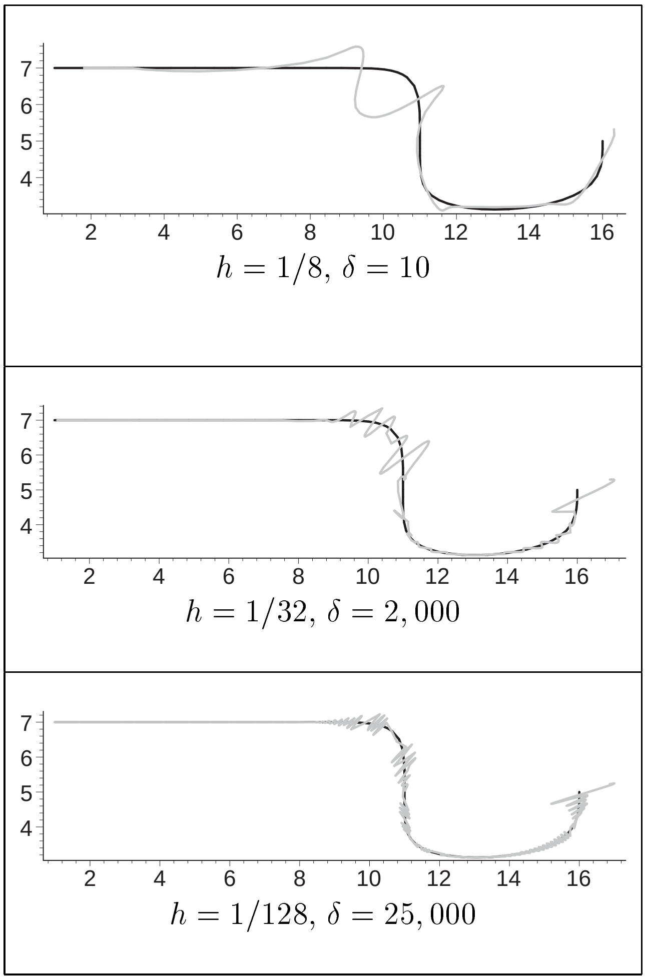

Figure 1.6 The original curve (black) and the error introduced by the quadratic B–spline approximation (gray) for different stepsize h. The error has been exaggerated by factor δ.

Chapter 2

Data Reduction

Using the B–spline approximation method derived in the previous chapter, an arbitrarily accurate approximation can be generated depending on the number of spline segments (number of knots). In order to get an accurate approximation, sufficiently many segments have to be used.

In order to reduce the number of spline segments, a data reduction (knot removal) technique is used. A new method, based on spline wavelets, is proposed. The method is not optimal but it is cheaper than existing methods.

This chapter is organized as follow: First, we give a brief introduction about the data reduction process. Then, in the second section, we show how B–spline wavelets can be used for data reduction. Finally, we give some numerical examples.

2.1 Data Reduction Process

The idea behind data reduction is to start with an initial curve approximation which is easy to compute, and give a good approximation given by a large number of parameters. Then knot removal can be performed to reduce the number of parameters, but hopefully retain the quality of the approximation. Such a reduction means that we approximate the given B–spline in a space $S_{k,h}$ defined over the knot sequence T by a B–spline in subspace $\hat{S}_{k,h}$ of $S_{k,h}$ defined over a knot sequence $\hat{T} \subset T$. In other words, the number of degrees of freedom has been reduced and we have obtained a data reduction. We will use the terms *" data reduction"* and *"knot removal"* interchangeably to denote this process.

Let $\mathbf{g}^*$ be an element of $S_{k,h}$ and ε a given nonnegative real number. Suppose also that the norm $\|.\|_{w(k,h)}$ on $S_{k,h}$ has been chosen. Our problem is to determine a subspace $\hat{S}_{k,h}$ of $S_{k,h}$ of lower dimension, and an element $\mathbf{g}^{**}$ of $\hat{S}_{k,h}$ such that $\|\mathbf{g}^* - \mathbf{g}^{**}\|_{w(k,h)} < \varepsilon$.

Some work has been published on the problem of knot removal. Related problems (dealing mainly with the removing of one knot) are considered in [21], [11], and [36]. An algorithm for removing all knots which can be removed exactly within the accuracy of the computer has been proposed in [44]. In [24], [25] a strategy is proposed to approximatively remove more than one knot by first ranking the knots and afterwards eliminating all knots with small weights by global approximation scheme. In [10], the problem of removing one inner knot from the knot sequence of a spline curve is discussed. An optimal technique has been proposed, the idea is to do 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 and evaluate the error introduced by removing specific knot.

2.2 Data Reduction via Wavelets

In [19], a special method for our special situation is proposed. We proposed a new technique that utilizes spline wavelets for knot removal. The proposed method is not optimal but it is cheaper than other methods since no sorting or ranking lists are required. For the convenience of the reader, an outline of the method is given in subsection 2.2.3. The reader, who is familiar with wavelets, may wish to proceed directly to subsection 2.2.3.

2.2.1 Wavelets

Wavelets are a mathematical tool for hierarchically decomposing functions. In addition to the hierarchical nature, they have many useful properties, such as linear time complexity, sparsity and adaptability. A wavelet transform of a function consists of a coarse overall approximation together with detail coefficients that influence the function at various scales.

Storing the wavelet transform of a function, rather than the function itself, has number of advantages. One advantage of the wavelets transform is that often a large number of the detail coefficients turn out to be *very small* *in magnitude*. Removing these small coefficients from the representation introduces only *small errors* in the reconstructed function.

Let us think of every function as a vector of coefficients in a vector space V^i. The basis functions for the space V^j are called *scaling functions* and are usually denoted by the symbol ϕ_i^j. Note that every vector in V^j is also contained in V^{j+1}, i.e. $V^0 \subset V^1 \subset V^2 \subset \cdots$.

Let the vector space W^j and the basis ψ_i^j be defined as the wavelet space and wavelet functions respectively. One can think about the wavelets in W^j as a means of representing the parts of a function in V^{j+1} that can not represented in V^j. Thus, the detail coefficient are coefficients of the wavelets basis functions.

In order to proceed more easily, matrix will be used. Let Φ^j be a row matrix containing all the scaling functions with superscript j; that is,

$$\mathbf{\Phi}^j = (\phi_0^j \; \phi_1^j \; \cdots)$$

Similarly, the wavelets functions can be written as:

$$\mathbf{\Psi}^j = (\psi_0^j \; \psi_1^j \; \cdots)$$

Let $v(j)$ and $w(j)$ be the dimension of V^j and W^j respectively. Because W^j is the complement of V^j in V^{j+1}, the dimension of these spaces satisfy $v(j+1) = v(j)+w(j)$. As mentioned in [43], Section 6.4, for all $j = 1, 2, 3, \cdots$ there exist a $v(j) \times v(j-1)$ matrix $\mathbf{P}^j$ and a $v(j) \times w(j-1)$ matrix $\mathbf{Q}^j$ such that:

$$\mathbf{\Phi}^{j-1} = \mathbf{\Phi}^j \; \mathbf{P}^j; \qquad \mathbf{\Psi}^{j-1} = \mathbf{\Phi}^j \; \mathbf{Q}^j$$

Note that the above equations can be also expressed as a single equation:

$$(\mathbf{\Phi}^{j-1} | \mathbf{\Psi}^{j-1}) = \mathbf{\Phi}^j \; (\mathbf{P}^j | \mathbf{Q}^j) \tag{2.1}$$

Consider a function in some approximation space V^j. Let us assume that, we have the coefficients of this function in term of some scaling function basis. We can write these coefficients as a column matrix of values $\mathbf{c}^j = (c_0^j \cdots c_{v(j)-1}^j)^T$. The low level version $\mathbf{c}^{j-1}$ of $\mathbf{c}^j$ can be expressed as:

$$\mathbf{c}^{j-1} = \mathbf{A}^j \; \mathbf{c}^j$$

where $\mathbf{A}^j$ is a constant $v(j-1) \times v(j)$ matrix.

Since $\mathbf{c}^{j-1}$ contains fewer entries than $\mathbf{c}^j$, it is intuitively clear that some amount of detail is lost. If $\mathbf{A}^j$ is appropriately chosen, it is possible to capture the lost detail as another column matrix $\mathbf{d}^{j-1}$, computed by

$$\mathbf{d}^{j-1} = \mathbf{B}^j\ \mathbf{c}^j$$

where $\mathbf{B}^j$ is a constant $w(j-1) \times v(j)$ matrix related to $\mathbf{A}^j$.

Similar to Eq. 2.1, the above two equations can also expressed as a single equation:

$$\begin{pmatrix} \mathbf{c}^{j-1} \\ \hline \mathbf{d}^{j-1} \end{pmatrix} = \begin{pmatrix} \mathbf{A}^j \\ \hline \mathbf{B}^j \end{pmatrix}\ \mathbf{c}^j \tag{2.2}$$

2.2.2 B-spline Wavelets

Spline wavelets, which are built from B–splines, have been developed to a large extent in [4], [5]. Of particular interest for curve editing applications are endpoint interpolating B–splines. These wavelets are *semiorthogonal wavelets*[1].

In what follows, we use the quadratic B–spline wavelets for knot removal process. The quadratic B–spline scaling functions (which is produced as in the previous chapter) are defined over the knot sequence T:

$$T = (t_1, \cdots, t_{2^j+5}) = \frac{1}{2^j}(0, 0, 0, 1, 2, \cdots, 2^j - 2, 2^j - 1, 2^j, 2^j, 2^j)$$

In this case, $V^j(2)$ is defined as the set of $2^j + 2$ quadratic B–splines defined over T.

To complete our development of the quadratic B–spline wavelets, we need to find a basis functions for the spaces W^j that are orthogonal complements to the spaces V^j. In other word, we need to find the matrices $\mathbf{P}^j$ and $\mathbf{Q}^j$ for the quadratic B–spline wavelets. These matrices are specified implicitly in [43], Appendix B.

In order to determine $\mathbf{c}^{j-1}$ and $\mathbf{d}^{j-1}$ from $\mathbf{c}^j$, for B–spline wavelets, one may use the following relation:

$$\begin{pmatrix} \mathbf{A}^j \\ \hline \mathbf{B}^j \end{pmatrix} = (\mathbf{P}^j|\mathbf{Q}^j)^{-1}; \qquad (\mathbf{P}^j|\mathbf{Q}^j)\ \begin{pmatrix} \mathbf{c}^{j-1} \\ \hline \mathbf{d}^{j-1} \end{pmatrix} = \mathbf{c}^j$$

[1]Wavelets of a given level are a least orthogonal to all coarser scaling function, though not necessarily orthogonal to each other. In mathematical terms, a semiorthogonal wavelet basis satisfy the conditions $\langle \phi^j_k | \psi^j_\ell \rangle = 0$ for all j, k, ℓ

Fig. 2.1 shows the quadratic B–spline scaling functions (to the left) and wavelets basis function (to the right) for $j = 2$. More details about spline wavelets can be found in [4, 43, 45].

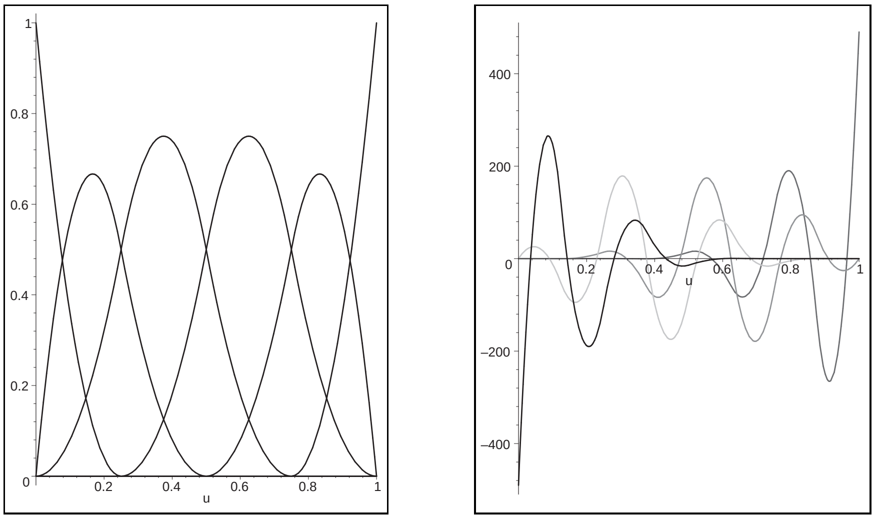

Figure 2.1 The quadratic B–spline scaling functions (to the left) and wavelets (to the right) for $j = 2$

2.2.3 Data Reduction via Quadratic B–spline Wavelets

Starting from a curve defined on a certain number of knots sufficient to guarantee the desired accuracy, we are interested in removing knots from representation, without perturbing it more than a given tolerance.

1. First, the quadratic B–spline wavelets transform of the given curve is computed;

2. Then, by setting all wavelets coefficients vectors with norm less than the threshold $\mathcal{T}$ to be zero, we can remove blocks of zero coefficient 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 spline final representation $\mathbf{g}^{**}$ of the given spline curve $\mathbf{g}^{*}$ is computed over the final "reduced" knot sequence $\hat{T}$.

The error can be bounded simply by applying the wavelets 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.

We conclude that, the threshold $\mathcal{T}$ should be selected such that the error be less than the given tolerance.

EXAMPLE

Starting from Example 1 in the previous chapter, according to our experimental results:

- The relation between the maximum error and the threshold $\mathcal{T}$ seem to be almost linear. Figure 2.2, shows the logarithmic plot of the maximum error and the threshold $\mathcal{T}$.

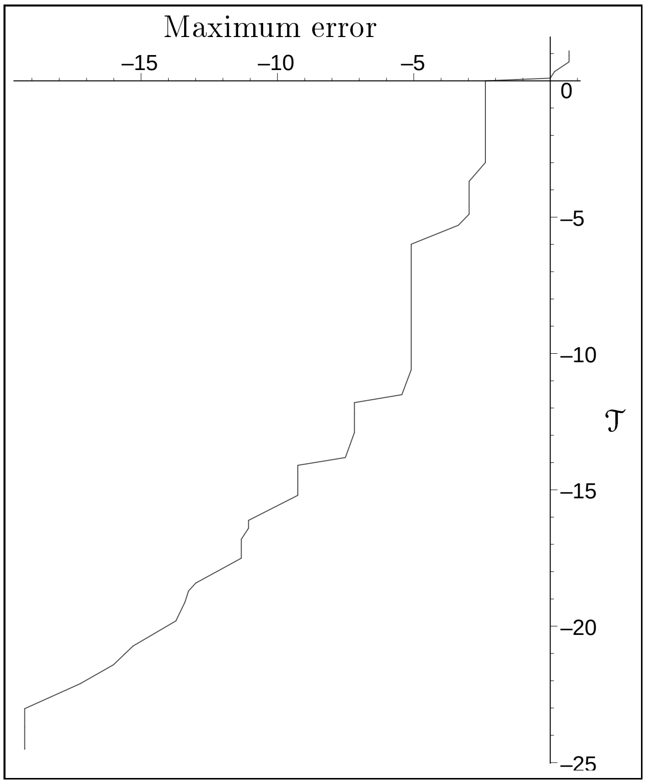

Figure 2.2 The logarithmic plot of the maximum error and the threshold $\mathcal{T}$.

- Figure 2.3, shows the relation between the logarithmic value of the threshold $\mathcal{T}$ and the number of non removed wavelets.

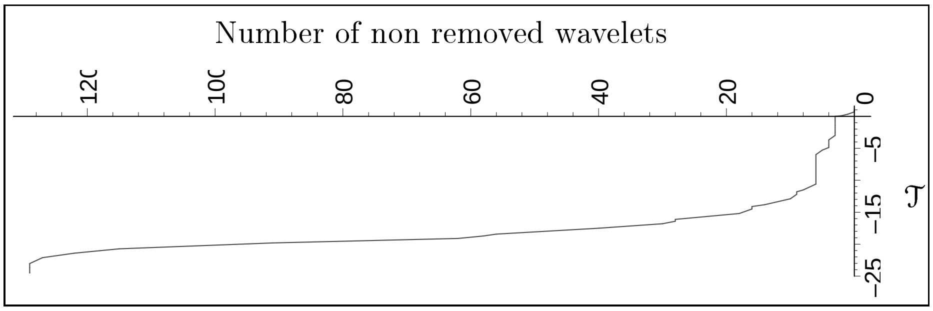

Figure 2.3 The relation between the logarithmic value of the threshold $\mathcal{T}$ and the number of non removed wavelets

- Figure 2.4 shows – on a logarithmic scale – the values of wavelets coefficients of the curve $\mathbf{g}^*$ (from the case $h = 1/128$) before (to the left) and after (to the right) data reduction. The threshold (10^{-4}) is plotted as thick black line.

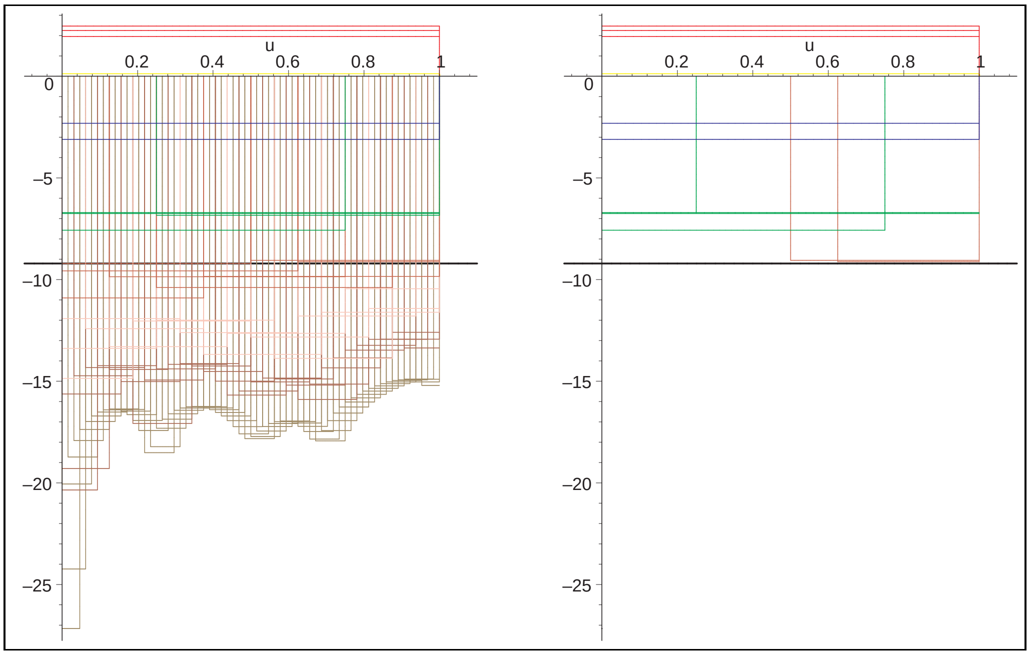

Figure 2.4 The logarithmic value of wavelets coefficients before (left) and after (right) data reduction.

2.3 Numerical Examples Continued

EXAMPLE 1. Continued

Starting from the quadratic spline curve $\mathbf{g}$ which is shown in Fig. 1.4, the case $h = 1/128$. Applying the data reduction process, the number of knots is reduced from 133 to 26 in the case of threshold equal to 10^{-5}. For the case of the threshold 10^{-4} the number of knots is reduced to 17. Finally, in the case of the threshold 10^{-3} the number is reduced to 13. Figure 2.5 shows the error between the original curve $\mathbf{g}$ (black) and the final representation "after the data reduction process" $\mathbf{g}^{**}$ (gray) over $\hat{T}$ for different threshold $\mathcal{T}$. The knots are plotted as small circles, note that, the knots at the boundary are counted with multiplicity 3. The error is exaggerated by a factor δ to make it visible.

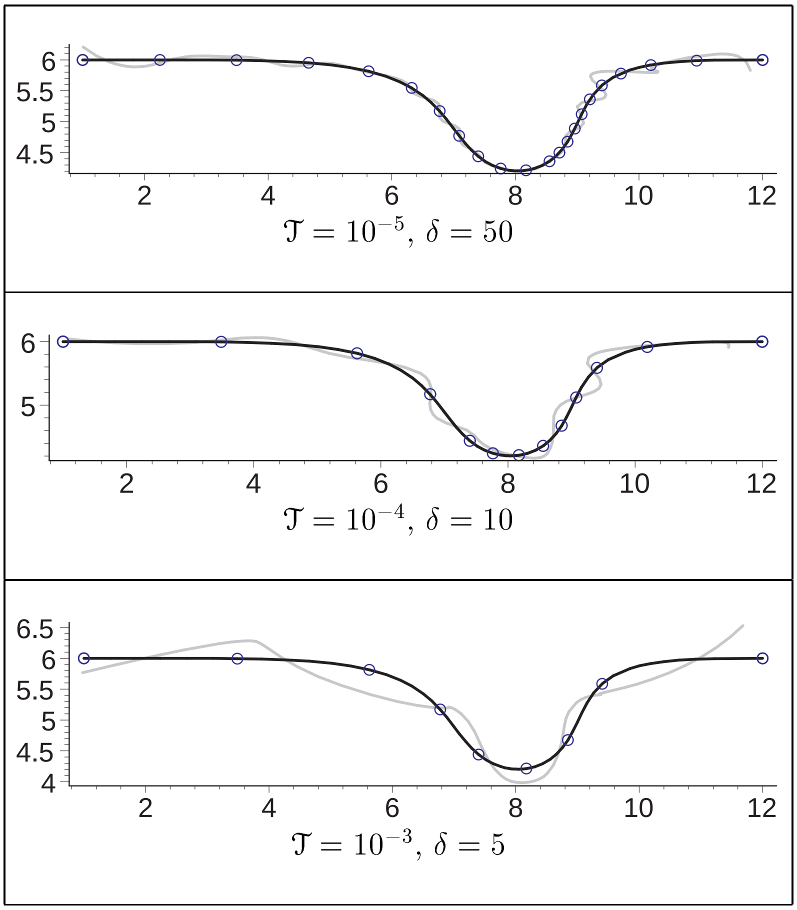

Figure 2.5 The original curve (black) and the error introduced after knot removal for threshold $\mathcal{T}$. The error has been exaggerated by factor δ.

