Approximate Parameterization by Planar Rational Curves

Bert Jüttler* and Pavel Chalmovianský†
Institute of Applied Geometry, Johannes Kepler University, Linz, Austria

Abstract

We describe a method for approximate parameterization of a planar algebraic curve by a rational Bézier (spline) curve. After briefly discussing exact methods for parameterization and methods for rational interpolation, we describe a new technique for rational parameterization. Our approach is based on the minimization of a suitable non–linear objective function, which takes both the distance from the curve and the positivity of the weight function (i.e., the numerator of the rational parametric representation) into account. The solution is computed by using an SQP–type optimization technique. In addition, we use a region–growing–type approach in order to obtain a good initial solution, which is crucial for the convergence of the nonlinear optimization procedure.

CR Categories: G.1.1 [Mathematics of Computing]: Numerical Analysis—Interpolation G.1.2 [Mathematics of Computing]: Numerical Analysis—Approximation I.3.5 [Computing Methodologies]: Computer Graphics—Computational Geometry and Object Modeling

Keywords: algebraic spline curves, approximate parameterization

1 Introduction

Among other possibilities, piecewise rational parametric representations [Hoschek and Lasser 1993] (NURBS – Non–Uniform Rational B–Splines) and implicitly defined curves and surfaces [Bloomenthal 1997] are two of the main approaches for describing geometrical objects in Computer Aided Design and Geometric Modeling. Until recently, most industrial applications were dominated by parametric representations. However, many computational problems (such as the analysis and computation of surface–surface intersections) are much simpler to deal with if both representations are available. This has motivated research on conversion methods between the implicit and the parametric form: *implicitization* and *parameterization*.

Exact methods for *implicitization* have been thoroughly explored in the field of Computer Algebra. Exact implicitization techniques rely on symbolic tools such as resultants [Cox et al. 1997], Gröbner bases, or moving curves and surfaces (syzygies) [Sederberg and Chen 1995]. They have some associated difficulties; e.g., even moderate polynomial degrees may generate huge data volumes and the result may suffer from unwanted/unexpected singular points. As an alternative, approximate techniques for generating implicit representations have emerged [Corless et al. 2001; Dokken 2001; Jüttler and Felis 2002; Jüttler et al. 2003; Sederberg et al. 1999]. Currently, the potential applications in industry (in particular addressing the issue of robustness of intersection algorithms) are under investigation in a European RTD project [Dokken et al. 2002–2005].

Exact *parameterization* of algebraic curves and surfaces is possible only for a very special subclass, which is characterized by a sufficiently high number of singularities. The zero contour $\{\mathbf{p} \in \mathbb{R}^2 \mid f(\mathbf{p}) = 0\}$ of a bivariate polynomial $f(\mathbf{p}) : \mathbb{R}^2 \to \mathbb{R}$ of degree n defines an algebraic curve of order n. This curves is said to be *rational* (or, equivalently, to have genus zero), if the number of singular points (counted with multiplicities) equals $\binom{n-1}{2}$. Such curves admit a rational parametric representation. It can be generated by considering an auxiliary one–parameter family of curves. Roughly speaking, the curves of this family are to pass through the singular points and a certain number of additional ones.

In the simplest case, $n = 2$, the auxiliary family of curves can be chosen as the pencil of lines passing through a point on the curve. The coordinates of the second intersection are rational functions of the parameter of the pencil. As a second example, we consider a general rational quartic curve. It is characterized by three singular points, see Figure 1. The rational quartic can be parameterized by generating the pencil of conics through four points on the quartic, three of them being the three double points. The conics intersect the quartic in 8 points, but 7 of them are already known to be the three singular points (each

*e-mail: bert.juettler@jku.at
†pavel.chalmoviansky@jku.at

of them counts twice) and the additional point. Consequently, the coordinates remaining intersection can be computed as a rational function of the parameter of the pencil, which leads to the rational parameterization.

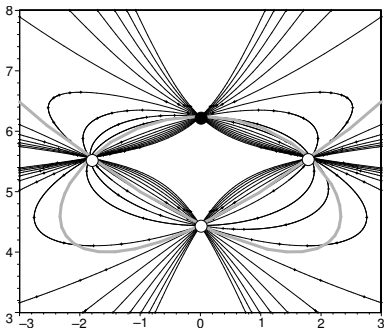

Figure 1: A pencil of conics (black) through the three double points (○) and one additional point (●) can be used for parameterizing a rational quartic curve (grey). In this example, the three double points are all real.

The general parameterization problem for rational surfaces was solved fairly recently [Schicho 1998]. In addition a number of approximate methods exist (see, e.g., [Bajaj and Xu 1997]), mostly producing polynomial parametric representations.

This paper is devoted to approximate parameterization of implicitly defined (not necessarily algebraic) planar curves by rational Bézier (spline) curves. As the main new feature, we actually use the degrees of freedom provided by the weights of the control points (i.e., by the denominator of the parametric representation). Consequently the method is able to give essentially exact results for curves of algebraic genus zero. The resulting non–linear optimization problem is solved efficiently by using sequential quadratic programming (SQP). This technique is described in more some detail in Section 3. In addition, we discuss a method for rational interpolation (Section 2), which provides additional motivation for our approach.

2 Rational interpolation

While interpolation by polynomial (spline) curves is well understood, the analogous problem for rational parametric representations has not received much attention. In the rational case, the weights of the control points (i.e., the denominator of the curve) provide some additional degrees of freedom, and it seems to be natural to use them for interpolating additional points.

An optimization–based approach has been taken in the Ph.D. thesis of Tae–Wan Kim [Kim 1996]: Given k points $\mathbf{p}_i$ with associated parameters t_i (which can be estimated from the data. e.g., using a chordal or centripetal parameterization), he constructs a rational Bézier or B–spline curve with the homogeneous coordinates

$$\bar{\mathbf{c}}(t) = (x(t), y(t), w(t)) = \sum_{i=0}^{d} B_i^d(t)(b_{i,1}, b_{i,2}, w_i), \quad (1)$$

where $B(t_i)$ are the Bernstein polynomials or, more general, B-splines. The interpolation conditions

$$\begin{array}{rcl} x(t_i) & = & w(t_i)p_{i,2} \\ y(t_i) & = & w(t_i)p_{i,2} \end{array}, \quad i = 1, \ldots k \quad (2)$$

lead to $2k$ homogeneous linear equations for the $3d+3$ unknown components of the control points. These components are then found by minimizing the objective function

$$\sum_{i=0}^{d} (w_i - 1)^2 \;\rightarrow\; \text{Min} \quad (3)$$

subject to the linear inequalities $w_i \geq \varepsilon$ and to the interpolation conditions (2), where ε with $0 < \varepsilon \ll 1$ is a user–defined constant.

The linear inequalities are needed in order to obtain a positive denominator. It is possible to derive weaker sufficient positivity conditions via subdivision.

The resulting quadratic programming problem can be solved efficiently, see [Fletcher 1990; Vanderbei 2004]. As an example, Figure 2 shows a cubic rational Bézier curve which has been generated by interpolating 5 points. This is one point more than the usual polynomial interpolation would be able to deal with. In this case, the interpolation conditions form a system of 10 homogeneous equations for 12 unknowns. The remaining two degrees of freedom are determined by minimizing (3).

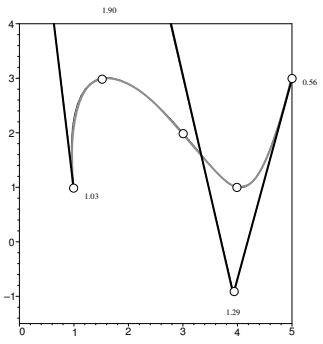

Figure 2: Interpolation of 4 points with equidistant parameters by a rational cubic Bézier curve. The weights of the control points are shown.

As a certain disadvantage of this approach, the existence of solutions cannot be guaranteed a priori for $k > d$, since it also depends on the location of the data. Only if $k = d$

holds, the problem reduces to the usual polynomial interpolation problems (where all weights equal 1), and the existence and uniqueness of solutions can be guaranteed by checking the Schoenberg–Whitney conditions $B(t_i) > 0$.