EXAMPLE 2. Continued

Starting from the quadratic spline curve $\mathbf{g}^*$ which is shown in Fig. 1.5, the case $h = 1/128$. Applying the data reduction process, the number of knots is reduced from 133 to 31 in the case of threshold equal to 10^{-5}. For the case of the threshold 10^{-4} the number of knots is reduced to 21. Finally, in the case of the threshold 10^{-3} the number is reduced to 13. Figure 2.6 shows the error between the original curve $\mathbf{g}$ (black) and the final representation "after the data reduction process" $\mathbf{g}^{**}$ (gray) over $\hat{T}$ for different threshold $\mathcal{T}$. The knots are plotted as small circles, note that, the knots at the boundary are counted with multiplicity 3. The error is exaggerated by a factor δ to make it visible.

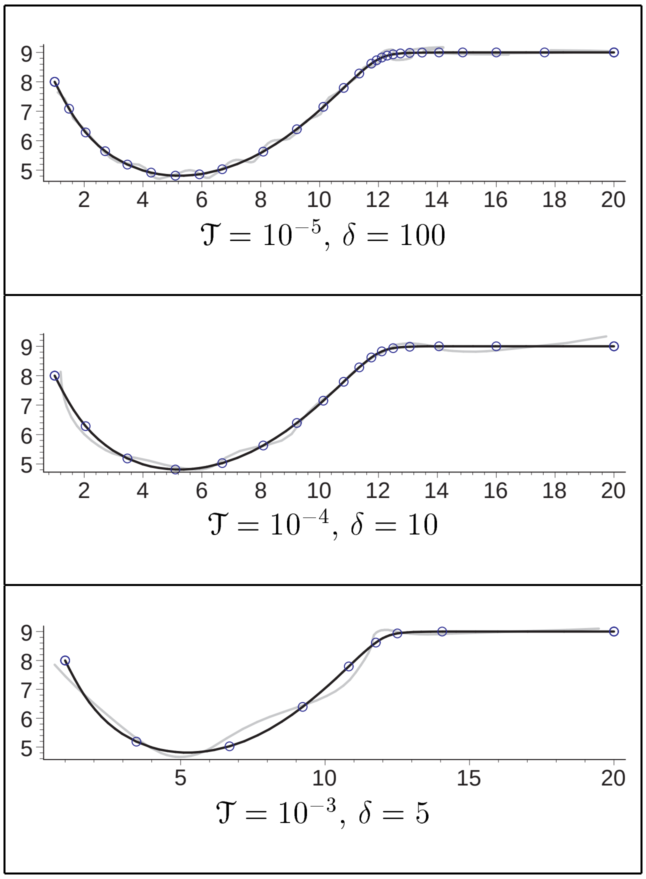

Figure 2.6 The original curve (black) and the error introduced after knot removal for threshold $\mathcal{T}$. The error has been exaggerated by factor δ.

EXAMPLE 3. Continued
Starting from the quadratic spline curve $\mathbf{g}^*$ which is shown in Fig. 1.6, the case $h = 1/128$. Applying the data reduction process, the number of knots is reduced from 133 to 37 in the case of threshold equal to 10^{-5}. For the case of the threshold 10^{-4} the number of knots is reduced to 21. Finally, in the case of the threshold 10^{-3} the number is reduced to 13. Figure 2.7 shows the error between the original curve $\mathbf{g}$ (black) and the final representation "after the data reduction process" $\mathbf{g}^{**}$ (gray) over $\hat{T}$ for different threshold $\mathcal{T}$. The knots are plotted as small circles, note that, the knots at the boundary are counted with multiplicity 3. The error is exaggerated by a factor δ to make it visible.

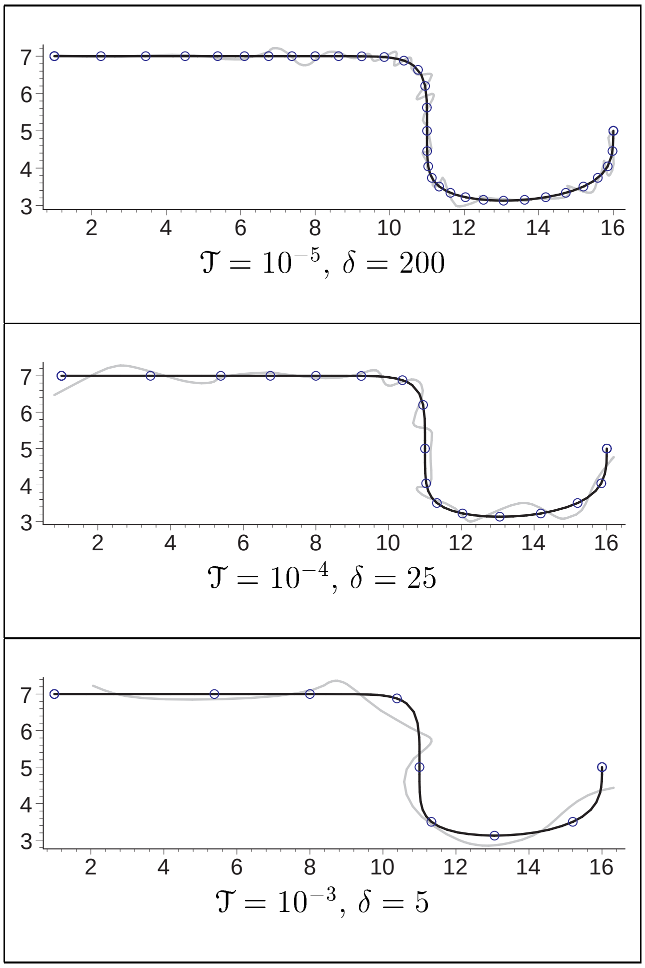

Figure 2.7 The original curve (black) and the error introduced after knot removal for threshold $\mathcal{T}$. The error has been exaggerated by factor δ.

Chapter 3

Spline Implicitization

Spline implicitization consists of a partition of the plane into polygonal pieces by transversal (partition) lines, and a bivariate polynomial G_i for each piece. On the boundaries, these bivariate polynomial pieces are joined together with C^m continuity for suitable choice of m (in our work $m = 0, 1$).

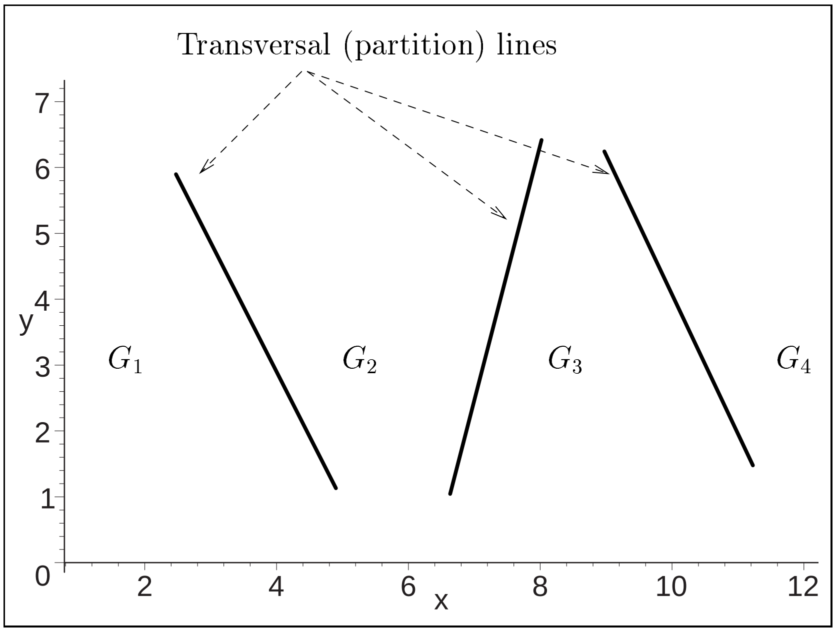

Figure 3.1 Spline Implicitization

In the following, first, we discuss segments implicitization process. Then, we show how these implicitized segments can be joined along the transversal lines.

3.1 Segment-wise Implicitization

After applying data reduction, we obtain a quadratic B–spline approximation $\mathbf{g}^{**}$ defined over the reduced (non uniform) knot sequence $\hat{T}$. In order to implicitize this curve:

- First, we split the representation of the curve into Bézier representation;
- Then, each quadratic Bézier segment is implicitized.

3.1.1 From B–spline to Bézier representation

Conversion from B–spline representation of the curve to Bézier representation can be easily done via knot insertion.

Inserting a parameter value t^* in a knot vector $T = [t_0, t_1, \cdots, t_{n-1}, t_n]$ between t_r and t_{r+1} gives a new knot vector T^* with knots:

$$\begin{aligned} t_i^* &= t_i && 0 \le i \le r, \\ t_{r+1}^* &= t^* && t^* \in [t_r, t_{r+1}], \\ t_{i+1}^* &= t_i && r+1 \le i \le n. \end{aligned}$$

The new B–spline coefficients (control points) $\mathbf{d}_i^*$ can be computed from the old one $\mathbf{d}_i$ with the help of T^*, T and the B–spline order (degree +1) d as follows:

$$\begin{aligned} &\mathbf{d}_i^* = \mathbf{d}_i && 0 \le i \le r-d+1, \\ &\mathbf{d}_i^* = (1-\alpha_i)\,\mathbf{d}_{i-1} + \alpha_i\,\mathbf{d}_i && r-d+2 \le i \le r, \\ &\mathbf{d}_{i+1}^* = \mathbf{d}_i && r \le i \le n. \end{aligned}$$

with

$$\alpha_i = \frac{t_{r+1}^* - t_i}{t_{i+d-1} - t_i}$$

By increasing the knot multiplicity at each knot to be equal to the degree of the curve (in our case to be 2), the representation is splitted into Bézier representation. Fig. 3.2 shows both B–spline coefficients $\mathbf{d}_i$ plotted as circles and the Bézier coefficients $\mathbf{d}_i^*$. The additional Bézier coefficients are plotted as diamonds. More about knot insertion can be found in [12, 15].

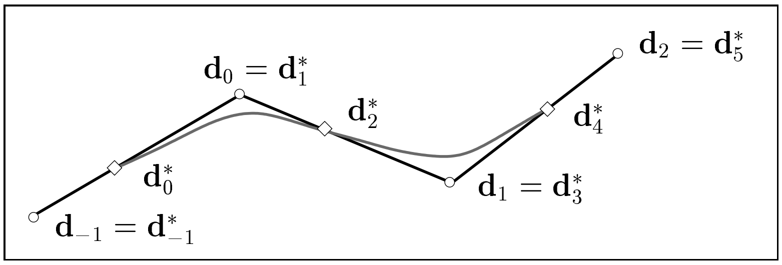

Figure 3.2 The B–spline control points (circles) and the additional Bézier control points (diamonds)

3.1.2 Bézier segments implicitization

It is well known that 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 parametric form of this segment is equal to:

$$\begin{pmatrix} p_0\,(1-t^2) + 2p_1\,(1-t)\,t + p_2\,t^2 \\ q_0\,(1-t^2) + 2q_1\,(1-t)\,t + q_2\,t^2 \end{pmatrix}$$

Applying an exact implicitization process for the above parametric Bézier segment, one get, the implicit form of this segment is equal to:

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

where:

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

This results in a sequence of quadratic bivariate polynomials, one for each Bézier segment of the spline curve $\mathbf{g}^{**}$.

3.2 Segments Joining

In order to generate a continuous function ($m = 0$) or C^1 function ($m = 1$), we join the bivariate polynomials, which have been produced by the implicitization process, along suitable transversal lines.

The choice of the transversal lines will be depending on the required function (if it is continuous or C^1). In the C^0 case, the transversal line is chosen as the line passing through the junction point $\mathbf{p}_0$ and one of the two other intersection points $\mathbf{p}_1$, $\mathbf{p}_1'$. In the C^1 case, we can still choose the transversal line as in C^0 or just choose it as an *arbitrary* line passing through the junction point $\mathbf{p}_0$.

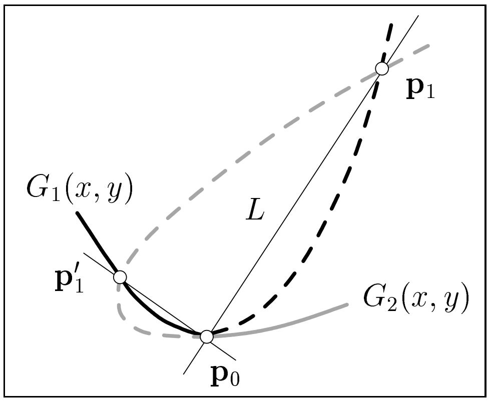

Figure 3.3 Choosing of the transversal line

In the next two chapters, we give a detailed discussion about the segments joining process.

Chapter 4

C^0 spline implicitization

After segment-wise implicitization process, we have for each segment an implicit form. We are interested in joining these segments such that it generate a continuous function. In order to achieve that:

- *First, we have to find out the transversal lines and study their behavior;*
- *Then, study how these segments can be joined with C^0 continuity along these transversal lines.*

This chapter consists of three sections. In the first one, we show how the transversal lines can be chosen. Then, in the second section, we study the asymptotic behavior of these transversal lines. In the third section, we show how curves can be parametrized numerically by arc length. Finally, in the fourth section, we give some solved numerical examples. Some results of the this chapter has been published in [19]

4.1 Defining a continuous function

Consider two neighboring Bézier segments of $\mathbf{g}^{**}$, with implicit representation $G_i(x, y)$, i=1, 2. These segments are parabolas which meet with tangent continuity at their junction point $\mathbf{p}_0$. Moreover, they intersect in two additional points $\mathbf{p}_1, \mathbf{p'}_1$, see Fig. 4.1. These two points can be real, conjugate complex, or even at infinity. Obviously, the transversal line has to be chosen as the line passing through the junction point $\mathbf{p}_0$ and one of the other two intersection points $\mathbf{p}_1, \mathbf{p'}_1$. If these points are real, then there are two possibilities to choose this line. We pick the line $L(\mathbf{p}_0, \mathbf{p}_1)$ which is closer to the normal vector of the curve. According to the following Lemma, we can always achieve a C^0 join along the transversal line $L(\mathbf{p}_0, \mathbf{p}_1)$.

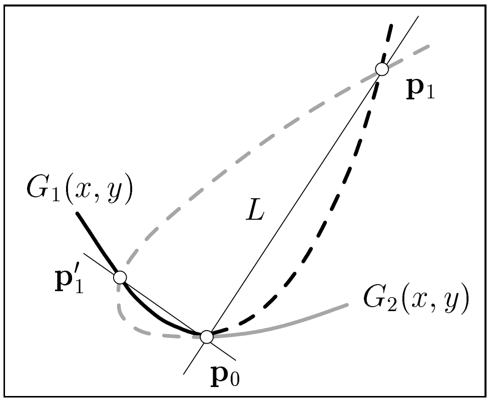

Figure 4.1 Choosing of the transversal line

LEMMA 1. *Given two quadratic functions $G_1(x, y)$ and $G_2(x, y)$ such that they have a common root and a parallel gradient at $\mathbf{p}_0(x_0, y_0)$, and intersect at $\mathbf{p}_1(x_1, y_1)$. Let $L(\mathbf{p}_0, \mathbf{p}_1)$ be the line joining $\mathbf{p}_0(x_0, y_0)$ and $\mathbf{p}_1(x_1, y_1)$. Then after multiplying G_2 by a suitable constant, $G_1(x, y)$ and $G_2(x, y)$ are identical along the line $L(\mathbf{p}_0, \mathbf{p}_1)$.*

Proof:

The restriction of $G_1(x, y)$ and $G_2(x, y)$ to the line $L(\mathbf{p}_0, \mathbf{p}_1)$ are two quadratic functions with common roots at $\mathbf{p}_0(x_0, y_0)$ and $\mathbf{p}_1(x_1, y_1)$. After multiplying G_2 by suitable constant, $G_1(x, y)$ and $G_2(x, y)$ are identical along the line $L(\mathbf{p}_0, \mathbf{p}_1)$. $\square$

After multiplying the implicitized segments with suitable constants, the collection of these bivariate polynomials – each restricted to a tile bounded by the transversal lines as described in Lemma 1 – forms a continuous function. This function is defined within a certain neighborhood of the curve[1].

4.2 Asymptotic Behavior of Transversal Lines

Lemma 1 can be applied if the transversal lines exist. Here, we will investigate the asymptotic behavior of transversal lines for decreasing step size (= segment length) h. For the sake of simplicity, we assume that the original curve is given by an arc length parameterization. It is well known that no polynomial and rational curves – except straight lines – can be equipped

[1]Clearly, it can be extended continuously beyond this neighborhood, using additional polynomial segments.

with a closed–form arc length parameterization [34]. However, one can always reparameterize a general parametric curve by its arc length approximately, using numerical methods.

In order to get results which can be interpreted geometrically, we use the so–called *canonical expansion* of the curve from differential geometry.

4.2.1 Canonical Taylor expansion

Consider a planar parametric curve. Let $\mathbf{g}(s) = \{\, g_1(s), g_2(s) \,\}$ be the parametric representation of the curve with respect to the arc length s; it is characterized by $\|\mathbf{g}'(s)\| \equiv 1$. Differentiation gives the unit tangent

$$\mathbf{g}'(s) = \vec{t}(s). \tag{4.1}$$

Its derivatives are governed by the Frenet–Serret formulas

$$\vec{t}'(s) = \kappa(s)\, \vec{n}(s), \quad \text{and} \quad \vec{n}'(s) = -\kappa(s)\, \vec{t}(s), \tag{4.2}$$

where $\vec{n}(s)$ and $\kappa(s)$ are the unit normal and the curvature at $\bar{\mathbf{g}}(s)$. At each point $\bar{\mathbf{g}}(s)$, unit tangent and unit normal form a positively oriented base of $\mathbb{R}^2$. Consequently, points on curves with positive curvature travel anti-clockwise for increasing arc–length parameter s. Starting from the Taylor expansion of $\bar{\mathbf{g}}(s)$,

$$\mathbf{g}(s) = \mathbf{g}(0) + \mathbf{g}'(0)\, s + \tfrac{1}{2}\, \mathbf{g}''(0)\, s^2 + \tfrac{1}{6}\, \mathbf{g}'''(0)\, s^3 + \mathcal{O}(s^4), \tag{4.3}$$

and with the help of the Frenet–Serret formulas, one may successively replace the derivatives of $\mathbf{g}(s)$ at $s = 0$ with linear certain combinations of $\vec{t}(0)$ and $\vec{n}(0)$. Expressions for the higher order derivatives of $\mathbf{g}$ are obtained by differentiating the Frenet–Serret formulas. Clearly, this generates expressions involving the derivatives of the curvature $\kappa(s)$.

Combining these formulas one gets the so–called canonical Taylor expansion with respect to the arc length parameterization, $\mathbf{g}(s) = \{\, g_1(s), g_2(s) \,\}$ with

$$\begin{aligned} g_1(s) &= s - \tfrac{1}{6}\, \kappa_0^2\, s^3 - \tfrac{1}{8}\, \kappa_0\, \kappa_1\, s^4 + \mathcal{O}(s^5) \qquad \text{and} \\ g_2(s) &= \tfrac{1}{2}\, \kappa_0\, s^2 + \tfrac{1}{6}\, \kappa_1\, s^3 + \tfrac{1}{24}\, (\kappa_2 - \kappa_0^3)\, s^4 + \mathcal{O}(s^5). \end{aligned} \tag{4.4}$$

Here, the system of coordinates has been chosen such that $\vec{t}(0) = \{1, 0\}^\top$ and $\vec{n}(0) = \{0, 1\}^\top$, with the origin at $\mathbf{g}(0)$. The coefficient κ_i is the i-th derivative of the curvature with respect to the arc length parameter at $s = 0$.

For more information the reader should consult any textbook on elementary differential geometry, e.g. [20, 27].

We will use the canonical Taylor expansion form of the curve to study the behavior of the transversal lines. We will classify the problem into two cases according to whether the curve has inflection point or not.

4.2.2 Case 1: The curve has no inflection point

We use *canonical Taylor expansion* of the curve from classical differential geometry to study the behavior of the transversal lines for decreasing step size (length of the segment tending to 0). According to the following theorem, if we consider a curve which is parameterized by its arc length, for sufficiently small step size h one of the intersection points $\mathbf{p}_1, \mathbf{p}'_1$ converges to the normal of the curve, i.e. transversal lines are always well behaved. It is well known that not all curves have an exact arc length parameterization. This is due to the fact, such curves do not admit closed-form expressions $s(u)$ for the function giving the arc length s in term of the parameter u. However, we can re-parameterize any general parametric curve by arc length *numerically*. This is what we are going to discuss in details in section 4.3.

THEOREM 1. *Consider a C^3 curve which is parameterized by its arc length, and which has no inflection point, $\kappa \neq 0$. We apply the process of spline implicitization (orthogonal projection in Sobolev space, implicitization, and joining of the segments) to the curve, where the knots of the quadratic B-Spline curve are uniformly spaced with stepsize h. If the stepsize h tends to zero, then the transversal line $L(\mathbf{p}_0, \mathbf{p}_1)$ gets closer and closer to the normal vector of the curve at the point $\mathbf{p}_0$.*

Proof

Start from a curve which is parameterized by its arc length. The canonical Taylor expansion with respect to the arc length parameterization, $\mathbf{g}(s) = (\, g_1(s), g_2(s)\,)$, is given as in equation (4.4).

By using quadratic defined over a knot sequence $[..., -2h, -h, 0, h, 2h, ...]$, we approximate the curve $\mathbf{g}(s)$ as in chapter 1. The quadratic B-spline coefficients are given as:

$$[\ \cdots,\ \begin{pmatrix} -\frac{3}{2}h+\frac{3}{8}\kappa_0^2h^3+\mathcal{O}(h^4) \\ \kappa_0h^2-\frac{3}{8}\kappa_1h^3+\mathcal{O}(h^4) \end{pmatrix},\ \begin{pmatrix} -\frac{1}{2}h-\frac{1}{24}\kappa_0^2h^3+\mathcal{O}(h^4) \\ \frac{1}{24}\kappa_1h^3+\mathcal{O}(h^4) \end{pmatrix},$$
$$\begin{pmatrix} \frac{1}{2}h+\frac{1}{24}\kappa_0^2h^3+\mathcal{O}(h^4) \\ -\frac{1}{24}\kappa_1h^3+\mathcal{O}(h^4) \end{pmatrix},\ \begin{pmatrix} \frac{3}{2}h-\frac{3}{8}\kappa_0^2h^3+\mathcal{O}(h^4) \\ \frac{3}{8}\kappa_1h^3+\mathcal{O}(h^4) \end{pmatrix},\ \cdots]$$

Figure 4.2 shows the distribution of coefficients $[..., \mathbf{d}_i, \mathbf{d}_{i+1}, \mathbf{d}_{i+2}, \mathbf{d}_{i+3}, ...]$ according to the first non vanishing term of the above coefficients.

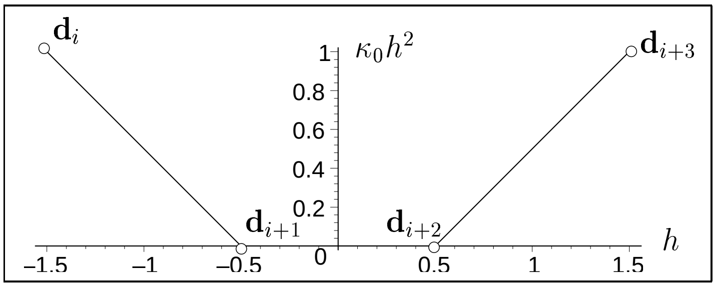

Figure 4.2 Coefficients distribution according to the first non vanishing term

The quadratic approximation $\mathbf{g}^*(s)$ of $\mathbf{g}(s)$ is given by piecewise function which consists of m segments ($\mathbf{g}_i^*, i = 1..m$). Picking up two consecutive segments, for example the segments defined over the intervals $[-h, 0]$ and $[0, h]$ respectively, and call them $\mathbf{g}_{\text{left}}^*$ and $\mathbf{g}_{\text{right}}^*$. Let $s = hS$, where S is the local parameter. Figure 4.3 shows the curve $\mathbf{g}(s)$ and it's B–spline approximation ($\mathbf{g}_i^*(hS), i = 1..m$). We showed only the part covered the knot interval $[-h, h]$.

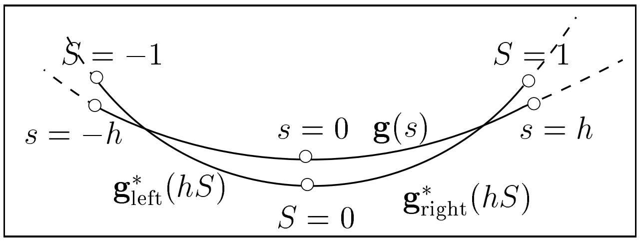

Figure 4.3 The curve $\mathbf{g}(s)$ and it's B–spline approximation $\mathbf{g}^*(s)$

The two parametric segments $\mathbf{g}_{\text{left}}^*(hS)$ and $\mathbf{g}_{\text{right}}^*(hS)$ are given as:

$$\mathbf{g}_{\text{left}}^*(hS) = \begin{pmatrix} (\frac{1}{4}\kappa_0^2h^3-\frac{1}{8}\kappa_0\kappa_1h^4)S^2+(h+\frac{1}{12}\kappa_0^2h^3)S+\frac{1}{80}\kappa_0\kappa_1h^4+\mathcal{O}(h^5) \\ (\frac{1}{2}\kappa_0h^2-\frac{1}{4}\kappa_1h^3-\frac{1}{24}(\kappa_0^3-\kappa_2)h^4)S^2-\frac{1}{12}\kappa_1h^3S+\frac{1}{240}(\kappa_0^3-\kappa_2)h^4+\mathcal{O}(h^5) \end{pmatrix}$$

$$\mathbf{g}^*_{\text{right}}(hS) = \begin{pmatrix} (\frac{-1}{4}\kappa_0^2h^3 - \frac{1}{8}\kappa_0\kappa_1h^4)S^2 + (h + \frac{1}{12}\kappa_0^2h^3)S + \frac{1}{80}\kappa_0\kappa_1h^4 + \mathcal{O}(h^5) \\ (\frac{1}{2}\kappa_0h^2 + \frac{1}{4}\kappa_1h^3 - \frac{1}{24}(\kappa_0^3 - \kappa_2)h^4)S^2 - \frac{1}{12}\kappa_1h^3S + \frac{1}{240}(\kappa_0^3 - \kappa_2)h^4 + \mathcal{O}(h^5) \end{pmatrix}$$

By checking the approximation $\|\mathbf{g}^*{}_\ell(hS) - \mathbf{g}(hS)\|$, where $\ell = \text{left, right}$, we get:

$$\|\mathbf{g}^*_{\text{left}}(hS) - \mathbf{g}(hS)\| = \begin{pmatrix} (\frac{1}{12}S + \frac{1}{4}S^2 + \frac{1}{6}S^3)\ \kappa_0^2\ h^3 \\ (\frac{-1}{12}S - \frac{1}{4}S^2 - \frac{1}{6}S^3)\ \kappa_1\ h^3 \end{pmatrix}$$

$$\|\mathbf{g}^*_{\text{right}}(hS) - \mathbf{g}(hS)\| = \begin{pmatrix} (\frac{1}{12}S - \frac{1}{4}S^2 + \frac{1}{6}S^3)\ \kappa_0^2\ h^3 \\ (\frac{-1}{12}S + \frac{1}{4}S^2 - \frac{1}{6}S^3)\ \kappa_1\ h^3 \end{pmatrix}$$

Hence, the approximation is $\mathcal{O}(h^3)$. This confirms the approximation order.

In order to compute the intersections points of the two segments $\mathbf{g}^*_{\text{left}}$ and $\mathbf{g}^*_{\text{right}}$, we have implicitized the parametric quadratic polynomial $\mathbf{g}^*_{\text{left}}$. Equation (4.5) gives the implicit form of $\mathbf{g}^*_{\text{left}}$.

$$\begin{aligned}
&(-h^3\kappa_1^2 + 2h^2\kappa_1\kappa_0)x \\
&+(-\kappa_1(\kappa_0^3 - \kappa_2)h^3 + (2\kappa_0(\kappa_0^3 - \kappa_2) - 3\kappa_1^2)h^2 + 12h\kappa_1\kappa_0 - 12\kappa_0^2)x^2 \\
&+(-\kappa_0(-\kappa_0\kappa_2 + \kappa_0^4 - 3\kappa_1^2)h^3 - 12h^2\kappa_1\kappa_0^2 + 12h\kappa_0^3)xy \\
&+(-h^3\kappa_0^2\kappa_1 + (2\kappa_0^3 + 2\kappa_2)h^2 - 12\kappa_1h + 24\kappa_0)y \\
&+(3h^3\kappa_1\kappa_0^3 - 3h^2\kappa_0^4)y^2 + \mathcal{O}(h^4)
\end{aligned} \tag{4.5}$$

Then by substituting the parametric form of $\mathbf{g}^*_{\text{right}}$ into equation 4.5 . This gives a quartic equation in the curve parameter S. But S^2 factors out, as both parabolic arcs are joined with C^1 continuity. The Laurent series of other two solutions S_1 and S_2 of the quartic equation are:

$$\begin{aligned}
S_1 &= -\frac{\kappa_1}{\kappa_0^3}h^{-1} + \frac{1}{4}\frac{\kappa_1^2}{\kappa_0^4} - \frac{1}{12}\frac{\kappa_1(3\kappa_1^2 - \kappa_0\kappa_2 + \kappa_0^4)}{\kappa_0^5}h \\
&+ \frac{1}{192}\frac{\kappa_1^2(39\kappa_1^2 + 12\kappa_0^4 - 16\kappa_0\kappa_2)}{\kappa_0^6}h^2 + \mathcal{O}(h^3)
\end{aligned}$$

$$\begin{aligned}
S_2 &= 4\frac{1}{\kappa_0^2}h^{-2} - \frac{\kappa_1}{\kappa_0^3}h^{-1} - \frac{1}{12}\frac{-8\kappa_0\kappa_2 + 4\kappa_0^4 + 15\kappa_1^2}{\kappa_0^4} + \frac{1}{4}\frac{\kappa_1(3\kappa_1^2 - \kappa_0\kappa_2 + \kappa_0^4)}{\kappa_0^5}h \\
&- \frac{1}{192}\frac{\kappa_1^2(39\kappa_1^2 + 12\kappa_0^4 - 16\kappa_0\kappa_2)}{\kappa_0^6}h^2 + \mathcal{O}(h^3)
\end{aligned}$$

By substituting the value of S_1 and S_2 into $\mathbf{g}^*_{\text{right}}$, we obtain two Laurent series for the additional intersection,

$$\mathbf{p}_1 = \begin{pmatrix} -\frac{\kappa_1}{\kappa_0^3} + \frac{2}{3}\frac{(3\kappa_1^2 - \kappa_0\kappa_2 + \kappa_0^4)}{\kappa_0^4}h + \mathcal{O}(h^2) \\ \frac{8}{\kappa_0^3}h^{-2} - \frac{1}{6}\frac{(39\kappa_1^2 + 12\kappa_0^4 - 20\kappa_0\kappa_2)}{\kappa_0^5} + \mathcal{O}(h) \end{pmatrix}$$
$$\mathbf{p}'_1 = \begin{pmatrix} -\frac{\kappa_1}{\kappa_0^3} - \frac{1}{12}\frac{\kappa_1(3\kappa_1^2 - \kappa_0\kappa_2 + 2\kappa_0^4)}{\kappa_0^5}h + \mathcal{O}(h^3) \\ \frac{1}{2}\frac{\kappa_1^2}{\kappa_0^5} + \frac{1}{96}\frac{\kappa_1^2(15\kappa_1^2 + 12\kappa_0^4 - 4\kappa_0\kappa_2)}{\kappa_0^7} + \mathcal{O}(h^3) \end{pmatrix} \tag{4.6}$$

Clearly, if the curve has no inflection at $\mathbf{p}_0$ (i.e., $\kappa_0 \neq 0$), then the first intersection point $\mathbf{p}_1$ converges for decreasing stepsize, $h \to 0$, to the infinite point of the normal at $\mathbf{p}_0$. The second intersection $\mathbf{p}'_1$ tends to a fixed limit position, which is fully determined by the curvature and its derivative with respect to arc length at $\mathbf{p}_0$. $\square$

The above proof is a Maple based proof. A Maple worksheet of the proof can be downloaded from www.risc.uni-linz.ac.at/people/shalaby/phdthesis. We have also included a printed version in Appendix A.

We illustrate this result by an example (see Fig. 4.4), showing the curve segments (gray) and the possible transversal lines (black lines, solid and dashed) for three stepsizes. The black transversal line converges to the curve normal.

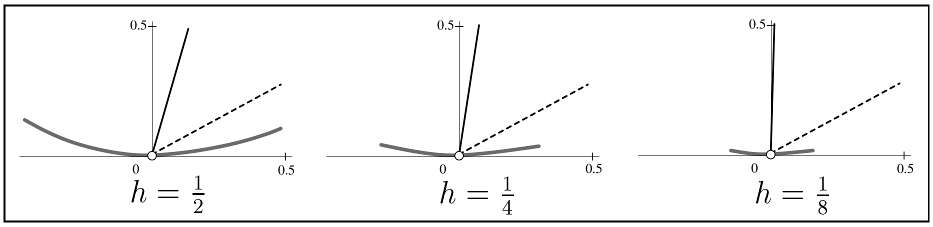

Figure 4.4 The behavior of the transversal lines for decreasing stepsize

According to this result, the system of transversal lines behaves nicely, provided that the curve has no inflections, is parameterized by an arc length parameter, and is approximated with sufficiently small stepsize (without knot removal). Under these assumptions, the two additional intersections of the parabolas are real, and one of them can be brought arbitrarily close to the normal of the curve. In the limit, for stepsize (segment length) $h \to 0$, these transversal lines envelop the *evolute* of the given curve, see [20]. Thus, for sufficiently small stepsize h, we get a system of transversal lines through the junction points of neighboring segments, such that the implicit equations can be joined (at least) continuously along them.

Thus, if the step size h is sufficiently small, then we can always have a system of lines through the junction points such that the implicit equations can be joined *at least* with C^0 continuity along the transversal line.

According to our numerical experiments, the arc length parameterization is not really needed in practice, and also knot removal can be used, in order to reduce the number of segments. However, inflections of the given curve cause a problem, which has to be dealt with.

4.2.3 Singular Case: Inflection points

The above construction fails at inflections of the quadratic spline curve, as the additional intersection points $\mathbf{p}_1$, $\mathbf{p}'_1$ do not exist in this case. In this situation, the two implicitized segments can still be joined continuously, provided that they can be seen as a graph of a univariate quadratic spline *function* with respect to a suitable coordinate system; the parallels of the y-axis then form a system of suitable transversal lines. That is, the axes of both parabolas have to be parallel; the two parabolas then share the infinite point of the y-axis.

Generally, however, the axes of the two parabolic segments are not parallel. As an example, Fig. 4.5 shows an inflected curve consisting of two parabolic segments G_1, G_2 and the control polygon. The control points $(\mathbf{d}_{-1}, \ldots, \mathbf{d}_2)$ are shown as diamonds, and the additional Bézier control points $(\mathbf{b}_{-2}, \ldots, \mathbf{b}_2)$ are shown as circles.

The axes of the parabolas are parallel if and only if there exists an auxiliary line $\overline{L}$, such that the orthogonal projections of $\mathbf{d}_0$ and $\mathbf{d}_1$ are the midpoints of the projected line segments $\overline{\mathbf{b}_{-2}\mathbf{b}_0}$ and $\overline{\mathbf{b}_0\mathbf{b}_2}$. We choose the line $\overline{L}$ parallel to $\overline{\mathbf{b}_{-2}\mathbf{b}_2}$. In order to obtain parabolas with parallel axes, we adjust the location of the control points $\mathbf{d}_0$ and $\mathbf{d}_1$ on the lines $\overline{\mathbf{d}_{-1}\mathbf{d}_0}$, $\overline{\mathbf{d}_1\mathbf{d}_2}$, leading to modified points $\mathbf{d}'_0$ and $\mathbf{d}'_1$. The new locations can be found by a simple (short) calculation.

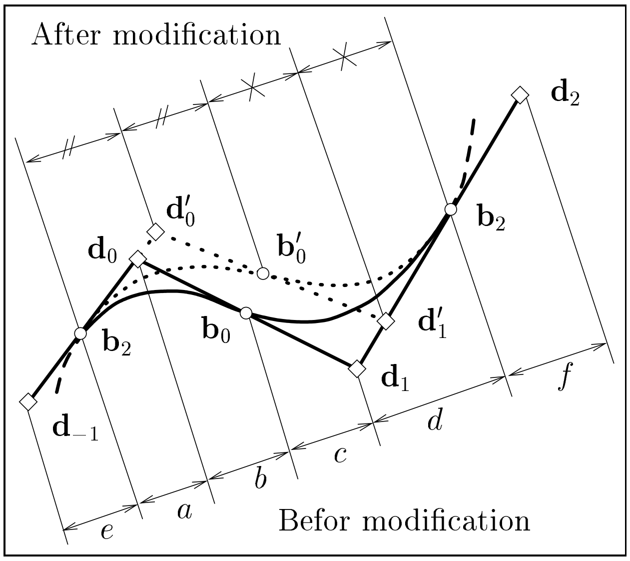

Figure 4.5 Modification of the control points at an inflection

Recall that, the control points $\mathbf{b}_{-2}$, $\mathbf{b}_0$ and $\mathbf{b}_2$ come from the knot insertion process. Then, each one of these control points can be written as:

$$\begin{aligned} \mathbf{b}_{-2} &= (1-\alpha_1)\ \mathbf{d}_{-1} + \alpha_1\ \mathbf{d}_0 \\ \mathbf{b}_0 &= (1-\alpha_2)\ \mathbf{d}_0 + \alpha_2\ \mathbf{d}_1 \\ \mathbf{b}_2 &= (1-\alpha_3)\ \mathbf{d}_1 + \alpha_3\ \mathbf{d}_2 \end{aligned}$$

where α_i is a constant which depends on the knot sequence. It is clear from Fig. 4.5 that:

$$e + a + b + c + d + f = \overline{\mathbf{d}_2\mathbf{d}_{-1}} \tag{4.7}$$

and

$$e = \frac{\alpha_1}{(1-\alpha_1)}\ a, \quad c = \frac{(1-\alpha_2)}{\alpha_2}\ b, \quad f = \frac{(1-\alpha_3)}{\alpha_3}\ d$$

Substituting the above into equation 4.7 and letting $a = b$, $c = d$, one get the new control polygon $\mathbf{d}_{-1}\,\mathbf{b}_{-2}\,\mathbf{d}'_0\,\mathbf{b}'_0\,\mathbf{d}'_1\,\mathbf{b}_2\,\mathbf{d}_2$, see Fig. 4.5.

4.3 Arc Length Parameterization

In subsection 4.2.2, we assumed that the parametric curve is parametrized by its arc length. It is well known that not all curves have an exact arc length parameterization. This is due to the fact, such curves do not admit closed-form expressions $s(u)$ for the function giving the arc length s in term of the parameter u. In [34, 14], an expression for the cumulative arc length function $s(u)$ for Bézier conics, amenable to accurate and efficient evaluation, is described exactly in terms of incomplete elliptic integrals of the first and second kind. However, we can re-parameterize any general parametric curve by arc length *numerically* . This can be done as follows:

- Start with a curve $\mathbf{g}$ in general parameterization u.
- Choose a number of points on $\mathbf{g}$, i.e. choose a number of parameter values u_i.
- Compute the arc length s_i at these points.
- Draw the points (u_i, s_i) in the (u, s) plane, see Fig. 4.6, where s_i and u_i are the arc length and the parameter value at the *i-th* point.
- Interpolate these points by piecewise linear function $\hat{\mathbf{g}}(s)$.
- For each segment substitute $u = \hat{\mathbf{g}}_i(s)$ into the components of $\mathbf{g}$, where $\hat{\mathbf{g}}_i$ is the *i-th* piece of $\hat{\mathbf{g}}$.
- The resulting curve $\bar{\mathbf{g}}$ is a piecewise curve of the same degree of $\mathbf{g}$.

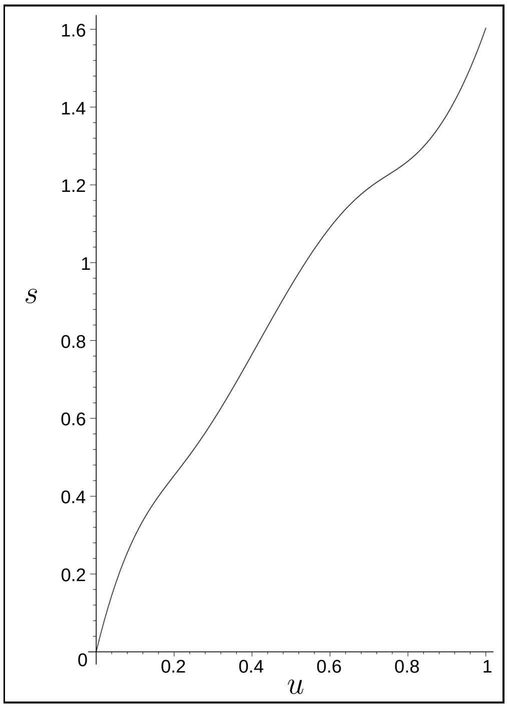

Figure 4.6 Approximate arc length parameterization

EXAMPLE

Given: A parametric curve **g**

$$X := 2 - u - 2u^2 + 5u^3 - 3u^4, \qquad Y := 1 + 4u - 14u^2 + 14u^3 - 4u^4$$

Find: An approximate arc length parameterization of **g**.

We choose 100 points on **g** , one on each end and 98 intermediate points, the arc length at these points are computed, the piecewise linear function g is computed and the curve is re-parametrized by it's arc length. Figure 4.7 shows the distribution of points along the curve before (to the left) and after (to the right) the re-parameterization by arc length. From the figure, it is clear that the distribution of points is uniform after the re-parameterization by arc length.

Figure 4.8 shows porcupine diagram of the parametric speed (plot the parametric speed as a scale normal to the curve) before (to the left) and after (to the right) the re-parameterization by arc length. From the figure, it is clear that the parametric speed is uniform after the re-parameterization

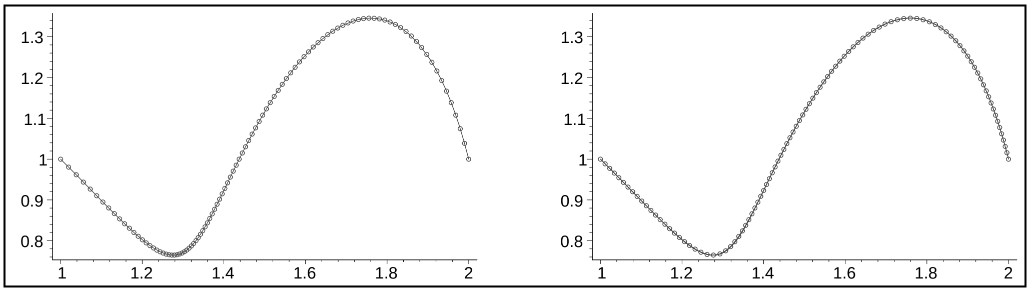

Figure 4.7 Distribution of parametric speed along the curve

by arc length.

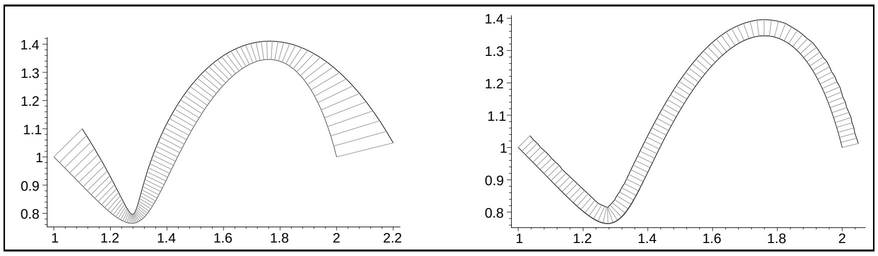

Figure 4.8 Porcupine diagram of the parametric speed

4.4 Numerical Examples Continued

EXAMPLE 1 Continued

Using the C^0 spline implicitization technique, we derived a piecewise quadratic function G, whose zero contour $G(x, y) = 0$ is the quadratic spline curve (see Fig. 2.5, the case $\mathcal{T} = 10^{-3}$). In order to visualize the quality of the implicitization, Fig. 4.9 shows the algebraic parallel curves or *algebraic offsets* (thin lines), $G(x, y) = c$ for certain constants (algebraic distances) c, and the transversal lines (which define the polynomial pieces of G) through the junction point. In order to make the picture clearer, we enlarged a part of the curve (see Fig. 4.10). Generally, the algebraic offsets (which consist of parabolic arcs) are not tangent continuous.