According to the experiences with this method and its predecessors [Schneider 1992], it is possible to actually use the weights of the rational curves, but one has to be very careful to restrict them to a reasonable interval. Otherwise, the curve may exhibit a very non–uniform parametric speed distribution, or it may even have points at infinity.

Remark. In many cases, it suffices to minimize (3) without taking the linear inequalities into account. Afterwards one may check whether the weights are positive, and additional degrees of freedom can be introduced as needed.

3 Approximate parameterization

We consider the problem of parameterizing an algebraic curve. Note that the results presented here can be generalized to any implicitly defined curves.

3.1 Outline

Consider a bivariate polynomial $f = f(\mathbf{p})$ of degree m. Let

$$\mathscr{C} = \{\mathbf{p} \in \mathbb{R}^2 \colon f(\mathbf{p}) = 0\} \tag{4}$$

be the algebraic curve which is defined by this polynomial. We are interested in a regular arc of $\mathscr{C}$ connecting two points $\mathbf{a}_0, \mathbf{a}_1 \in \mathscr{C}$. We construct a (possibly piecewise) rational parametric curve of degree n

$$\bar{\mathbf{c}}(t) = (x(t), y(t), w(t)) \quad \text{for} \quad t \in [0,1] \tag{5}$$

which interpolates the points $\mathbf{a}_0$, $\mathbf{a}_1$ and the tangents at these points. In addition, the curve is to approximate the algebraic curve $\mathscr{C}$ with a certain accuracy along the arc. The method consists of the following steps:

1. We generate a suitable initial solution. This is done by using simple cubic Hermite interpolation for a small segment.
2. Using nonlinear optimization, we generate a highly accurate rational approximation to the arc of the algebraic curve.
3. The segment is extrapolated, in order to find a new initial solution for a bigger segment.
4. Steps 2 and 3 are repeated, until the segment can no longer be extended (this decision is governed by the maximum deviation).
5. If the extension in Step 4 fails, we start a new segment, which is attached to the previous one with G^1 continuity.

By using this region–growing type approach, we can always be sure to have a sensible initial solution for the next non–linear iteration.

Each segment of the approximating curve is described in Bernstein-Bézier form, with the homogeneous coordinates

$$\bar{\mathbf{c}}(t) = \sum_{i=0}^{n} B_i^n(t)(b_i, b_{n+1+i}, b_{2n+2+i}). \tag{6}$$

The vector $\mathbf{b} = (b_0, \ldots, b_{d_n})$ of control values with $d_n = 3n+2$ represents our curve as a point in $\mathbb{R}^{d_n+1}$ space. The vectors

$$\mathbf{p}_i = (b_i, b_{n+1+i}, b_{2n+2+i}) \tag{7}$$

are the homogeneous control points of the rational Bézier curve $\mathbf{c}(t)$ for $i = 0, \ldots, n$.

In order to obtain the Cartesian coordinates, we apply a projection of $\bar{\mathbf{c}}(t)$ to the plane $w = 1$,

$$\mathbf{c}(t) = \left(\frac{x(t)}{w(t)}, \frac{y(t)}{w(t)} \right) \quad \text{for} \quad t \in [0,1]. \tag{8}$$

As a well–known fact, rational Bézier curves admit bilinear parameter transformations, which lead to a one–parameter family of equivalent representations. This leads to the so–called *standard form*, where the boundary weights b_{2n+2} and b_{3n+2} are both equal to one. See [Hoschek and Lasser 1993] for more information. Throughout this paper, we consider rational curves in standard form.