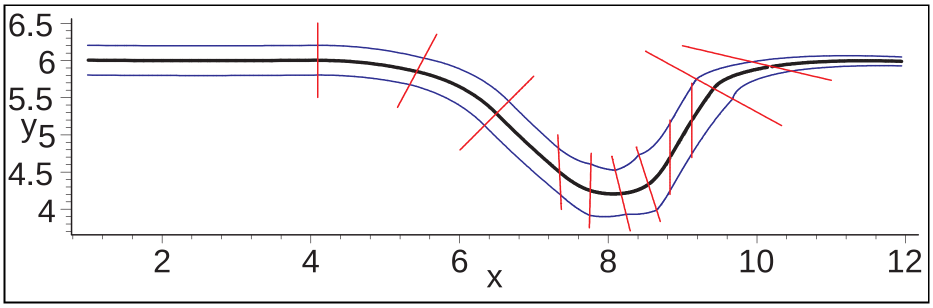

Figure 4.9 C^0 implicitized curve (thick) and its algebraic offsets (thin)

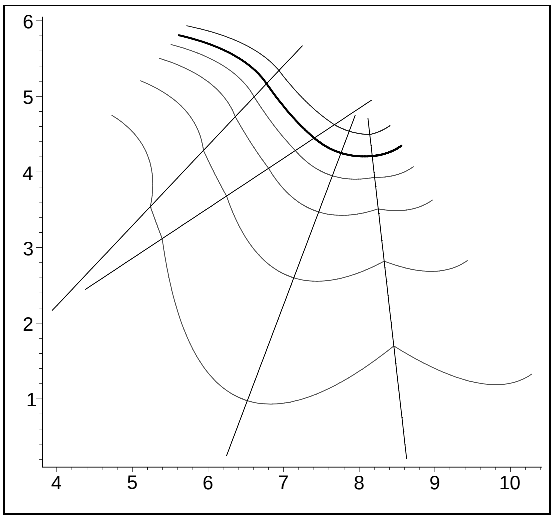

Figure 4.10 C^0 implicitized curve and its algebraic offsets (enlarged)

EXAMPLE 2 Continued

Using the C^0 spline implicitization technique, we derived a piecewise quadratic function G, whose zero contour $G(x, y) = 0$ is the quadratic spline curve (see Fig. 2.6, the case $\mathcal{T} = 10^{-3}$). In order to visualize the quality of the implicitization, Fig. 4.11 shows the algebraic parallel curves or *algebraic offsets* (thin lines), $G(x, y) = c$ for certain constants (algebraic distances) c, and the transversal lines (which define the polynomial pieces of G) through the junction point. In order to make the picture clearer, we enlarged a part of the curve (see Fig. 4.12). Generally, the algebraic offsets (which consist of parabolic arcs) are not tangent continuous.

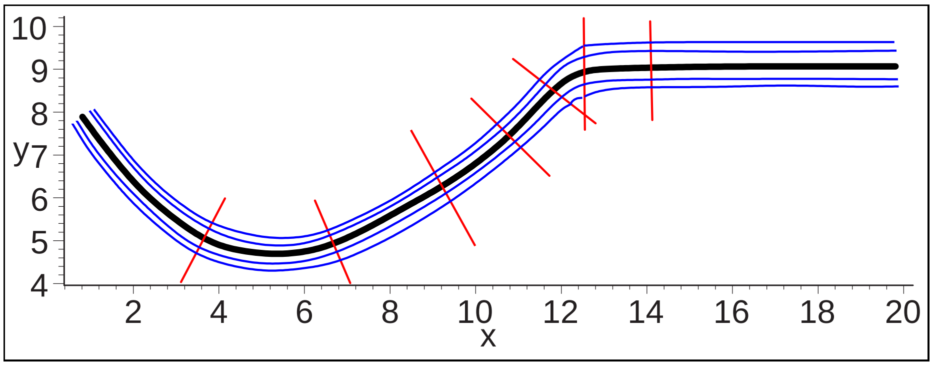

Figure 4.11 C^0 implicitized curve (thick) and its algebraic offsets (thin)

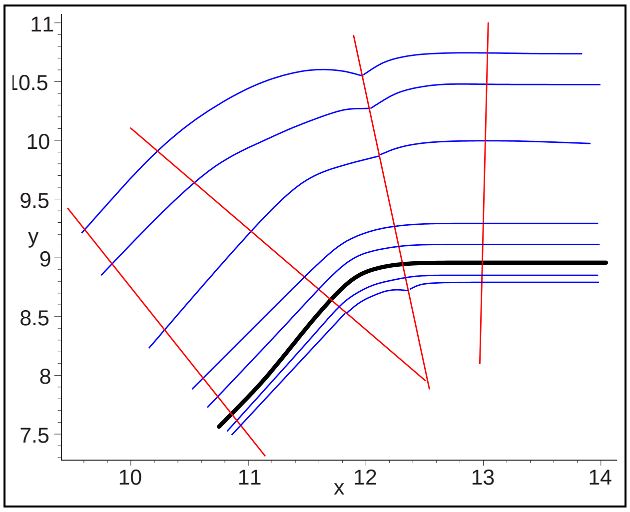

Figure 4.12 C^0 implicitized curve and its algebraic offsets (enlarged)

EXAMPLE 3 Continued

Using the C^0 spline implicitization technique, we derived a piecewise quadratic function G, whose zero contour $G(x, y) = 0$ is the quadratic spline curve (see Fig. 2.7, the case $\mathcal{T} = 10^{-3}$). In order to visualize the quality of the implicitization, Fig. 4.13 shows the algebraic parallel curves or *algebraic offsets* (thin lines), $G(x, y) = c$ for certain constants (algebraic distances) c, and the transversal lines (which define the polynomial pieces of G) through the junction point. In order to make the picture clearer, we enlarged a part of the curve (see Fig. 4.14). Generally, the algebraic offsets (which consist of parabolic arcs) are not tangent continuous.

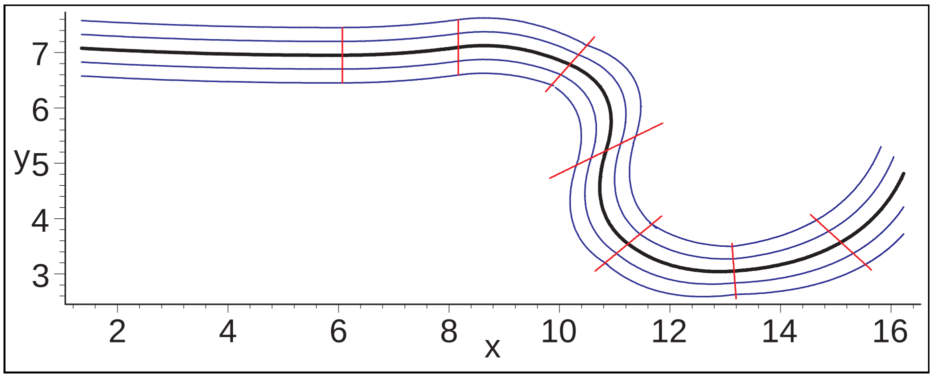

Figure 4.13 C^0 implicitized curve (thick) and its algebraic offsets (thin)

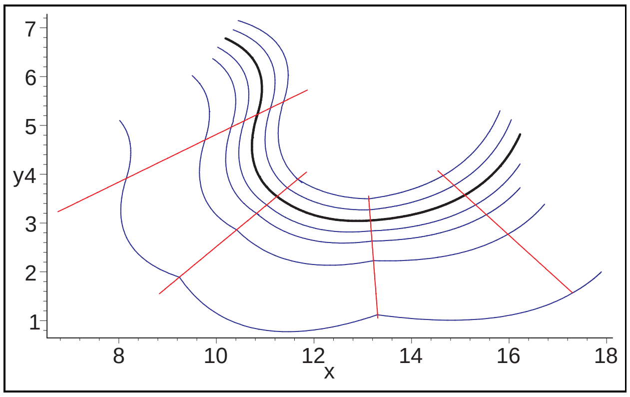

Figure 4.14 C^0 implicitized curve and its algebraic offsets (enlarged)

Chapter 5

C^1 Spline Implicitization

Differentiability (C^1) is needed for many applications, e.g., for footpoint generation[a] [1]. As the computation of the gradient is needed, the implicit curve should be C^1. Other applications are the computation of distance bound between two planar curves [18], and the computation of the intersection point between two curves [13].

In order to generate a C^1 function, we modify the bivariate polynomials, which have been produced by the implicitization process in chapter 3, by multiplying them with suitable polynomial factors.

In this chapter, we propose two approaches to achieve C^1 joined along the transversal lines. In the first approach we propose multiplying by suitable linear polynomial factors. In this approach, the degree of the global implicit representation depend on the number of segments. Also, the linear factors may introduce singularities in the area of interest.

In the second approach, which solves the problems that arise in the first approach, we multiply by quadratic polynomial factors. In this case, we obtained a globally C^1 bivariate quartic function. We shall use the following abbreviation:

DEFINITION 3. *For a given bivariate function $F(x, y)$, let $\mathcal{Z}(F)$ be the zero set $\{(x, y) \mid F(x, y) = 0\}$.*

Some results of the this chapter has been published in [42]. Another publication is on preparation.

[a] The footpoint of $\mathbf{p}$ on a curve $F(\mathbf{x}) = 0$ is a point $\mathbf{X}$ such that $[(\mathbf{p} - \mathbf{X}), \nabla F(\mathbf{X})] = 0$.

5.1 Approach 1: Multiplying by suitable linear polynomial factors

In this approach we identify the transversal lines similar to the case of C^0, as the line pass through the junction point $\mathbf{p}_0$ and one of the intersection points $\mathbf{p}_1$. To achieve C^1 joining along the transversal line $L(\mathbf{p}_0, \mathbf{p}_1)$, we multiply the implicitized segments by suitable linear polynomial factors.

5.1.1 Joining two segments

Consider two neighboring Bézier segments of $\mathbf{g}^{**}$, with implicit representations $G_i(x, y)$, $i = 1, 2$. We choose the transversal line L as the line passing through $\mathbf{p_0}$ and $\mathbf{p}_1$ (see Fig. 5.1). We multiply $G_1(x, y)$ and $G_2(x, y)$ by linear polynomial factors $\lambda(x, y)$ and $\eta(x, y)$ respectively such that:

(I) $\mathcal{Z}(\eta)$ passes through $\mathbf{p}_1$ and tangent to $\mathcal{Z}(G_1)$.

(II) $\mathcal{Z}(\lambda)$ passes through $\mathbf{p}_1$ and tangent to $\mathcal{Z}(G_2)$.

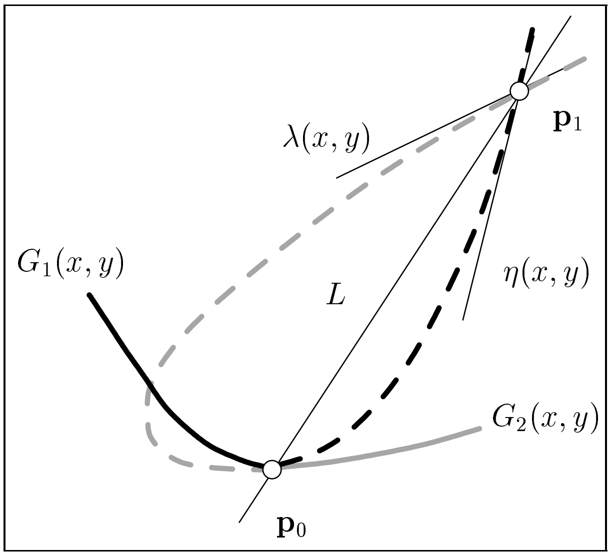

Figure 5.1 Multiply $G_1(x, y)$ and $G_2(x, y)$ by linear factors $\lambda(x, y)$ and $\eta(x, y)$ respectively

We consider the bivariate polynomials

$$F_1(x, y) = G_1(x, y)\,\lambda(x, y) \qquad \text{and} \qquad F_2(x, y) = G_2(x, y)\,\eta(x, y).$$

THEOREM 2. *If the conditions (I), (II) are satisfied, then, after multiplying F_2 by suitable constant, the two functions $F_1(x, y)$ and $F_2(x, y)$ meet with C^1 continuity along the line $L(\mathbf{p}_0, \mathbf{p}_1)$.*

Proof. Let:

$$F_1(x, y) = \sum_{\substack{i+j\leq 3\\ i,j\geq 0}} A_{i,j}\, x^i\, y^j, \qquad F_2(x, y) = \sum_{\substack{i+j\leq 3\\ i,j\geq 0}} B_{i,j}\, x^i\, y^j,$$

By normalization and simple change of coordinates, we can achieve that $\mathbf{p}_0 = (0, 1)$ and $\mathbf{p}_1 = (0, 0)$, i.e $L(\mathbf{p}_0, \mathbf{p}_1)$ is the y–axis. The zero contours of both polynomials pass through $\mathbf{p}_0$ and $\mathbf{p}_1$, and the tangents of the zero contours at these points are identical. Further, $\mathbf{p}_1$ is a double point of both zero contour.

To show that $F_1(x, y)$ and $F_2(x, y)$ are joined with C^1 along $L(\mathbf{p}_0, \mathbf{p}_1)$, we have to show that $A_{i,j} = B_{i,j}$ whenever $i \leq 1$.

As $\mathbf{p}_1$ is a double point, we have $A_{0,0} = A_{1,0} = A_{0,1} = B_{0,0} = B_{1,0} = B_{0,1} = 0$. As the tangents of the zero contours at $\mathbf{p}_1$ are identical, the Hessians at $(0, 0)$ are linearly dependent. Hence, after a suitable re-normalization $A_{2,0} = B_{2,0}$, $A_{1,1} = B_{1,1}$, $A_{0,2} = B_{0,2}$.

The zero contours of both polynomials pass through p, therefore $A_{0,3} = B_{0,3} = -A_{0,2} = -B_{0,2}$. Still to be shown: $A_{1,2} = B_{1,2}$.

The two vectors

$$\begin{aligned} \nabla\ F_1(0, 1) &= (A_{1,1} + A_{1,2}, 2A_{0,2} + 3A_{0,3}) \\ \nabla\ F_2(0, 1) &= (B_{1,1} + B_{1,2}, 2B_{0,2} + 3B_{0,3}) \end{aligned}$$

are linearly dependent.

Case 1: $2A_{0,2} + 3A_{0,3} = 2B_{0,2} + 3B_{0,3} = 0$. As $A_{0,2} = -A_{0,3}$, this implies $A_{0,2} = A_{0,3} = 0$, which is a degenerate case: both F_1 and F_1 degenerate to the transversal line L (y-axis) and another quadratic function (which is the case of multiplying both G_1 and G_2 by the transversal line L).

Case 2: If the second component of the gradients is not zero, then the linear dependency implies $A_{1,2} = B_{1,2}$.

Consequently, the two cubic polynomials are joined with C^1 continuity along the transversal line, and they even meet with C^2 continuity at $\mathbf{p}_1$. □

REMARK 3. *It is not necessary to have $\mathcal{Z}(\eta)$ and $\mathcal{Z}(\lambda)$ to be the tangents of $\mathcal{Z}(G_1)$ and $\mathcal{Z}(G_2)$ at $\mathbf{p}_1$ respectively. It is sufficient to have $\mathcal{Z}(\eta)$ and $\mathcal{Z}(\lambda)$ pass through $\mathbf{p}_1$ and the sum of the slopes of $\mathcal{Z}(\lambda)$ and T_1 is equal to the sum of the slopes of $\mathcal{Z}(\eta)$ and T_2, where T_1 and T_2 are the tangent of $\mathcal{Z}(G_1)$ and $\mathcal{Z}(G_2)$ at $\mathbf{p}_1$ respectively.*

5.1.2 Joining several segments

Figure 5.2 shows a piecewise curve consisting of three quadratic segments $G_1(x, y)$, $G_2(x, y)$ and $G_3(x, y)$. To achieve C^1 along L_1 we multiply by $T_1(x, y)$ in the first patch and by $T_2(x, y)$ in the second patch where $\mathcal{Z}(T_1)$, $\mathcal{Z}(T_2)$ are the tangents of $\mathcal{Z}(G_2)$, $\mathcal{Z}(G_1)$ at $\mathbf{p_1}$ respectively. Similarly, to achieve C^1 along L_2, we multiply by $T_3(x, y)$ in the second patch and by $T_4(x, y)$ in the third patch where $\mathcal{Z}(T_3)$, $\mathcal{Z}(T_4)$ are the tangents of $\mathcal{Z}(G_3)$, $\mathcal{Z}(G_2)$ at $\hat{\mathbf{p}}_\mathbf{1}$ respectively.

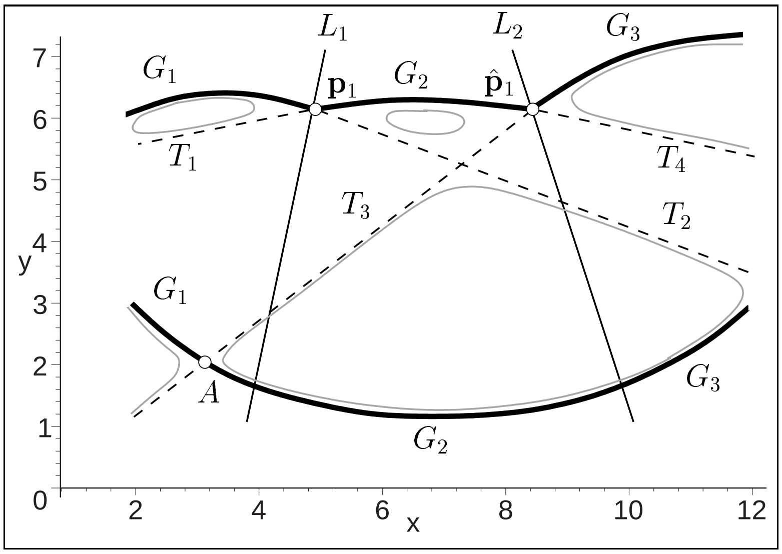

Figure 5.2 C^1 continuity based on multiplying by linear factors

To have a global C^1 construction, we have to extend T_3 into the first patch and also extend T_2 into the third patch. This will introduce unwanted branches and may introduce singularities in the area of interest (point A for example). Moreover, the degree of the global implicit representation will depend on the number of segments. This due to the fact, every linear factor multiplier will increase the degree by 1. The degree of the global implicit representation will be equal to $2+(m-1)$, where m is the number of patches.

The algebraic offset is plotted in gray. It is clear from the figure that the algebraic offset is C^1. The implicit curve has a global degree 4.

5.1.2.1 Singular Case: Inflection Points

The above construction fails at inflections of the quadratic spline curve, as the additional intersection points $\mathbf{p}_1$, $\mathbf{p'}_1$ do not exist in this case. They are either conjugate complex or at infinity. In this situation, the two implicitized segments can still be C^1–joint, provided that they can be seen as a graph of a univariate quadratic spline *function* with respect to a suitable coordinate system; the parallels of the y-axis then form a system of suitable transversal lines. That is, the axes of both parabolas have to be parallel; the two parabolas then share the infinite point of the y-axis.

Generally, however, the axes of the two parabolic segments are not parallel. To force them to be parallel, we modify the control points as in 4.2.3.

EXAMPLE 3 Continued

Figure 5.3 shows the implicitized curve of the quadratic B-spline curve from Figure 2.7 and its algebraic offsets after multiplying by linear factors. It is clear from the figure that the algebraic offsets are C^1. In this case, the linear factors did not introduce any singular points in the area of interest. This is due the fact that the second point of intersection $\mathbf{p}_1$ is always far away from the area of interest. Still the problem of the degree of the final implicit representation exists. In this example we have a degree 6 global implicit curve since we have 5 patches (bound by thick line). In the first patch the three parabolic segments have a parallel axis, parallel to the transversal lines (thin line), so we can consider these three segments as one patch. As they are already C^1 along these transversal lines. Similarly, in the second patch, the two parabolic segments have parallel axes (parallel to the transversal line through the inflection point).

To avoid the above problems, we propose another approach based on multiplying by quadratic polynomial factors. In this case, we will have a global C^1 continuity spline curve of degree 4.

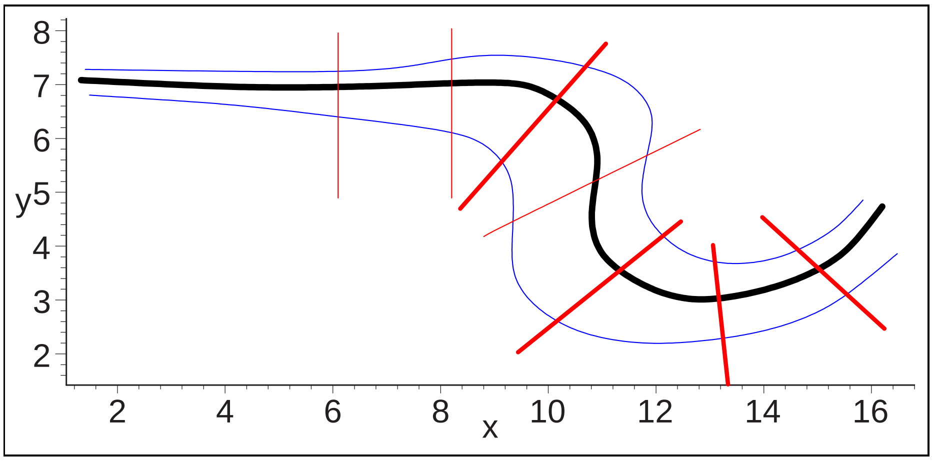

Figure 5.3 C^1 Implicitized curve and its algebraic offsets (C^1 achieved after multiplying by linear factors)

5.2 Approach 2: Multiplying by quadratic polynomial factors

In this approach we choose the transversal lines L as an arbitrary line passing through the junction point $\mathbf{p_0} = \mathbf{p}$, to be called $\mathbf{p}$ henceforth. In this case, we get additional degree of freedom. It should be noted that, this can be done also in the previous approach (approach 1) but this will not avoid the problems appeared above. We multiply the implicit segments by quadratic polynomial factors.

5.2.1 Joining two segments

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. 5.4). Clearly, this point is non–singular. In order to achieve a C^1–joint of G_1 and G_2:

- First, 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 other 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 $\mathbf{r}$.
- 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 $\mathbf{s}$.

- Consider the bivariate polynomials

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

Then according to the following theorem, the two bivariate polynomials $F_1(x, y)$ and $F_2(x, y)$ can always join with C^1 along the chosen transversal line L.

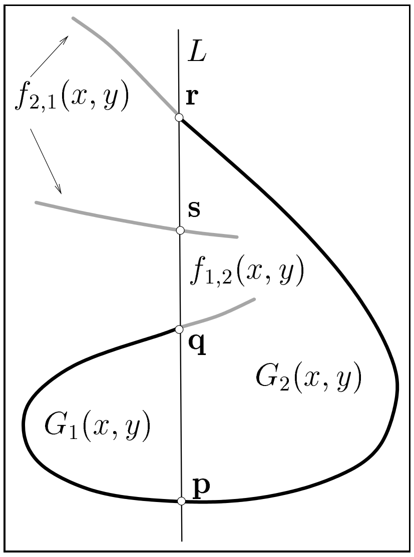

Figure 5.4 Multiplying by quadratic polynomial factors

THEOREM 3. *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 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 equations, 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 $i \le 1$. 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)). ☐

REMARK 4.

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 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 thesis, we assume that this assumption is always satisfied.*

4. *It should be noted that a* C^1 *joint along* L *can be achieved also if the polynomial factor vanishes along the transversal line but this case is not interesting for applications, due to the presence of singularities.*

5.2.2 Joining several segments

In order to join more than two segments, we use the above technique recursively. Consider three neighboring segments which are described by bivariate polynomials G_{i-1}, G_i and G_{i+1}, see Figure 5.5.

- 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. In order to obtain a *globally* C^1 joined spline function, we propose the following algorithm.

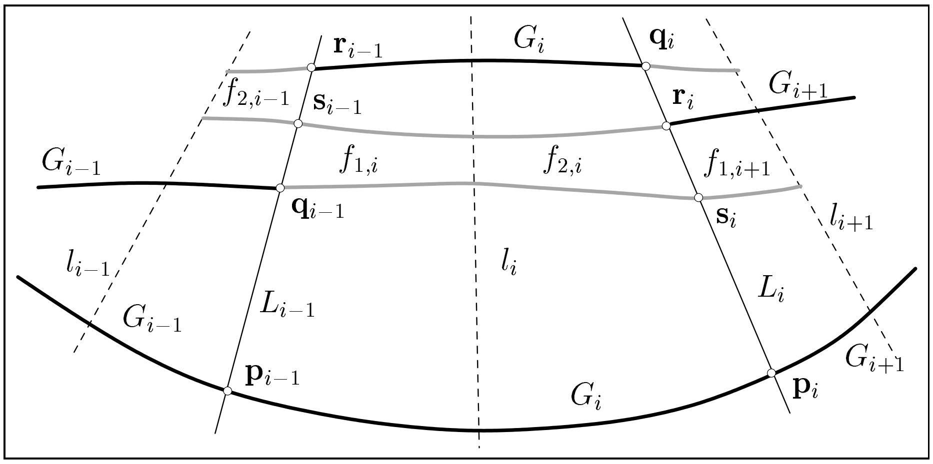

Figure 5.5 The original curve (black) and the quadratic polynomial factors (gray)

5.2.2.1 The algorithm

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 subpatches. 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.

It should be noted that, 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.

5.2.2.2 Solvability of the homogeneous system of step 4

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.

[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.

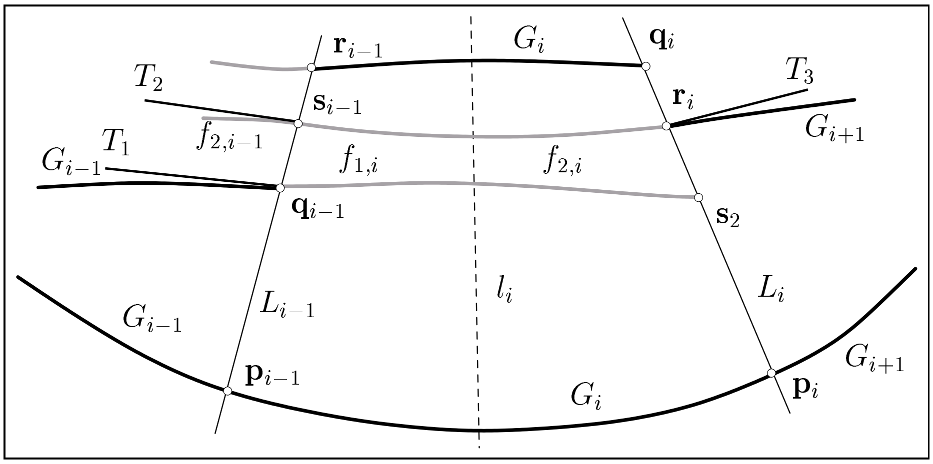

Figure 5.6 Step 4 of the construction.

LEMMA 2. *Consider the situation of step 4, see figure 5.6. In addition, we assume that the following assumptions are satisfied.*

(A) *The three points* $\mathbf{p_{i-1}}$, $\mathbf{q_{i-1}}$ *and* $\mathbf{s_{i-1}}$ *are distinct.*

(B) *The three points* $\mathbf{p_i}$, $\mathbf{r_i}$ *and* $\mathbf{s_i}$ *are distinct.*

(C) L_{i-1} *does not pass through* $\mathbf{r_i}$.

(D) L_i *does not pass through* $\mathbf{q_{i-1}}$ *or* $\mathbf{s_{i-1}}$.

(E) l_i *does not pass through* $\mathbf{r_i}$, $\mathbf{q_{i-1}}$ *or* $\mathbf{s_{i-1}}$.

(F) l_i *does not pass through the intersection points of* L_{i-1} *and* T_3, *where* T_3 *is the tangent of* $f_{2,i}$ *at* $\mathbf{r_i}$.

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

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 has tangent T_3 at it. 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

$$2\,\underbrace{y_1\,y_2\,(y_1 - y_2)}_{\mathcal{F}_1, \mathcal{F}_2, F_3}\;\underbrace{x_3}_{\mathcal{F}_4}\;\underbrace{(y_3 - y_0 - m_0(x_3 - x_0))}_{\mathcal{F}_5}\;\underbrace{(y_3 - m_3x_3 - y_0 + m_0x_0)}_{\mathcal{F}_6}$$

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 $\mathcal{F}_1, \mathcal{F}_2, \mathcal{F}_3$ vanish if the three points $\mathbf{p_{i-1}}$, $\mathbf{q_{i-1}}$ and $\mathbf{s_{i-1}}$ are not identical. The fourth factor $\mathcal{F}_4$ vanishes if L_{i-1} passes through $\mathbf{r_i}$, and the fifth factor $\mathcal{F}_5$ 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 $\mathcal{F}_6$ 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. $\square$

5.2.2.3 Bad patches

In the above 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 $\mathbf{s}_\ell$, $\ell = 2, \ldots, m-1$ where m is the number of the patches[2]. Hence, $\mathbf{s}_\ell$ may coincide with $\mathbf{p}_\ell$, $\mathbf{q}_\ell$ or $\mathbf{r}_\ell$, i.e the polynomial factors vanish on the transversal line. Moreover, the polynomial factors may intersect with the original curve at the area of interest.

As an example, Fig. 5.7 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 do such a local modification at bad patches.

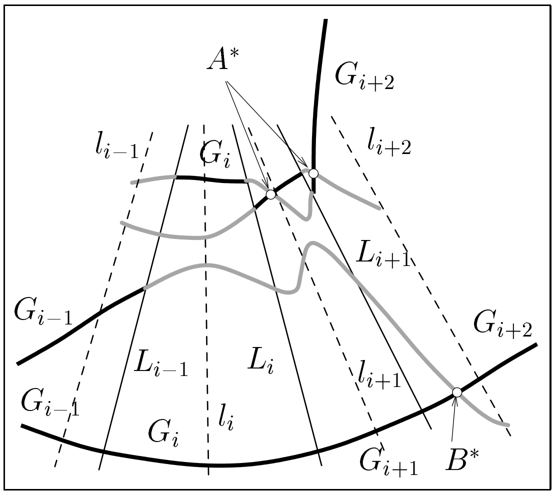

Figure 5.7 The quadratic factors intersect the original curve at A^*, B^*

5.2.3 Local modification at bad patches

If there is an intersection of the quadratic polynomial factors and the original curve, or $\mathbf{s}_\ell$ coincides with $\mathbf{p}_\ell$, $\mathbf{q}_\ell$ or $\mathbf{r}_\ell$, 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 to avoid 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. 5.8).

[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

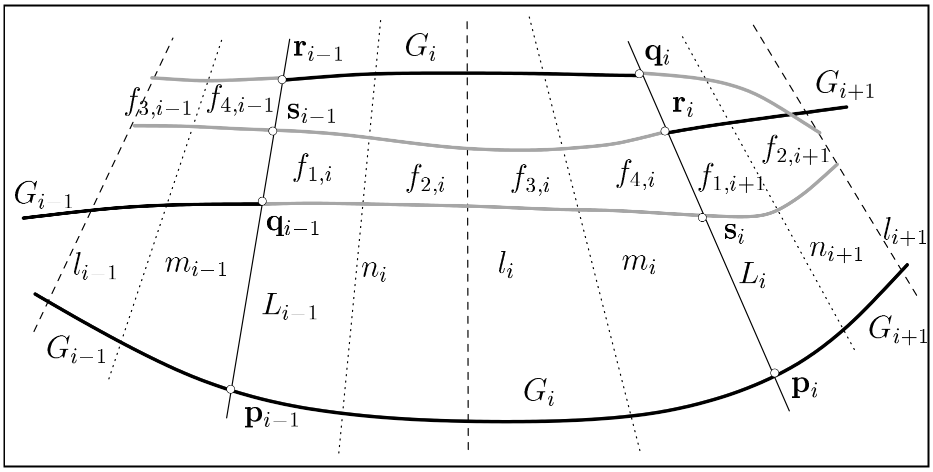

Figure 5.8 The original curve (black) and the modified quadratic factors (gray)

5.2.3.1 Modified Algorithm

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 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 homogeneous linear conditions each, so we gain two additional degrees of freedom. This results in a homogeneous system of equations, after choosing these two degree of freedoms, which provide 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.

Dividing into 4 sub-patches will introduce 2 additional degrees of freedom. This two additional degrees of freedom are used to control the polynomial factors locally. In the sequel, we use these additional degrees of freedom to choose the position of $\mathbf{s}_i$ and the slope of T_i. Fig. 5.9 shows the original curve (black) and the polynomial factors (gray) for different choices of the tangent T_i at $\mathbf{s_i}$.

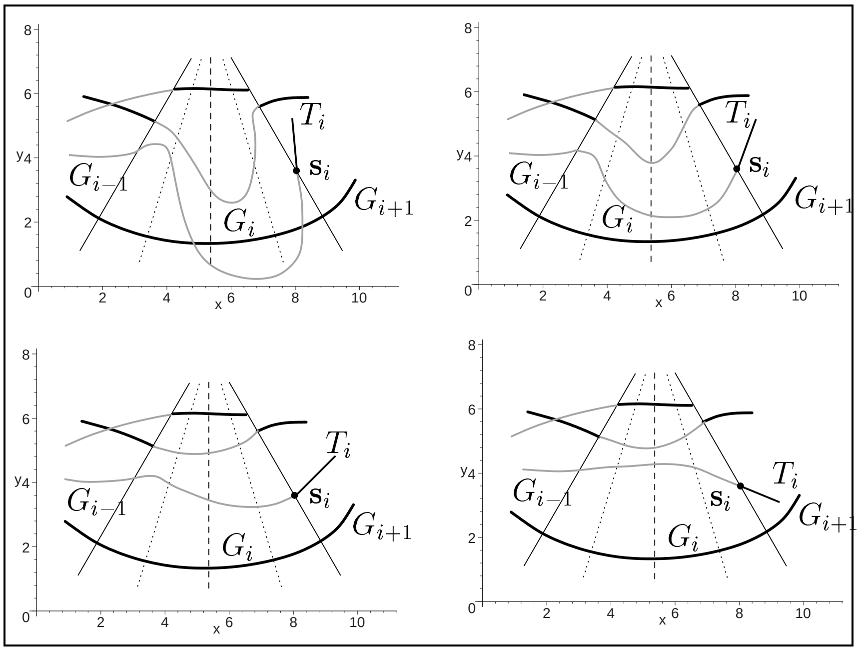

Figure 5.9 Using the slope of the tangent T_i to control the curve locally

If there 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 choose the coordinate of $\mathbf{s}_i$ and 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 them into four sub–patches.

Fig. 5.10 shows the same example of Fig. 5.7. 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.

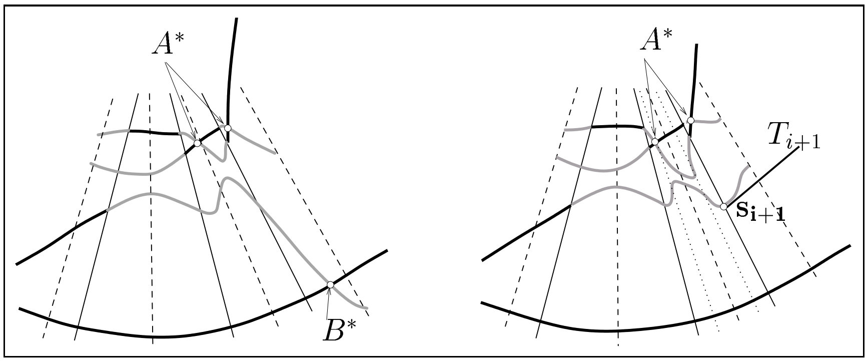

Figure 5.10 Using the two additional degree of freedom to avoid the intersection between the quadratic factors and the original curve.

REMARK 5. • If the curve has an inflection point, the standard construction works. Only we have to take care that each of polynomial multiplier does not intersect the original curve in the area of interest.

- At the inflection point, the two implicitized segments can be C^1–joined after a suitable modification of the control points, see Fig. 4.5. This fact can be used to *reduce* the number of segments. In Figure 5.11, we consider the two implicit segments G_2 and G_3 as *one* segment because they are already C^1 along L_2 (after a suitable modification of the control points). Consider the transversal line L_2 as the line dividing the patch bounded by L_1 and L_3. In this patch, we multiply by a piecewise quadratic multiplier on each of the two sub–patches such that it must be C^1 on the whole patch.
- In C^1 technique, the parametric speed is important for the distribution of the algebraic offsets. Figure 5.12 b shows the porcupine diagram of

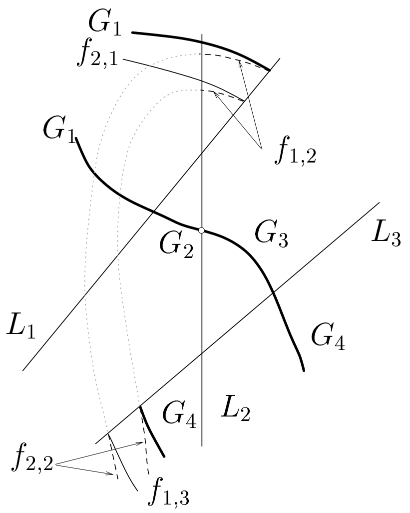

Figure 5.11 Choosing the quadratic factors at an inflection point

the parametric speed (plot the parametric speed as a scale normal to the curve) of the curve which we used throughout the paper. When we change the parametric speed, see Fig. 5.12 d, the distribution of the algebraic offsets is changed, see Fig. 5.12 c.

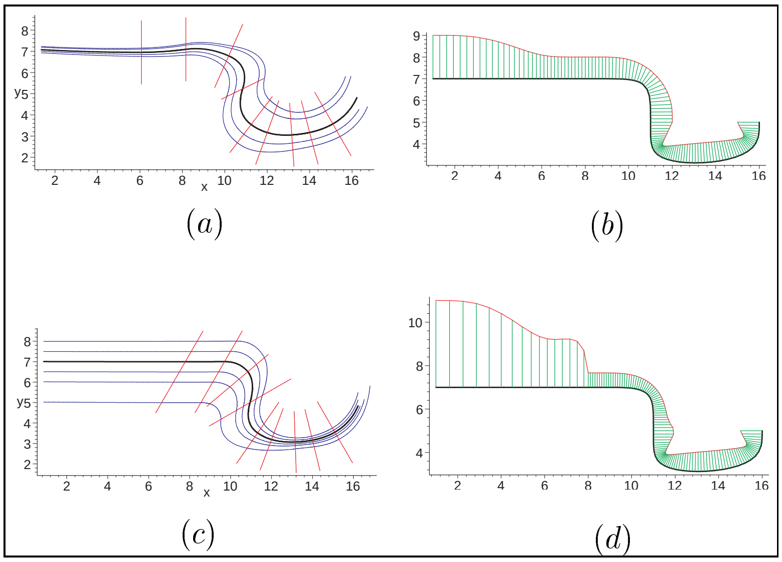

Figure 5.12 C^1 Implicitized curves and its algebraic offsets

5.3 Numerical Examples Continued

EXAMPLE 1 Continued

Using the C^1 spline implicitization technique, we derived a piecewise quadratic function G, whose zero contour $G(x, y) = 0$ is the quadratic spline curve (see Fig. 2.5, the case $\mathcal{T} = 10^{-3}$). In order to visualize the quality of the implicitization, Fig. 5.13 shows the algebraic parallel curves or *algebraic offsets* (thin lines), $G(x, y) = c$ for certain constants (algebraic distances) c, and the transversal lines (which defined the polynomial pieces of G) through the junction point. In order to make the picture clearer, we enlarged a part of the curve (see Fig. 5.14). Generally, the algebraic offsets (which consist of parabolic arcs) are not tangent continuous.

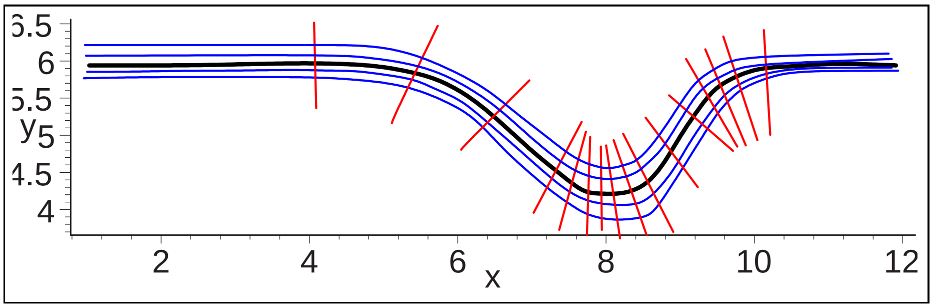

Figure 5.13 C^1 implicitized curve (thick) and its algebraic offsets (thin)

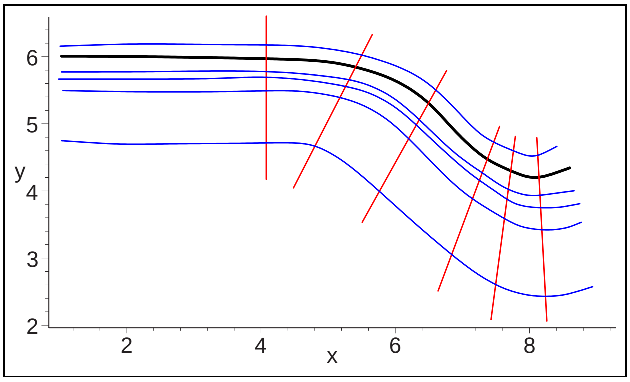

Figure 5.14 C^1 implicitized curve and its algebraic offsets (enlarged)

EXAMPLE 2 Continued

Using the C^1 spline implicitization technique, we derived a piecewise quadratic function G, whose zero contour $G(x, y) = 0$ is the quadratic spline curve (see Fig. 2.6, the case $\mathcal{T} = 10^{-3}$). In order to visualize the quality of the implicitization, Fig. 5.15 shows the algebraic parallel curves or *algebraic offsets* (thin lines), $G(x, y) = c$ for certain constants (algebraic distances) c, and the transversal lines (which defined the polynomial pieces of G) through the junction point. In order to make the picture clearer, we enlarged a part of the curve (see Fig. 5.16). Generally, the algebraic offsets (which consist of parabolic arcs) are not tangent continuous.

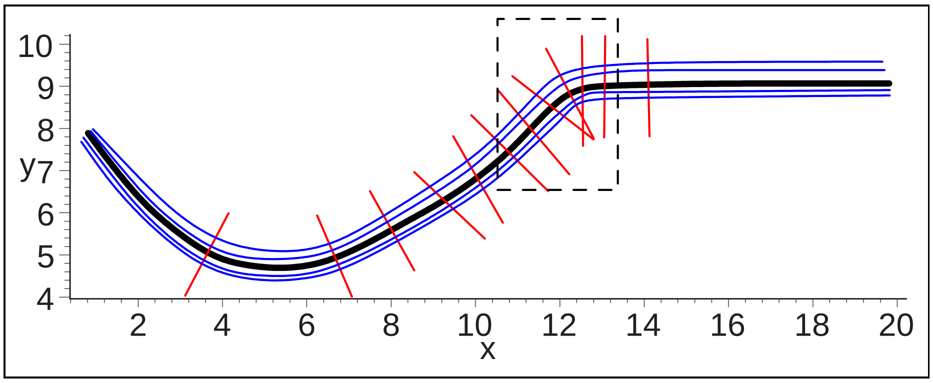

Figure 5.15 C^1 implicitized curve (thick) and its algebraic offsets (thin)

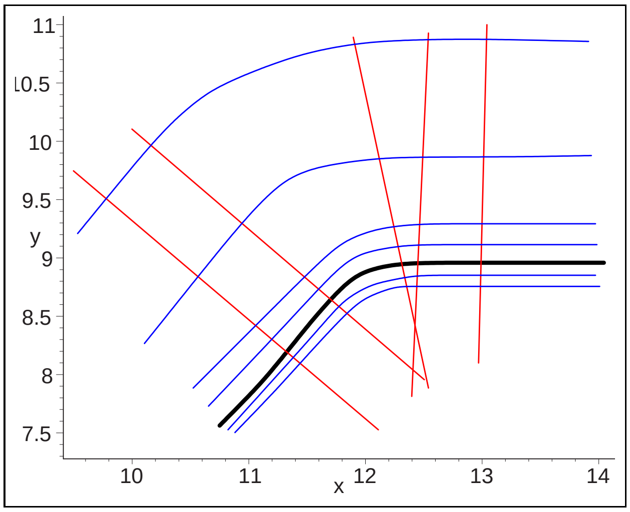

Figure 5.16 C^1 implicitized curve and its algebraic offsets (enlarged)

EXAMPLE 3 Continued

Using the C^1 spline implicitization technique, we derived a piecewise quadratic function G, whose zero contour $G(x, y) = 0$ is the quadratic spline curve (see Fig. 2.7, the case $\mathcal{T} = 10^{-3}$). In order to visualize the quality of the implicitization, Fig. 5.17 shows the algebraic parallel curves or *algebraic offsets* (thin lines), $G(x, y) = c$ for certain constants (algebraic distances) c, and the transversal lines (which defined the polynomial pieces of G) through the junction point. In order to make the picture clearer, we enlarged a part of the curve (see Fig. 5.18). Generally, the algebraic offsets (which consist of parabolic arcs) are not tangent continuous.