3.2 Parameterizing a segment

We generate a rational curve segment $\mathbf{c}$ which approximates the algebraic curve,

$$f(\mathbf{c}(t)) \approx 0 \quad \text{for} \quad t \in [0,1] \tag{9}$$

and satisfies the boundary conditions

$$\mathbf{c}(0) = \mathbf{a}_0 \tag{10}$$

$$\mathbf{c}(1) = \mathbf{a}_1 \tag{11}$$

$$\nabla f(\mathbf{a}_0) \cdot \dot{\mathbf{c}}(0) = 0 \tag{12}$$

$$\nabla f(\mathbf{a}_1) \cdot \dot{\mathbf{c}}(1) = 0. \tag{13}$$

In order to satisfy the G^1 boundary conditions, we define suitable unit tangent vectors $\vec{\mathbf{v}}_0$ and $\vec{\mathbf{v}}_1$ at the end points, see Figure 3, left.

We use optimization techniques in order to find the solution. If no initial solution is known, then we start with a

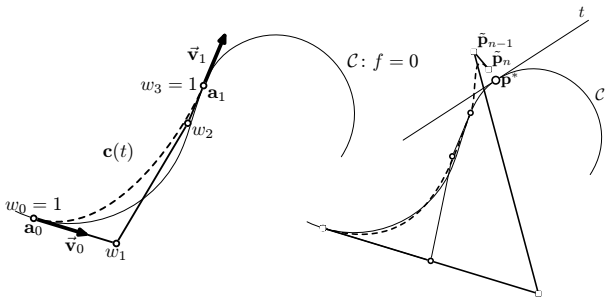

Figure 3: Left: Constructing the initial solution. Right: Extending a segment

cubic Hermite segment interpolating the given endpoint conditions. In case $n > 3$, degree raising is needed in order to get the higher degree initial curve.

The solution is computed by minimizing the objective function $F(\mathbf{b})$ subject to G^1 boundary conditions,

$$F(\mathbf{b}) = F_1(\mathbf{b}) + w_{\text{weight}} F_2(\mathbf{b}) + w_{\text{cp}} F_3(\mathbf{b}), \tag{14}$$

with

$$F_1(\mathbf{b}) = \int_0^1 \frac{f(\mathbf{c}(t))^2}{\|\nabla f(\mathbf{c}(t))\|^2} \mathrm{d}t, \tag{15}$$

$$F_2(\mathbf{b}) = \int_0^1 (w(t)-1)^8 \mathrm{d}t \tag{16}$$

$$F_3(\mathbf{b}) = \sum_{i=0}^{n-1} \|\mathbf{p}_{i+1} - \mathbf{p}_i\|^2, \tag{17}$$

where the weights w_{weight}, w_{cp} satisfy

$$0 < w_{\text{cp}} \ll w_{\text{weight}} \ll 1. \tag{18}$$

The term F_1 is responsible for minimizing the distance of the curve (8) from the curve $\mathscr{C}$.

If the lengths of the gradients of f do not vary too much in the vicinity of the curve, one may use the simpler objective function

$$F_1^*(\mathbf{b}) = \int_0^1 \bar{f}(\bar{\mathbf{c}}(t))^2 \mathrm{d}t, \tag{19}$$

instead of F_1, where $\bar{f}$ is the homogenized version of the polynomial $f(x,y)$ describing the given curve, and $\bar{\mathbf{c}}$ are the homogeneous coordinates. As an advantage, this leads to *integrals of polynomials*, which can be evaluated *exactly* (e.g., by using Gaussian quadrature). Similarly, all derivatives needed for the numerical optimization can then be evaluated exactly.

As another, more sophisticated, objective function, one could integrate the distance error estimate $(f(\mathbf{c}(t))^2)/(\|\nabla f(\mathbf{c}(t))\|^2)$ over the arc length of the curve.

The term F_2 penalizes the deviation of the weights of the rational curve (i.e., of the denominator) from one, making the curve as close to a polynomial one as possible. Clearly, the eighth power could be replaced with any even number. According to our numerical experience, the eighth power produces a reasonable weight distribution. The last term F_3 serves to regularize the solution.

Since F_1 and F_2 are non-linear, we use an SQP-type method to compute a stationary point of the objective function.