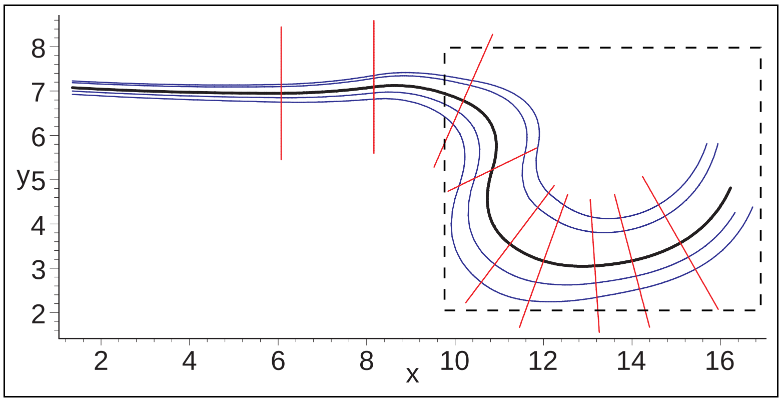

Figure 5.17 C^1 implicitized curve (thick) and its algebraic offsets (thin)

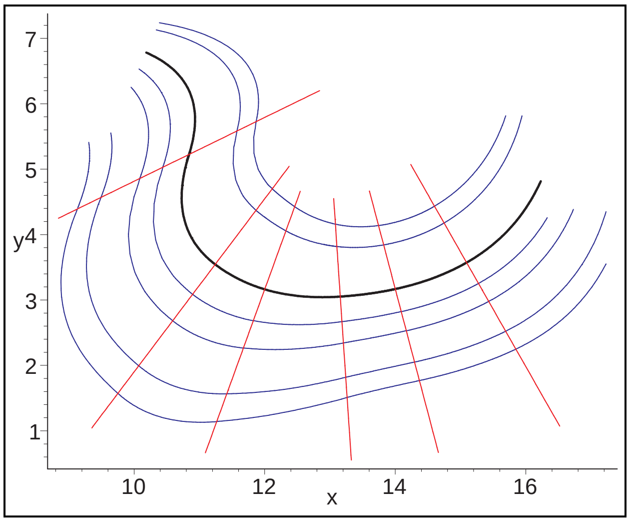

Figure 5.18 C^1 implicitized curve and its algebraic offsets (enlarged)

5.4 Applications

In this section, as application of the C^1 spline implicitization method, we show how the method can be used to compute the intersection points between two planar curves, and to generate a footpoint of a given point on a given planar curve.

5.4.1 Intersection of Two Planar Curves

Computation of the intersection points between two curve is very popular problem [13]. The problem is became easier if one of the curves is defined with the parametric equation while the other is given by its implicit form. In this case, the problem reduces to a one–dimensional root–finding problem.

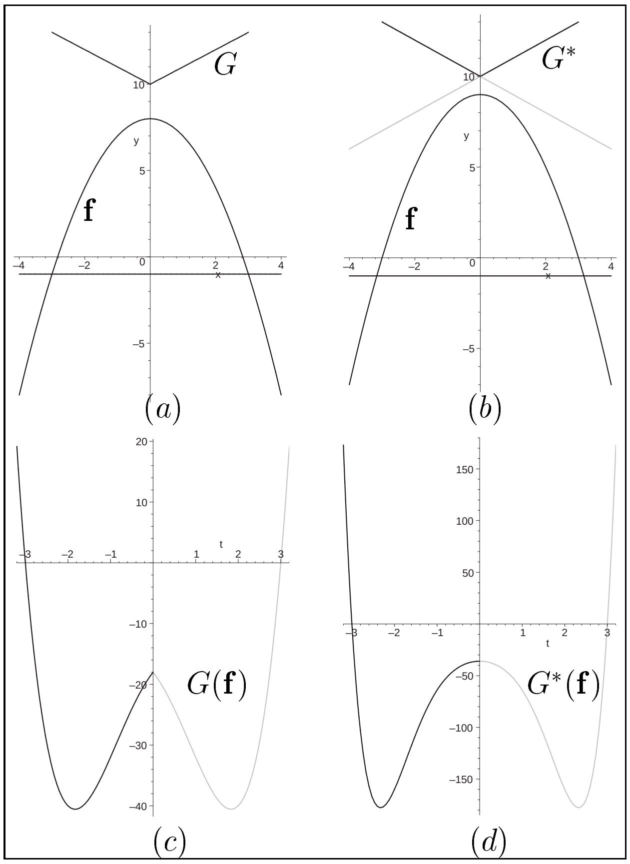

Figure 5.19: Compute the intersection points between a parabola $\mathbf{f}$ and a spline curve G for both C^0 (a, c) and C^1 (b, d) cases

Figure 5.19 a, b shows a parabola $\mathbf{f} = (f_1(t), f_2(t))$, and a spline implicit curve $G(x, y)$ plotted as black line. In order to find the intersection points between the $\mathbf{f}$ and $G(x, y)$, we compute the roots of the univariate polynomial $G(\mathbf{f}) = G(f_1(t), f_2(t))$. In order to compute the roots of $G(\mathbf{f})$, e.g. using *Newton's method*, $G(\mathbf{f})$ should be differentiable.

It is clear that G is continuous but not C^1 function, i.e. $G(\mathbf{f})$ is not differentiable. In order to achieve C^1 along y–axis we multiply G by additional polynomial factors plotted in gray in figure *b*. This give a C^1 function G^*. From the figure, it is clear that $G(\mathbf{f})$ is continuous (Figure 5.19 c) while $G^*(\mathbf{f})$ is a tangent continuous (Figure 5.19 d).

From above, we may conclude that: In order to compute the intersection between two curves, both curves should be differentiable. As an application for the C^1 spline implicitization technique, we show how the proposed technique can be used to compute the intersection between two parametric curves $\mathbf{f}(f_1(t), f_2(t))$ and $\mathbf{g}(g_1(t), g_2(t))$.

- First, we apply the proposed technique to $\mathbf{g}$ as a result we get a bivariate C^1 spline function $G(x, y)$.
- Then, we substitute the parametric form of $\mathbf{f}(\mathbf{t})$ into the implicit form of $G(x, y)$. This give a univariate polynomial $G(\mathbf{f})$.
- Finally, we compute the roots of $G(\mathbf{f})$.

In the sequel, we choose $\mathbf{g}$ as the parameter curve shown in Fig. 1.6. Clearly, the implicit representations G of $\mathbf{g}$ is shown in Fig. 5.17. Figure 5.20 (top) shows a parametric curve $\mathbf{f} = (f_1(t), f_2(t))$ (thin line), and the spline curve G (thick line). We plot the univariate polynomial $G(\mathbf{f})$ in the bottom figure. It is clear that $G(\mathbf{f})$ is a tangent continuous. Using Newton's method, one can compute the root of $G(\mathbf{f})$, i.e. the intersection points between $\mathbf{f}$ and $\mathbf{g}$.

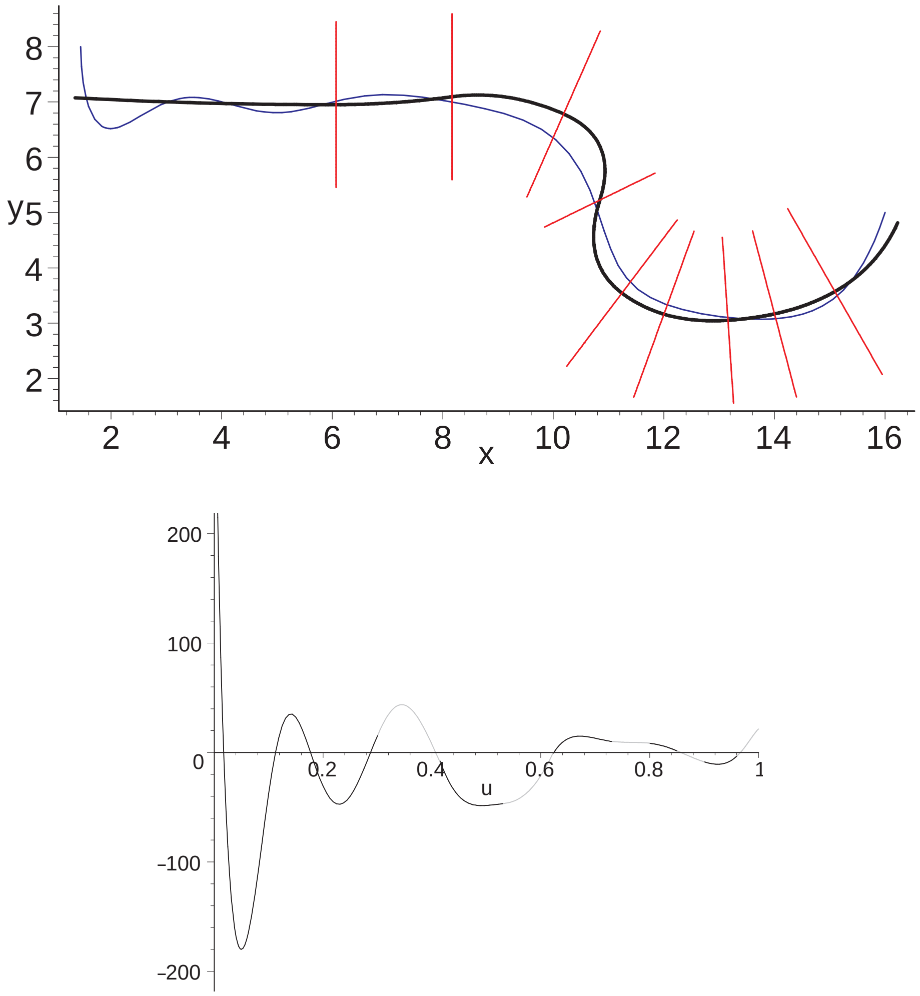

Figure 5.20: Intersection points between the implicit curve G and a given parametric curve $\mathbf{f}$

5.4.2 Foot Point Generation

The foot point of $\mathbf{p}$ on a curve $F(\mathbf{x}) = 0$ is a point $\mathbf{X}$ such that $[(\mathbf{p} - \mathbf{X}), \nabla F(\mathbf{X})] = 0$. Footpoint problem has been discussed by several authors, see e.g. [1] and the references cited therein.

In [1], the authors present an efficient and robust algorithm for addressing the foot point problem for arbitrary data sets and a parametric curves. The emphasis was on finding accurate initial parameter estimates as starting values for standard locally convergent non–linear iterative techniques.

As an application for the C^1 implicitization technique, we propose the following algorithm for generating a foot point $\mathbf{x}$ of a given point $\mathbf{p}$ on a given planar curve $F(\mathbf{X}) = 0$. For the following algorithm, the curve needs to be differentiable (this is the case for the C^1 implicitization) but not need to be twice differentiable as required in [1].

Algorithm

1. Let $i = 0$, and $\mathbf{m}_0 = \mathbf{p}$,
2. draw the line $L_i : \mathbf{m}_i + t\,\nabla F(\mathbf{m}_i)$,
3. let $\mathbf{n}_i$ be the intersection point of L_i and the curve $F(\mathbf{X}) = 0$,
4. draw the tangent T_i to $F(\mathbf{X}) = 0$ at $\mathbf{n}_i$,
5. let $\mathbf{m}_{i+1}$ be the orthogonal projection of $\mathbf{p}$ on T_i,
6. let $i := i + 1$,
7. repeat starting from step 2.

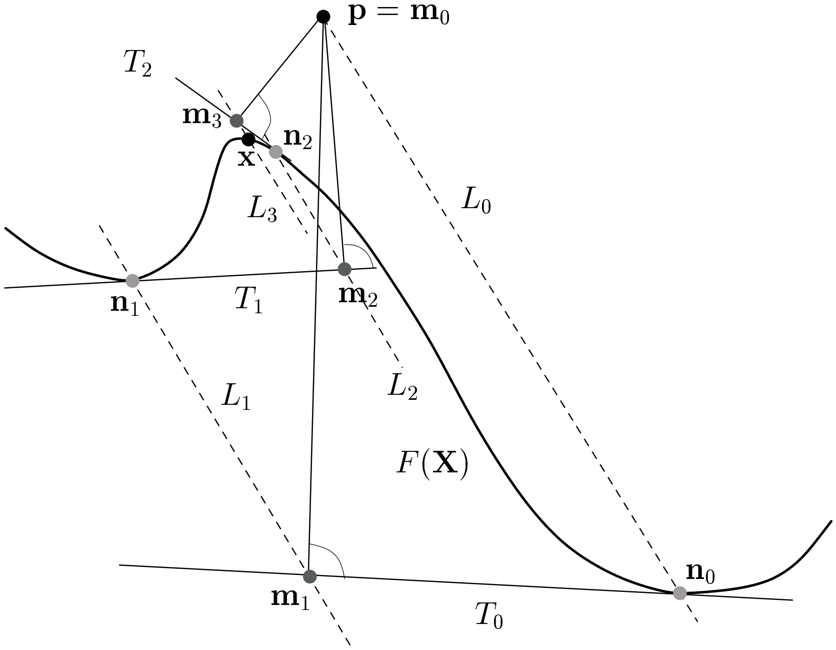

Figure 5.21: The foot point $\mathbf{x}$ of $\mathbf{p}$ on $F(\mathbf{X})$.

After few steps, we are very closed to the foot point $\mathbf{x}$ of $\mathbf{p}$ on $F(\mathbf{X})$. According to our numerical experiments, the above algorithm converge linearly.

Chapter 6

Generating a Hierarchy of Planar Shapes

Multigrid hierarchy of functions play a fundamental role in the area of numerical simulation. In this chapter, as an extended research of the C^1 spline approximation technique, we propose a method for generating a hierarchy of planar shapes which it may help in the generating a hierarchy of functions. A hierarchy of shapes is a sequence of approximation of fixed shape which is simpler and simpler, see Fig. 6.1.

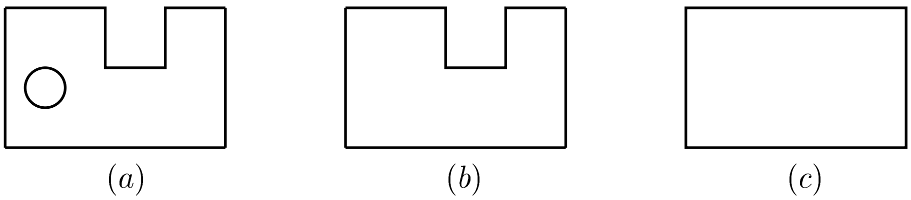

Figure 6.1 A planar shape (a) and its hierarchy of shapes (b, c)

Starting with a planar shape defined by a set of parametric planar curves, first, we apply the C^1 spline implicitization technique for each these parametric planar curve. Then, using the quadratic spline wavelets, we generate a hierarchy of shapes. The main idea is to set all levels detail coefficients to be zero. Going from level to a lower level, we obtained a hierarchy of shapes. It should be noted that: this research is still incomplete and it is open for future investigation.

This chapter is organized as follows: First, we recall briefly our method for C^1 spline implicitization of parametric planar curves. Then, in the second section, we show how the function representation can be modified to be a tensor product representation. Finally, in the third section, we show how the hierarchy of shapes can be generated.

Throughout this chapter, we will use the following example to illustrate the proposed idea.

Example

We consider a planar shape consists of a circle and a piecewise parametric curve $\mathbf{g}$ consists of 16 pieces of degree 4 each, see Fig. 6.2.

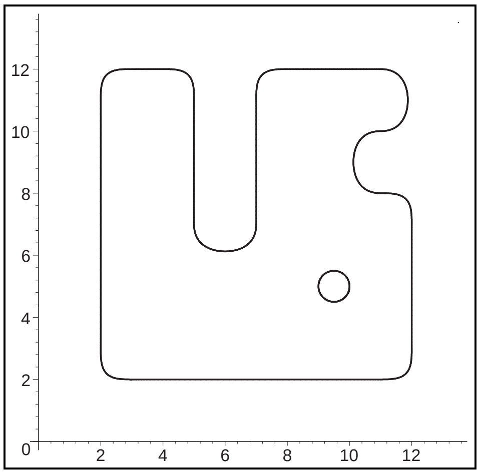

Figure 6.2 A planar shape consists of a circle and a piecewise parametric curve

6.1 C^1 Spline Implicitization

Recall that, the C^1 spline implicitization process consists of four main steps:

- Quadratic B–spline approximation.
- Data reduction.
- Segment-wise implicitization.
- Segments joining along transversal lines.

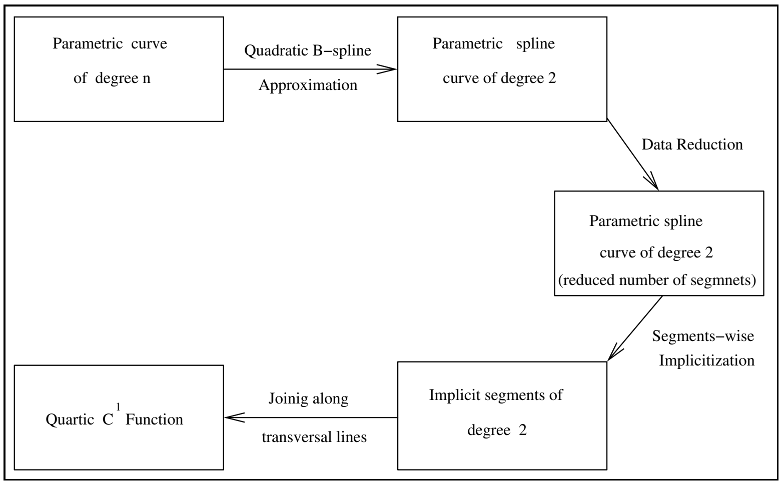

Figure 6.3 C^1 implicitization process

6.1.1 Quadratic B–spline approximation

We apply the quadratic B–spline approximation technique, see Ch. 1, to **g**. Fig. 6.4 shows the original curve (black) and the quadratic spline approximation (gray). The knots are plotted as small circles in the right figure.

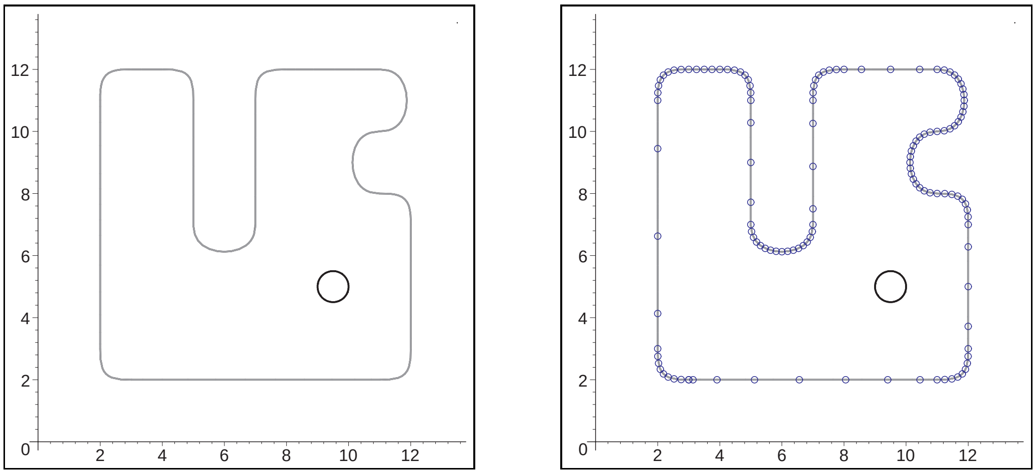

Figure 6.4 The original curve (black) and the quadratic spline approximation (gray). The knots are plotted as small circles (right figure)

6.1.2 Data reduction via wavelets

In order to reduce the number of knots, we apply the data reduction process, Ch. 2. Fig. 6.5 shows the original curve (black) and the spline approximation curve after the data reduction. The knots are plotted as circles.

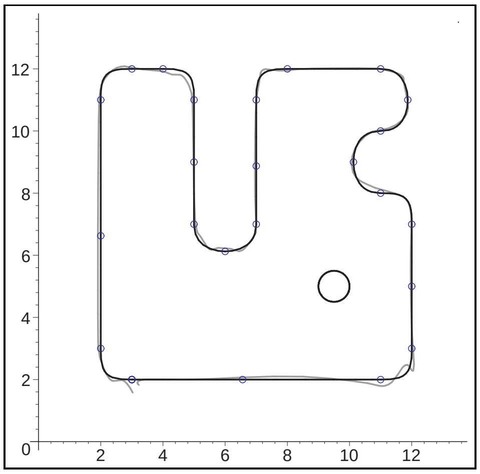

Figure 6.5 The original curve (black) and the quadratic spline approximation after data reduction (gray). The knots are plotted as small circles

6.1.3 Segment–wise implicitization

After applying data reduction, we obtain a quadratic spline approximation $\mathbf{g}^{**}$ defined over the reduced knot sequence. In order to implicitize this curve, the segment–wise implicitization process, Ch. 3, is applied. As a result, we obtain m quadratic bivariate polynomial segments $G_i(x, y)$, $i = 1 \ldots m$.

6.1.4 Joining process

Starting from the quadratic bivariate functions, whose zero contour $G_i(x, y) = 0$, $i = 1 \ldots m$ is the quadratic spline curve $\mathbf{g}^{**}$. We choose the transversal lines as an arbitrary lines passing through the junction points. Following the technique proposed in Ch. 5, we multiplying the quadratic bivariate functions by suitable quadratic polynomial factors. As a result, we obtain a globally C^1 bivariate quartic function $\hat{G}$, whose zero contour $\hat{G}_i(x, y) = 0$, $i = 1 \ldots \hat{m}$ is the quartic spline curve.

Fig. 6.6, shows the algebraic offsets (thin lines) of $\hat{G}$ (thick line) and the transversal lines through the junction points. From the Figure, it can clearly be seen that the algebraic offsets are C^1.