Let $\mathbf{b}$ be the vector of control points for the curve to be optimized. We compute

$$\frac{\partial F(\mathbf{b})}{\partial b_i} \quad \text{and} \quad \frac{\partial^2 F(\mathbf{b})}{\partial b_i \partial b_j} \tag{20}$$

for $i, j = 0, \ldots, d_n$. Using the auxiliary vectors $\vec{\mathbf{v}}_0, \vec{\mathbf{v}}_1$, the boundary conditions can be rewritten as

$$\mathbf{p}_0 = \mathbf{a}_0 \tag{21}$$

$$\mathbf{p}_1 = b_{2n+3}\mathbf{p}_0 + k_0 \frac{1}{n}(\vec{\mathbf{v}}_0, 0) \tag{22}$$

$$\mathbf{p}_n = \mathbf{a}_1 \tag{23}$$

$$\mathbf{p}_{n-1} = b_{3n+2}\mathbf{p}_n - k_1 \frac{1}{n}(\vec{\mathbf{v}}_1, 0). \tag{24}$$

Hence, we will optimize with respect to the parameters

$$b_2, \ldots, b_{n-2}, b_{n+3}, \ldots, b_{2n-1}, b_{2n+3}, \ldots, b_{3n+1}, k_0, k_1 \tag{25}$$

Since (21), (22), (23) and (24) represent a linear change of coordinates, we can calculate the gradient of F and the matrix of second order derivatives with respect to variables (25) easily from (20). More precisely, consider the vector $\mathbf{r}$ of the new variables (25). Then, let

$$\mathbf{r} = A\mathbf{b} \tag{26}$$

be a linear change of coordinates given by the matrix A of type $(3n-1) \times (3n+3)$. Then

$$\nabla_{\mathbf{r}} F = A \nabla_{\mathbf{b}} F \quad \text{and} \quad \nabla^2_{\mathbf{r}} F = A \nabla^2_{\mathbf{b}} F A^\top \tag{27}$$

If the curve $\mathbf{r}$ does not represent a stationary point of the functional F, we calculate the vector for correcting the current values from

$$\vec{\mathbf{q}} = (\nabla^2_{\mathbf{r}} F)^{-1} \nabla F_{\mathbf{r}}. \tag{28}$$

In order to make the algorithm more stable, we use a simple bisection procedure to adjust the stepsize. More precisely, according to the sign of the directional derivative $\nabla_{\mathbf{r}} F \cdot \vec{\mathbf{q}}$, we compute the new point $\mathbf{r}^*$ by bisection between points

$$\mathbf{r} \quad \text{and} \quad \mathbf{r} - \operatorname{sgn}(\nabla_{\mathbf{r}} F \cdot \vec{\mathbf{q}}) \vec{\mathbf{q}}. \tag{29}$$

We stop the bisection if $F(\mathbf{r}^*) < F(\mathbf{r})$ holds. This produces a sequence of curves satisfying

$$F(\mathbf{b}^{i+1}) > F(\mathbf{b}^i) \tag{30}$$

If $\mathbf{r}^*$ has been found, we get $\mathbf{b}^*$ via (26).

Starting with $\mathbf{b}^0$ representing a Hermite interpolant of the boundary conditions, the iteration step generates a sequence of curves

$$\{\mathbf{b}^i\}_{i=0}^{\infty} \quad \text{with} \quad \mathbf{b}^{i+1} = (\mathbf{b}^i)^*. \tag{31}$$

If the initial value $\mathbf{b}^0$ is within an appropriate neighborhood of a local minimum, then the sequence is known to converge towards this minimum.

Remark. In practice, the adjustment of the stepsize should also take the condition $k_0^*, k_1^* > 0$ into account, which is introduced in order to preserve the direction of the derivatives at the endpoints.