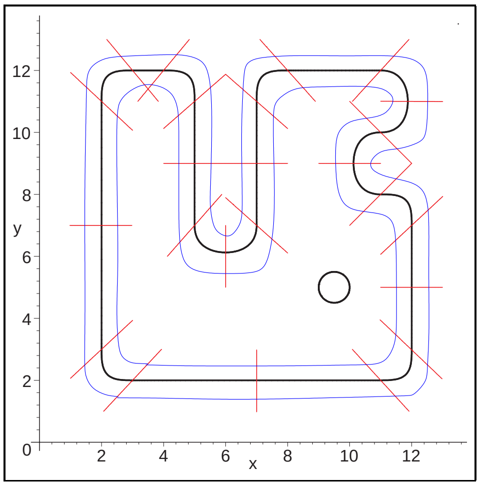

Figure 6.6 C^1 Implicitized curve (thick line) and its algebraic offsets (thin lines)

6.2 Function Representation Modification

In order to generate a hierarchy of shapes, we apply the wavelets transform. Clearly, the spline function should be represented as a tensor product function, in order to compute the wavelets transform. Therefore:

- First, we have to convert the spline function domain, which was produced from C^1 implicitization process, into a grid representation.
- Then, the wavelets transform can be applied to generate the hierarchy of shapes.

6.2.1 Filling Holes

Starting from the quartic bivariate functions, whose zero contour $\hat{G}_i(x, y) = 0$, $i = 1 \ldots \hat{m}$ is the quartic spline curve, see Fig. 6.7.

Each piece $\hat{G}_i$ of $\hat{G}$ is defined over a domain D_i bounded by two consequence transversal lines L_{i-1}, L_i. Clearly, D_i can not be extended beyond the intersection point of L_{i-1} and L_i. In other word, we should partition it

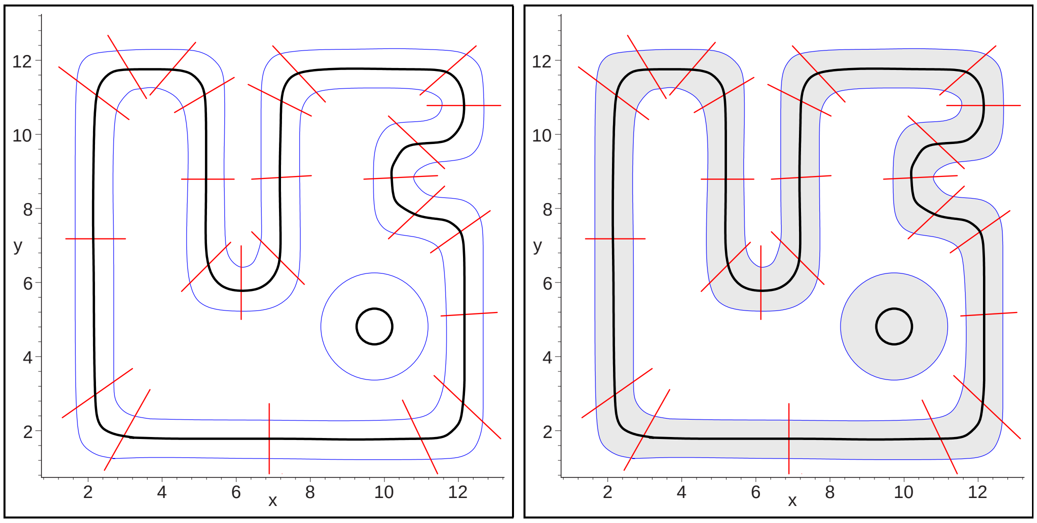

Figure 6.7 Spline function sub–domains

before this point. In the sequel, we partition it using a suitable pair of the algebraic offsets of $\hat{G}_i$. Fig. 6.7 shows the sub–domain $D_i's$ of $\hat{G}_i's$. To make the picture clearer, we filled these domains with gray color.

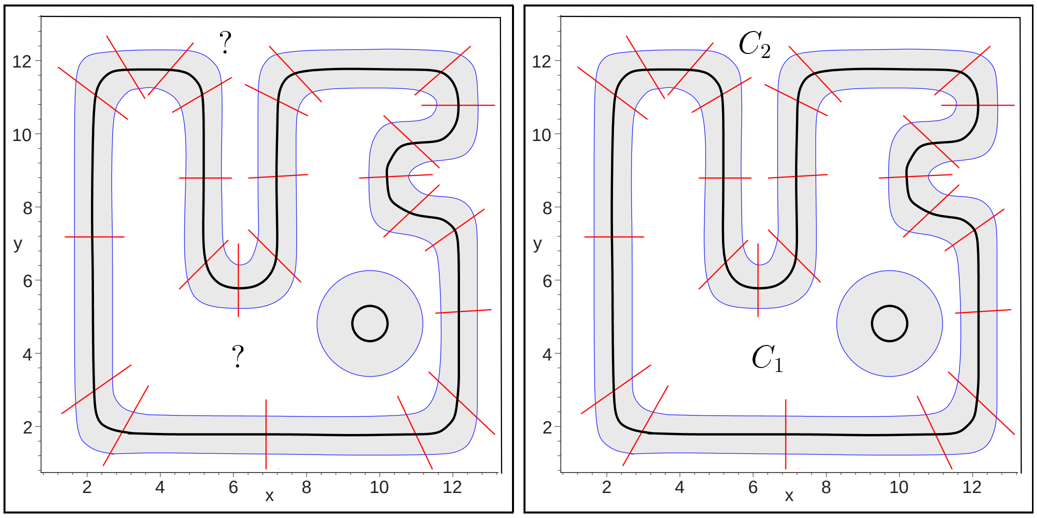

Figure 6.8 Filling the holes by constant functions

Clearly, there are still some sub–domains, which are denoted by the symbol "?" (filled with white color) in Fig. 6.8 (to the left), not including a zero contour of the spline function. Good candidates for filling these sub–domains are suitable constant functions C_j, $j = 1, 2$. We have to take care that these process will not introduce new branches of the zero contour. Filling with constant functions introduces C^1 discontinuity in the function. As an example, $\hat{G}_i$ and $\hat{G}_{i+1}$ are C^1 continuous while neither $\hat{G}_i$, C_1 or $\hat{G}_{i+1}$, C_1 are.

6.2.2 C^1 Continuous Function

In order to have a globally C^1 function, we blend the $\hat{G}_i$'s and $C_j, j = 1, 2$ together. Fig. 6.9 (to the left) shows the zero contour of $\hat{G}_i$ (plotted as thick line) and the algebraic offsets $\hat{G}_i = c_1$ and $\hat{G}_i = c_2$ (plotted as thin line). In order to blend $\hat{G}_i$ with C_1 and C_2, we draw the algebraic offsets $\hat{G}_i$ equal to a, b, $\bar{a}$, $\bar{b}$ respectively (dotted lines), see Fig. 6.9 (to the right), where $a = c_1 + \epsilon_1$, $b = c_1 - \epsilon_1$, $\bar{a} = c_2 - \epsilon_2$, $\bar{b} = c_2 + \epsilon_2$ and ϵ_1, ϵ_2 are suitable tolerances.

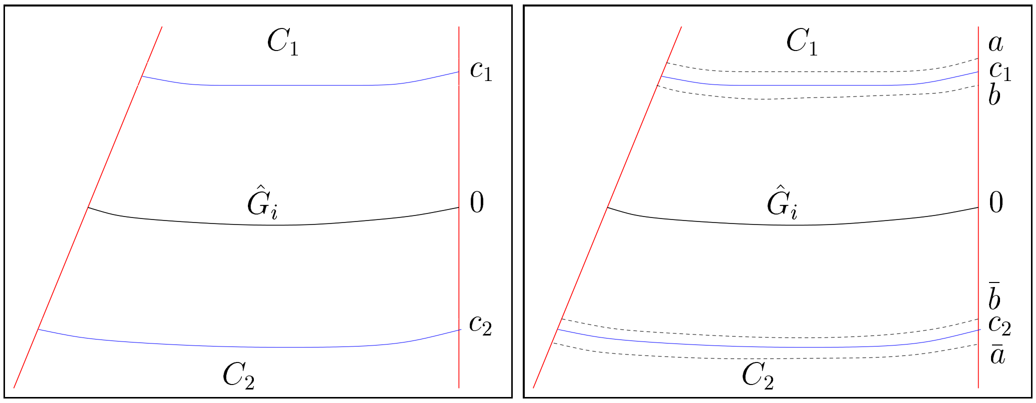

Figure 6.9 Function distribution along the sub–domain

The distribution of the C^1 continuous function $F_i(x, y)$ is defined as:

$$F_i(x,y) = \begin{cases} C_1 & if & \hat{G}_i(x,y) > a \\ \Phi_i(\hat{G}_i(x,y)) & if & b \le \hat{G}_i(x,y) \le a \\ \hat{G}_i(x,y) & if & \bar{b} < \hat{G}_i(x,y) < b \\ \bar{\Phi}_i(\hat{G}_i(x,y)) & if & \bar{a} \le \hat{G}_i(x,y) \le \bar{b} \\ C_2 & if & \hat{G}_i(x,y) < \bar{a} \end{cases}$$

The collection of $F_i(x, y)$ generate a globally C^1 function $F(x, y)$, whose zero contour is the quartic spline curve, see Fig. 6.11 (to the left). The computation of the blending functions $\Phi_i(\hat{G}_i(x, y))$ and $\bar{\Phi}_i(\hat{G}_i(x, y))$ is discussed below.

Blending Functions

The blending function $\Phi_i(\hat{G}_i)$ which blends $\hat{G}_i$, C_j is defined in domain bounded by the two algebraic offsets to $\hat{G}_i$, for example, $\hat{G}_i$ equal to a, b respectively. It is clear that, $\Phi_i(\hat{G}_i)$ should be C^1 continuous with both functions $\hat{G}_i$ and C_j along the algebraic offsets $\hat{G}_i = b$, $\hat{G}_i = a$ respectively, see Fig. 6.10 (to the left). Hence, $\Phi_i(\hat{G}_i)$ should satisfy at least 4 conditions, i.e. we may use a cubic function. Fig. 6.10 (to the right) shows a cross section of the functions. The blending function $\Phi_i(\hat{G}_i)$ blends $\hat{G}_i$, C_j from M into N.

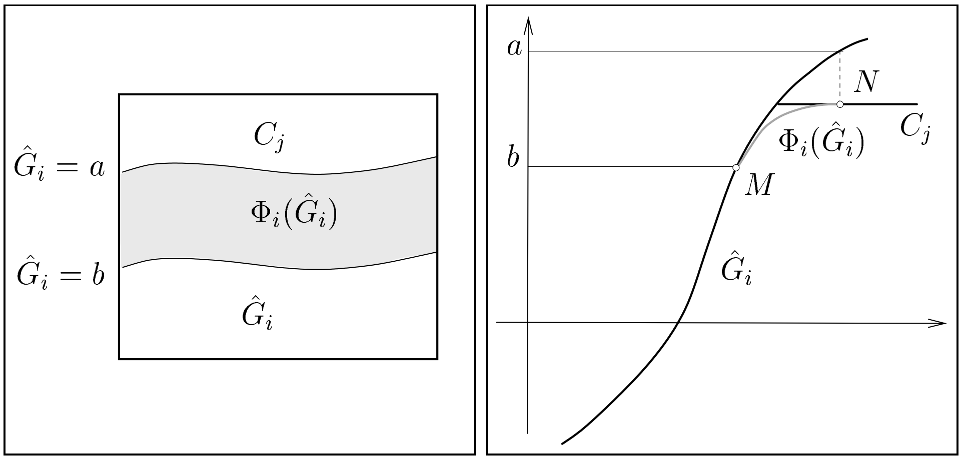

Figure 6.10 Blending function

Let the function Φ_i be defined as a cubic function of $\hat{G}_i$:

$$\Phi_i(\hat{G}_i) = A\,\hat{G}_i^3 + B\,\hat{G}_i^2 + D\,\hat{G}_i + E$$

Φ_i is a tangent continuous with $\hat{G}_i$ along the algebraic offset $\hat{G}_i = b$:

$$\Phi_i(b) = b, \qquad b = A\,b^3 + B\,b^2 + D\,b + E$$
$$\Phi_i'(b) = 1, \qquad 1 = 3\,A\,b^2 + 2\,B\,b + D$$

Φ_i is a tangent continuous with C_j along the algebraic offset $\hat{G}_i = a$:

$$\Phi_i(a) = C_j, \qquad C_j = A\,a^3 + B\,a^2 + D\,a + E$$
$$\Phi_i'(a) = 0, \qquad 0 = 3\,A\,a^2 + 2\,B\,a + D$$

Solving the above 4 equations, we get:

$$A = \frac{a + b - 2\,C_j}{a^3 + 3\,b^2\,a - 3\,b\,a^2 - b^3}, \qquad B = \frac{3\,a\,C_j + 3\,b\,C_j - 2\,b\,a - 2\,b^2 - 2\,a^2}{a^3 + 3\,b^2\,a - 3\,b\,a^2 - b^3},$$
$$D = \frac{a\,(b\,a + a^2 - 6\,b\,C_j + 4\,b^2)}{a^3 + 3\,b^2\,a - 3\,b\,a^2 - b^3}, \quad E = \frac{b^2\,(3\,a\,C_j - 2\,a^2 - b\,C_j)}{a^3 + 3\,b^2\,a - 3\,b\,a^2 - b^3}.$$

6.2.3 Quasi Interpolation

In order to have a spline tensor product representation of the above spline function, we apply a quasi interpolation method to the spline function.

Quasi interpolation has been discussed in a number of publication, see

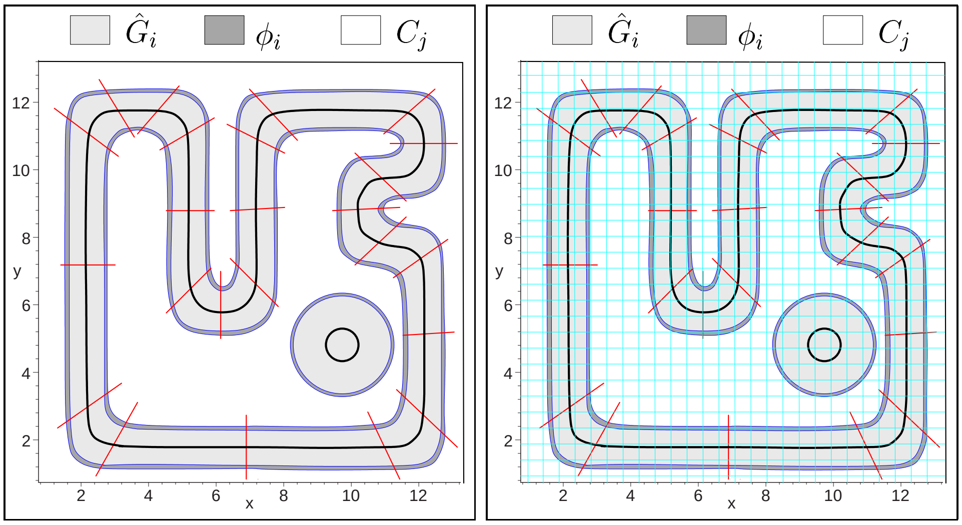

Figure 6.11 Quasi interpolation

e.g.[32] and the references cited therein. The quasi interpolation approximation $\hat{F}$ of F is written in the form

$$\hat{F}(x,y) = \sum_{i,j} Q(F(x,y))_{i,j}\ B_i(x)\ B_j(y)$$

where B_i are the B–splines and $Q(F(x,y))_{i,j}$ are the quasi coefficients.

In the sequel, we just use a very simple method of quasi interpolation. The method is good enough for our application. In order to compute these coefficients. We evaluate the function at the grid points. For each of the grid points one need to determine the value of F_i i.e. . This is some how a geometric searching problem (point location problem). This problem is a well know problem in computational geometry and one may use an existing algorithm to solve it, for more information one may consult a suitable text book e.g. [28, 29]. Using a fine grid structure one may get a good tensor product approximation $\hat{F}$ of F. Fig. 6.11 shows the domain of the spline function F and the grid structure. Fig. 6.12 (to the left) shows the spline function of F while the corresponding tensor product function $\hat{F}$ is shown in the right figure.

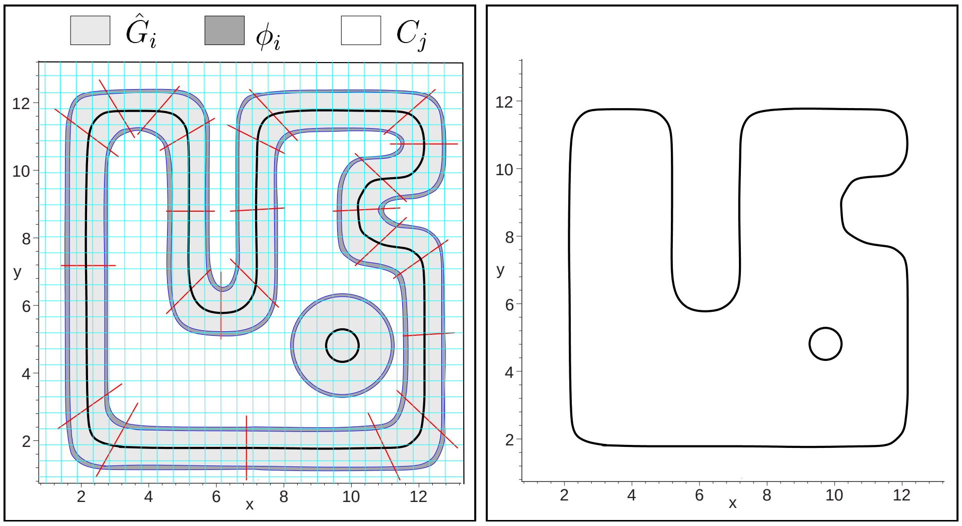

Figure 6.12 The spline function of F (left) and the corresponding tensor product function $\hat{F}$ (right)

6.3 Hierarchy Shape Structure

In order to generate a hierarchy of shapes, we apply the wavelets transform. Wavelets are a mathematical tool for hierarchically decomposing functions. In addition to the hierarchical nature, they have many useful properties, such as linear time complexity, sparsity and adaptability. A wavelet transform of a function consists of a coarse overall approximation together with detail coefficients that influence the function at various scales.

One advantage of the wavelets transform is that often a large number of the detail coefficients turn out to be *very small in magnitude*. Removing these small coefficients from the representation introduce only *small errors* in the reconstructed function.

In the sequel, we use a quadratic spline wavelet to generate the hierarchy of shapes. Spline wavelets, which are built from B–splines, have been developed to a large extent in [4], [5]. Of particular interest for curve editing applications are endpoint interpolating B–splines. These wavelets are *semiorthogonal wavelets*[1]. For more information about spline wavelets the reader may consult a wavelet textbook, e.g., [43, 45].

[1]Wavelets of a given level are a least orthogonal to all coarser scaling function, though not necessarily orthogonal to each other. In mathematical terms, a semiorthogonal wavelet basis satisfy the conditions $\langle \phi_k^j | \psi_\ell^j \rangle = 0$ for all j, k, ℓ.

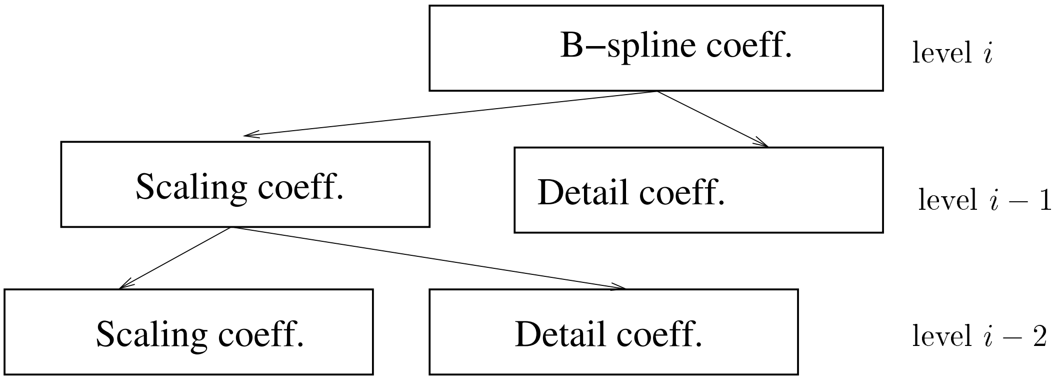

Figure 6.13 Wavelets Transform

Our main idea to generate a hierarchy of shapes is to run the wavelets transform. Then, for each of the level hierarchy, we set the detail coefficients to be zero. As an output, we obtained a hierarchy of shapes.

Starting from the tensor product spline function, Fig. 6.12 (to the right), we compute the wavelets transform of the given spline tensor product function. The grid size for the level i is $2^i + 2 \times 2^i + 2$. For each level of the hierarchy, we set the detail coefficients to be zero and proceed with the scaling coefficients. In other word, we reduce the grid size by $2^{i-1} \times 2^{i-1}$ by going from level i into level $i-1$. By computing the wavelets transform recursively, one may get a hierarchy shapes of the given function.

For our example we start from level 7, when we apply the wavelet transform the grid size reduced to 66×66 in level 6. Applying the wavelet transform recursively, one may get the hierarchy shapes of the given function.

Fig. 6.14 shows the hierarchy shape structure of the given planar shape. It is clear from the figure that the shapes become simpler and simpler when we go from a level to a lower one. We hope that this results will help in generating a hierarchy of functions. It should be noted that: this is still a primary results. This research is still incomplete and it is open for future investigation.

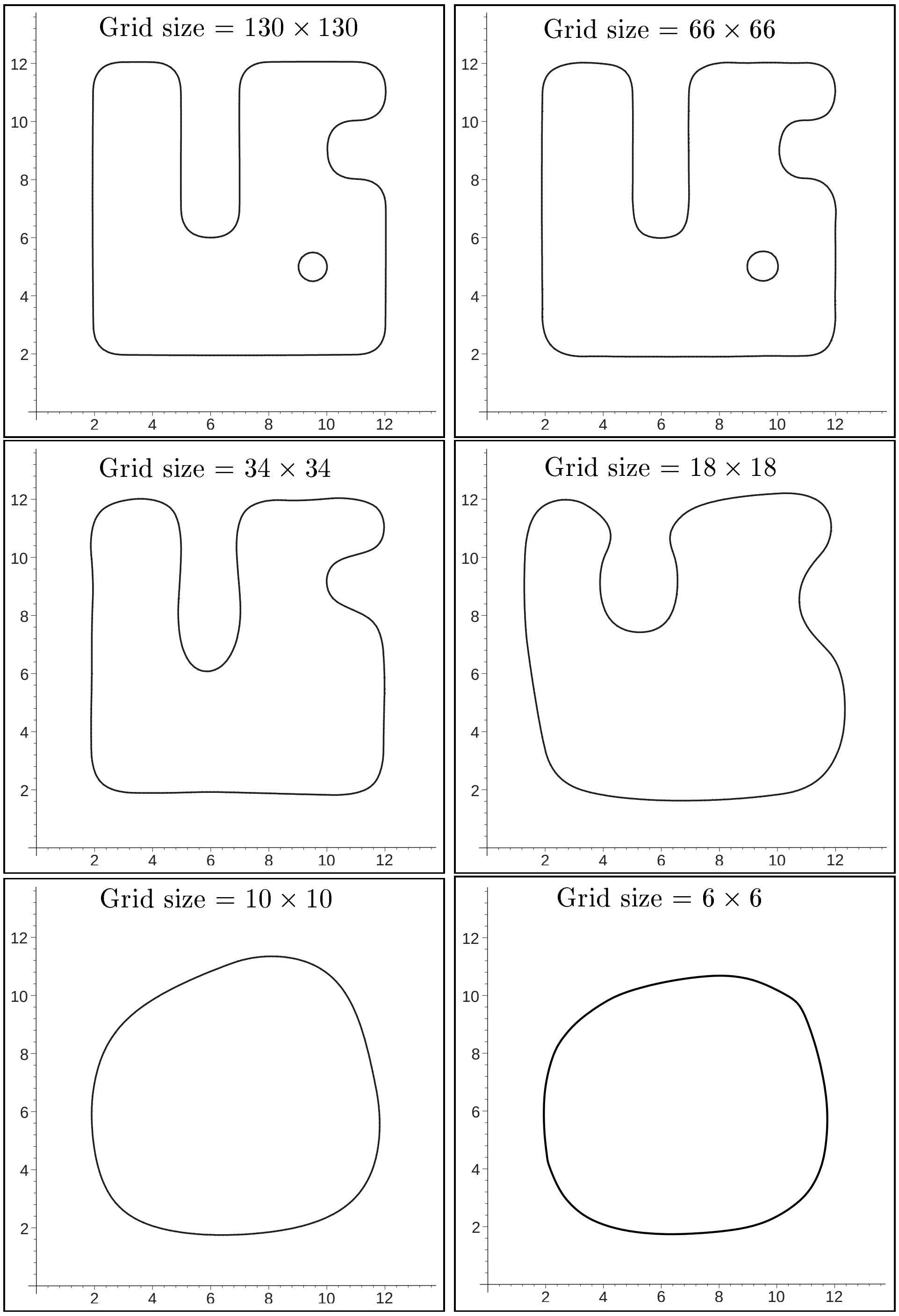

Figure 6.14 Hierarchically shapes structure of a planar shape

Bibliography

[1] I. J. Anderson, M. G. Cox, A. B. Forbes, J. C. Mason and D. A. Turner, *An Efficient and Robust Algorithm for Solving the Foot Point Problem*, in: M. Dæhlen, T. Lyche, and L. L. Schumaker (eds.), Mathematical Methods for Curves and Surfaces II, Vanderbilt University Press, Nashville 1998, pp. 9–16.

[2] B. Buchberger, *Application of Gröbner bases in non-linear computational geometry*, Mathematical Aspects of Scientific Software(J. Rice, Ed.), Springer-Verlag, New York/Berlin, 1988.

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

[4] C. K. Chui, *An Introduction to Wavelets*, Academic Press, Boston, 1992.

[5] C. K. Chui and Ewald Quak, *Wavelets on a bounded Interval*, In D. Braess and L. L. Schumaker, editors, *Numerical Methods in Approximation Theory*, volume 9, pp. 53–75, Birkhauser Verlag, Basel, 1992.

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

[7] C., De Boor, *A Practical Guide to Splines*, Springer Verlag, New York, 1978.

[8] T. Dokken, *Aspects of Intersection Algorithms and Approximation*, Ph.D. thesis, University of Oslo, July 1997.

[9] T. Dokken, *Approximate implicitization*, Mathematical Methods in CAGD, T. Lyche, L. L. Schumaker (eds.), Vanderbilt University Press (Nashville & London), 2001.

[10] M. Eck, and J. Hadenfeld, *Knot removal for curves*, Computer Aided Geometric Design 12, 1995, pp. 259–282.

[11] G. Farin, G. Rein, N. Sapidis, and A. J. Worsey, *Fairing cubic curves*, Computer Aided Geometric Design 4, 1987, pp. 91–103.

[12] G. Farin, *Curves and Surfaces for Computer Aided Geometric Design*, Academic Press, ING., San Diego, CA., 1988.

[13] G. Farin, J. Hoschek and M. Kim (editors), *Handbook of Computer Aided Geometric Design*, Elsevier, 2002.

[14] R. T. Farouki, C. Manni, A. Sestini, *Real-time CNC interpolators for Bézier conics*, Computer Aided Geometric Design 18, 2001, pp. 639–655.

[15] J. Hoschek and D. Lasser, *Fundamentals of Computer Aided geometric Design*, A K Peters, Ltd., Wellesley, Massachusetts, 1993.

[16] B. Jüttler, *Hermite Interpolation by Pythagoran Hodograph Curves of Degree Seven*, Mathematics of Computation, 70, 235, 2000, pp. 1089–1111.

[17] B. Jüttler, *Least-squares fitting of algebraic spline curves via normal vector estimation*, in: R. Cipolla and R.R.Martin (ed.), The Mathematics of Surfaces IX, Springer, London 2000, pp. 263–280.

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

[19] B. Jüttler, J. Schicho and M. Shalaby, *Spline Implicitization of Planar Curves*, in: T. Lyche, M. Mazure, and L. L. Schumaker (eds.), Curves and Surfaces Design: Saint–Malo 2002, Nashboro press, Brentwood 2003, pp. 225–234.

[20] E. Kreyszig, *Differential geometry*, Dover (Toronto), 1991.

[21] J. Kjellander, *Smoothing of cubic parametric splines*, Computer Aided Design 15, 1983, pp. 175–179.

[22] J. D. Lawrence, *A Catalog of Special Plane Curves*, New York, Dover, 1972.

[23] T. Lyche, *Local Spline Approximation Methods*, Journal of Approximation Theory 15, 1975, pp. 294–325.

[24] T. Lyche, and K. Mørken, *Knot removal for parametric curves and surfaces*, Computer Aided Geometric Design 4, 1987, pp. 217–230.

[25] T. Lyche, and K. Mørken, *A data-reduction strategy for splines with applications to the approximation of functions and data*, IMA J. Numer. Anal., 1988, pp. 185–208.

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

[27] A. W. Nutbourne and R. R. Martin, *Differential Geometry Applied to Curves and Surfaces Design*, Ellis Horwood Ltd., Chichester, England, 1988.

[28] F. P. Preparata and M. L. Shamos, *Computational Geometry: An Introduction*, Springer–Verlag, New York, 1985.

[29] J. O'Rourke, *Computational geometry in C*, Cambridge University Press, 1994.

[30] Ulrich Reif, *Orthogonality Of Cardinal B-Spline In Weighted Sobolev Spaces*, SIAM J. Math. Anal. 28, 5, 1997, pp. 1258–1263.

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

[32] Paul Sablonniere, *On some multivariate quadratic quasi-interpolants on bounded domains*, Prepublication IRMAR 02–55, INSA de Rennes, France, 2002.

[33] Paul Sablonniere, *BB–coefficients of basic quadratic spline on rectangular domains with uniform criss–cross triangulations*, Prepublication IRMAR 02–56, INSA de Rennes, France, 2002.

[34] T. Sakkalis and R. T. Farouki, *Real rational curves are not "unit speed"*, Computer Aided Geometric Design 8, 1991, pp. 151–157.

[35] G. Salmon, *Higher Plane Curves*, G. E. Stechert & Co., New York, (1934).

[36] N. Sapidis, and G. Farin, *Automatic fairing algorithm for curves*, Computer Aided Geometric Design 22, 1990, pp. 121–129.

[37] L. L. Schumaker, *Spline Functions: Basic Theory*, John Wiley, New York, 1981.

[38] T. W. Sederberg, D. C. Anderson and R. N. Goldman, *Implicit representation of parametric curves and surfaces*, Computer Vision, Graphics, and Image Processing 28, 1984, pp. 72–84.

[39] T. W. Sederberg, *Planar Piecewise algebraic Curves*, Computer Aided Geometric Design 1, 1984, pp. 241–255.

[40] T. W. Sederberg and F. Chen, *Implicitization using moving curves and surfaces*, Computer Graphics (SIGGRAPH 95 Conference Proceedings) (R. Cook, Ed.), Vol. 29, pp. 301–308. Addison-Wesley, Reading, MA, 1995.

[41] T. W. Sederberg, J. Zheng, K. Klimaszewski and T. Dokken, *Approximate Implicitization Using Monoid Curves and Surfaces*, Graphical Models and Images processing 61, 1999, pp. 177–198.

[42] M. shalaby, B. Jüttler and J. Schicho C^1 *Spline Implicitization of Planar Curves*, ADG'02, Linz, Austria.

[43] E. J. Stollnitz,T. D. Derose, and D. H. Salesin, *Wavelets for Computer Graphics: Theory and Applications*, Morgan-Kaufmann Publishers, Inc., San Francisco, 1996, pp. 90–97.

[44] W. Tiller, *Knot-removal algorithms for NURBS curves and surfaces*, Computer Aided Design 24, 1992, pp. 445–453.

[45] P. Wojtaszczyk, *A Mathematical Introduction to wavelets*, Cambridge University Press, 1997, pp. 51–65.

Appendix A

Maple worksheet

For the convenience of the reader, the Maple worksheet which used in the proof of theorem 1 is listed below. Please check which Maple version you have and uncomment the corresponding lines below:

```
> restart: with(linalg):
> # ******* spline segments and derivatives ************
> b:=vector(4): bd := vector(4): bdd := vector(4):
> for i from 1 to 4 do b[i]:=vector(6, 0); od:
> # let n the degree of the bspline
> n:=2:
>
> ################################################
> ########  Important note !!!! ###################
> ################################################
> ################################################
> # For Maple 5 users, uncomment the following line
> # readlib(bspline)
> # for Maple 6 users,uncomment the following line
> # with(BSpline):
> # for Maple 7 users, uncomment the following two lines.
> # (Maple 7 use n as order not as the degree)
> # with(CurveFitting):
> # n:=n+1:
> ##################################################
>
> pb:=vector(4):
> for i from 1 to 4 do
> pb[i]:=bspline(2,s,[(i-4)*h,(i-3)*h,(i-2)*h,(i-1)*h]); od:
> # b: vector of spline segments
> for i from 1 to 4 do for j from 1 to 3 do
>   b[i][j+i-1] := factor(op(pb[i])[2*j+2])
> od; od;
> # bd: vector of 1st derivatives
> for i from 1 to 4 do bd[i] := map(diff,b[i],s); od:
> # bdd: vector of 2nd derivatives
> for i from 1 to 4 do bdd[i] := map(diff,bd[i],s); od:
>
> # ****** integral of piecewise functions ************
> intspl := proc(f,g,h) local ii; global s;
>   RETURN(
> int(f[1]*g[1], s=-3*h..-2*h)+int(f[2]*g[2],
>         s=-2*h..-1*h)+int(f[3]*g[3], s=-1*h..0*h)+
```

```
> int(f[4]*g[4], s=0*h..1*h)+int(f[5]*g[5],
>         s=1*h..2*h)+int(f[6]*g[6], s=2*h..3*h));
> end:
>
> # ******* Reif's weights ******************************
>
> eq1:=w0*intspl(b[1],b[1],h)+w1*intspl(bd[1],bd[1],h)
>         +w2*intspl(bdd[1],bdd[1],h)=1:
> eq2:=w0*intspl(b[1],b[2],h)+w1*intspl(bd[1],bd[2],h)
>         +w2*intspl(bdd[1],bdd[2],h)=0:
> eq3:=w0*intspl(b[1],b[3],h)+w1*intspl(bd[1],bd[3],h)
>         +w2*intspl(bdd[1],bdd[3],h)=0:
> assign(solve({eq1,eq2,eq3},{w0,w1,w2}));w0,w1,w2;
>
> # ******* Taylor expansion of arc length *****************
>
> SS := series(Delta + O(1)*Delta^10, Delta, 10);
> # ** canonical Taylor expansion of curve wrt arc length ***
> # results from Frenet's formulas
> X := map(simplify, vector(2, [
>   taylor(SS - Kappa0^2*SS^3/6 - 3*Kappa0*Kappa1*SS^4/24 +
>           c51*SS^5 + c61*SS^6  + c71*SS^7 + c81*SS^8 +
>c91*SS^9 + O(1)*SS^10, Delta, 10),
>   taylor(Kappa0*SS^2/2 + Kappa1*SS^3/6 +
>         (Kappa2-Kappa0^3)*SS^4/24+ c52*SS^5
>  + c62*SS^6 + c72*SS^7 + c82*SS^8 + c92*SS^9 + O(1)*SS^10,
>          Delta, 10)
>     ])):
> Y:=vector(2,[subs(Delta=s,X[1]), subs(Delta=s,X[2])]):
> # derivatives
> Yd:=map(diff, Y, s): Ydd:=map(diff, Y, s$2):
> # vectors of components and derivatives
> # (needed for piecewise integration)
> Y1vec:=vector(6, [seq(convert(Y[1],polynom),i=1..6)]):
> Y2vec:=vector(6,[seq(convert(Y[2],polynom), i=1..6)]):
> Yd1vec:=vector(6, [seq(convert(Yd[1],polynom),i=1..6)]):
> Yd2vec:=vector(6,[seq(convert(Yd[2],polynom), i=1..6)]):
> Ydd1vec:=vector(6, [seq(convert(Ydd[1],polynom),i=1..6)]):
> Ydd2vec:=vector(6,[seq(convert(Ydd[2],polynom), i=1..6)]):
>
> # ******* find spline coefficients *************
>
> d:=array(1..4);
> for i from 1 to 4 do
> d[i]:=vector(2):
> d[i][1]:= taylor(factor(w0*intspl(Y1vec,b[i],h)
>                       +w1*intspl(Yd1vec,bd[i],h)
>                       +w2*intspl(Ydd1vec,bdd[i],h)),h,6);
> d[i][2]:= taylor(factor(w0*intspl(Y2vec,b[i],h)
>                       +w1*intspl(Yd2vec,bd[i],h)
>                       +w2*intspl(Ydd2vec,bdd[i],h)),h,6);
> od:
>
> # ***  generate segments of approximating quadratic spline *********
>
> for j from 1 to 6 do f1[j]:=subs(s=h*S,sum('d[i][1]*b[i][j]','i'=1..4));
>                 od:
> for j from 1 to 6 do f2[j]:=subs(s=h*S, sum('d[i][2]*b[i][j]','i'=1..4));
>                  od:
> # *** check approximation order of segments 3 (-h,0) and 4 (0,h)
> taylor(collect(factor(convert(f1[3],polynom))-subs(s=h*S,Y[1]),h),h,3);
> taylor(collect(factor(convert(f2[3],polynom))-subs(s=h*S,Y[2]),h),h,3);
```

```
> taylor(collect(factor(convert(f1[4],polynom))-subs(s=h*S,Y[1]),h),h,3);
> taylor(collect(factor(convert(f2[4],polynom))-subs(s=h*S,Y[2]),h),h,3);
> # show spline coefficients
> taylor(d[1][1],h,4),taylor(d[2][1],h,4),taylor(d[3][1],h,4),
>                   taylor(d[4][1],h,4);
> taylor(d[1][2],h,4),taylor(d[2][2],h,4),taylor(d[3][2],h,4),
>                     taylor(d[4][2],h,4);
>
> # ********  implicitize segment #3 "left" *****************
>
> impl3:=subs(seq(h^i=0,i=8..20),eliminate({X=convert(f1[3],polynom),
>            Y=convert(f2[3],polynom)},{S})[2][1]);
>
> # *** find intersections between segments 3 "left" and 4 "right" **
>
> # subs segment #4 "right" into implicit equation of segmnet #3 "left"
> eq2:= factor(subs(seq(h^i=0, i=8..20),
>                expand(subs(X=convert(f1[4],polynom),
>              Y=convert(f2[4],polynom),impl3)))):
> # gives a quartic equation, but h^3 * S^2 factors out ->
> #                                quadratic equation eq3
> factor(eq2);
> eq3:=collect(factor(-eq2/14400/h^3/S^2/36/h^3/Kappa0^6),S):
> coeff(eq3,S,2);
> peq3:=coeff(eq3,S);qeq3:=coeff(eq3,S,0);
> # discriminant of this quadratic equation
> h4Disk:=series(factor(h^4*(peq3^2/4-qeq3)),h):
> cc2:=solve(coeff(expand((2/Kappa0^2+X*h^2)^2),h,2)=coeff(h4Disk,h,2),X):
> cc3:=solve((coeff(expand((2/Kappa0^2+cc2*h^2+X*h^3)^2),h,3)=
>            coeff(h4Disk,h,3)),X):
> cc4:=solve((coeff(expand((2/Kappa0^2+cc2*h^2+cc3*h^3+X*h^4)^2),h,4)=
>                coeff(h4Disk,h,4)),X):
> cc5:=solve((coeff(expand((2/Kappa0^2+cc2*h^2+cc3*h^3+cc4*h^4+X*h^5)^2)
>                                                               ,h,5)=
>              coeff(h4Disk,h,5)),X):
> # square root of discriminant (manually):
> h4Disksqrt:=2/Kappa0^2+cc2*h^2+cc3*h^3+cc4*h^4+cc5*h^5:
> taylor(expand(h4Disksqrt^2-h4Disk),h,6);
> # parameter values of intersections (wrt 4th segment)
> # from quadratic equation
> S1:=factor(-peq3/2-h4Disksqrt/h^2): S2:=factor(-peq3/2+h4Disksqrt/h^2):
>
> # check that they are solutions...
>
> taylor(factor(subs(S=S1,h^3*eq3)),h,6);
> taylor(factor(subs(S=S2,h^3*eq3)),h,6);

> # coordinates of 1st intersection
>
> x[1]:=taylor(factor(subs(S=S1, convert(f1[4],polynom))),h,2);
> y[1]:=taylor(factor(subs(S=S1, convert(f2[4],polynom))),h,2);
>
> # coordinates of 2nd intersection
>
> x[2]:=taylor(factor(subs(S=S2, convert(f1[4],polynom))),h,2);
> y[2]:=series(factor(subs(S=S2, convert(f2[4],polynom))),h,2);
>
>
> # result: second intersection converges to normal of the curve,
> # provided that kappa0 <> 0
> # (no inflection point) and original curve is given in
> # arc-length parameterization.
```

List of Figures

Curriculum Vitae

Personal Data

- Name: Mohamed Shalaby
- Date of Birth: 29th of November, 1967
- Place of Birth: Alexandria (Egypt)
- Nationality: Egyptian
- Address: Home: Weingarten 1, A-4232 Hagenberg, Austria.
- Office: RISC-Linz, Johannes Kepler University, A-4040 Linz, Austria
- Contact:++43 (732) 2468 9955 (phone), 9930 (fax)
  shalaby@risc.uni-linz.ac.at http://www.risc.uni-linz.ac.at/people/shalaby

Marital Status

Married (Hebatalla Fouad), two children (Mahmoud 1.3.1995, Fathy 11.2.1999)

Education, Positions

- 1985 Sep.-1990 Jun. B.Sc. in Mechanical Engineering, Alexandria University, Alexandria, Egypt.
- 1991 Jan.-1992 Mar. Civil service.
- 1992 Apr.-1996 Dec. M.Sc.in Mathematical Engineering, Alexandria University, Alexandria, Egypt.
- 1997 Jan.-1999 Sep. Teacher Assistance in Mathematical Engineering Department, Alexandria University, Alexandria, Egypt.

- 1999 Oct.-2001 Mar. Research Assistance SFB01303, [1] Linz, Austria.
- 2001 Apr.- Research Assistance SFB01315, [2] Linz, Austria.

Papers

1. B. Jüttler, J. Schicho and M. Shalaby: Spline Implicitization of Planar Curves. In *Curves and Surfaces Design*, T. Lyche, M. Mazure, and L. L. Schumaker editors, pages 225-234. Saint–Malo 2002, Nashboro press, Brentwood 2003.
2. M. Shalaby, B. Jüttler, and J. Schicho: C1 Spline Implicitization of Planar Curves. In *Automated Deduction in Geometry*, Linz, Austria, 2002.

Conference Talks, Posters

1. M. Moustafa, A. El Sherif, A. El sheik and M. Shalaby: Exact Spherical Function Generator by Means of Stereograhic Projection, poster at *ISSAC 2000*, St Andrews University, Scotland, (2000).
2. G. Bodnár, P. Pau, S. Ratschan, J. Schicho,M. Shalaby: Exact Real Computation in Computer Algebra, poster at *ISSAC 2000*, St Andrews University, Scotland, (2000).
3. G. Bodnár, P. Pau, S. Ratschan, J. Schicho, M. Shalaby: Exact Real Number Arithmetic in Maple, *SYNASC 2000*, University of Timisoara, Romania (2000).
4. B. Jüttler, J. Schicho and M. Shalaby: Spline Implicitization of Planar Curves *Geometrie-Tagun*, Graz, Austria (2002).
5. B. Jüttler, J. Schicho and M. Shalaby: C0 Spline Implicitization of Planar Curves *Curves and Surfaces*, Saint–Malo, France (2002).
6. B. Jüttler, J. Schicho and M. Shalaby: C1 Spline Implicitization of Planar Curves, poster at *ISSAC 2002*, Lille, France (2002).
7. B. Jüttler, J. Schicho and M. Shalaby: C1 Spline Implicitization of Planar Curves. *ADG 2002*, Linz, Austria (2002).

[1] See URL http://www.sfb013.uni-linz.ac.at/projects/projectf1303.html
[2] See URL http://www.sfb013.uni-linz.ac.at/projects/projectf1315.html

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Dissertation selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt bzw. die wärtlich oder sinngemäß entnommenen Stellen als solche kenntlich gemacht habe.