3.3 Extending the segment

After generating a curve segment, we extrapolate this along the algebraic curve $\mathscr{C}$, in order to get the initial value for the next step. Since all operations are in a neighborhood of the endpoint of the curve segment, we will calculate with corresponding Euclidean coordinates of the control points $\mathbf{p}_i$. The algorithm works as follows:

1. Let $\varepsilon > 0$. Calculate the control points of the curve $\bar{\mathbf{c}}$ on interval $[0, 1+\varepsilon]$. This can be done via de Casteljau algorithm. Let $\tilde{\mathbf{p}}_i$ for $i = 0, \ldots, n$ be the control points of the extended Bézier curve.
2. Project the endpoint $\tilde{\mathbf{p}}_n$ of the extension on the curve $\mathscr{C}$, see Figure 3, right. Let $\mathbf{p}^*$ be the result of the projection and calculate the new tangent at this endpoint of the segment as

   $$t\colon \nabla f(\mathbf{p}^*) \cdot (\mathbf{x} - \mathbf{p}^*) = 0. \tag{32}$$

   Hence, the new endpoint is shifted to $\mathbf{a}_1^* = \mathbf{p}^*$. In order to satisfy the G^1 endpoint conditions, we project also the tangent vector $k_1\vec{\mathbf{v}}_1$ to the endpoint tangent t of the new segment. We get a new tangent vector $k_1^*\vec{\mathbf{v}}_1^*$, where we suppose $\vec{\mathbf{v}}_1^*$ to be a unit vector. Its orientation is discussed later.
3. We reparameterize the new curve segment so that the weights on the endpoints are equal to 1.0 and the new segment is again parameterized over the interval $[0, 1]$ (conversion to standard form).

The extension parameter ε is chosen as follows: we start with $\varepsilon = \varepsilon_0$ (in our examples $\varepsilon_0 = 0.5$). We apply bisection until the following conditions are satisfied:

- The projection $\mathbf{p}^*$ of $\tilde{\mathbf{p}}_n$ to $\mathscr{C}$ exists and

  $$\|\mathbf{p}^* - \tilde{\mathbf{p}}_n\| < \max_{\text{pd}} \tag{33}$$

- The endpoint tangent direction satisfies

  $$\vec{\mathbf{v}}_1^* \cdot \vec{\mathbf{v}}_1 > \max_{\cos} \quad \text{and} \quad k_1^* > 0 \tag{34}$$

- If $\mathbf{b}$ represents the initial curve and $\tilde{\mathbf{b}}$ represents an extension of the initial curve, we require

  $$F(\tilde{\mathbf{b}}) < \max_{\text{OF}} \quad \text{and} \quad F(\tilde{\mathbf{b}}) < c_{\max} F(\mathbf{b}) \tag{35}$$

  for some positive coefficients with $c_{\max} > 1.0$.

Clearly, this procedure involves some user–defined constants, which have to be chosen carefully.

If the conditions (33), (34) and (35) are satisfied for reasonable extension ε parameter, we use again the optimization procedure from the previous section, in order to approximate the given algebraic curve.

Repeating these steps, we can considerably extend the first optimized curve segment in one or both parameter directions.

3.4 Applications and Examples

A first example of the algorithm for approximate parameterization is shown in Figure 4. We started with the cubic initial segment in (a). The control polygon is drawn as a thick line. The weights of all control points are equal to 1.0, as can be seen in the figure. After the optimization we got the segment on the part (b) and the weights of the middle points decreased to 0.86 and 0.84. Note that these figure (and all other ones) show the algebraic curve, the rational Bézier curve (for a segment larger than the domain $[0,1]$) and its control polygon. The first feasible extension is shown in (c), and (d) visualizes the situation after the projection of the endpoint and its tangent vector to the curve $\mathscr{C}$. The new segment was again optimized (e), and another extension is shown in (f). As demonstrated by this example, the method is able to automatically adjust the weights of the control points.

In the second example (Figure 5), we applied the method to parameterizing a circle. Since the circle is a rational curve, our approach is expected to give an almost exact solution in the limit. Clearly, if the terms F_2 and F_3 and used in the objective function, then the algorithm will not be able to reproduce the circle. Still, it produces a highly accurate approximation to it.

As a third example (see Figure 6), we have applied the method to an algebraic curve with a singular point. As demonstrated by this example, it is possible to pass through the singular point and continue along the branch of after the singularity. However, a more thorough analysis of the convergence in this case is still needed.

As an application, we apply this methods to the interpolation problem. Consider 9 points Let $\mathbf{p}_1, \ldots, \mathbf{p}_9 \in \mathbb{R}^2$ in

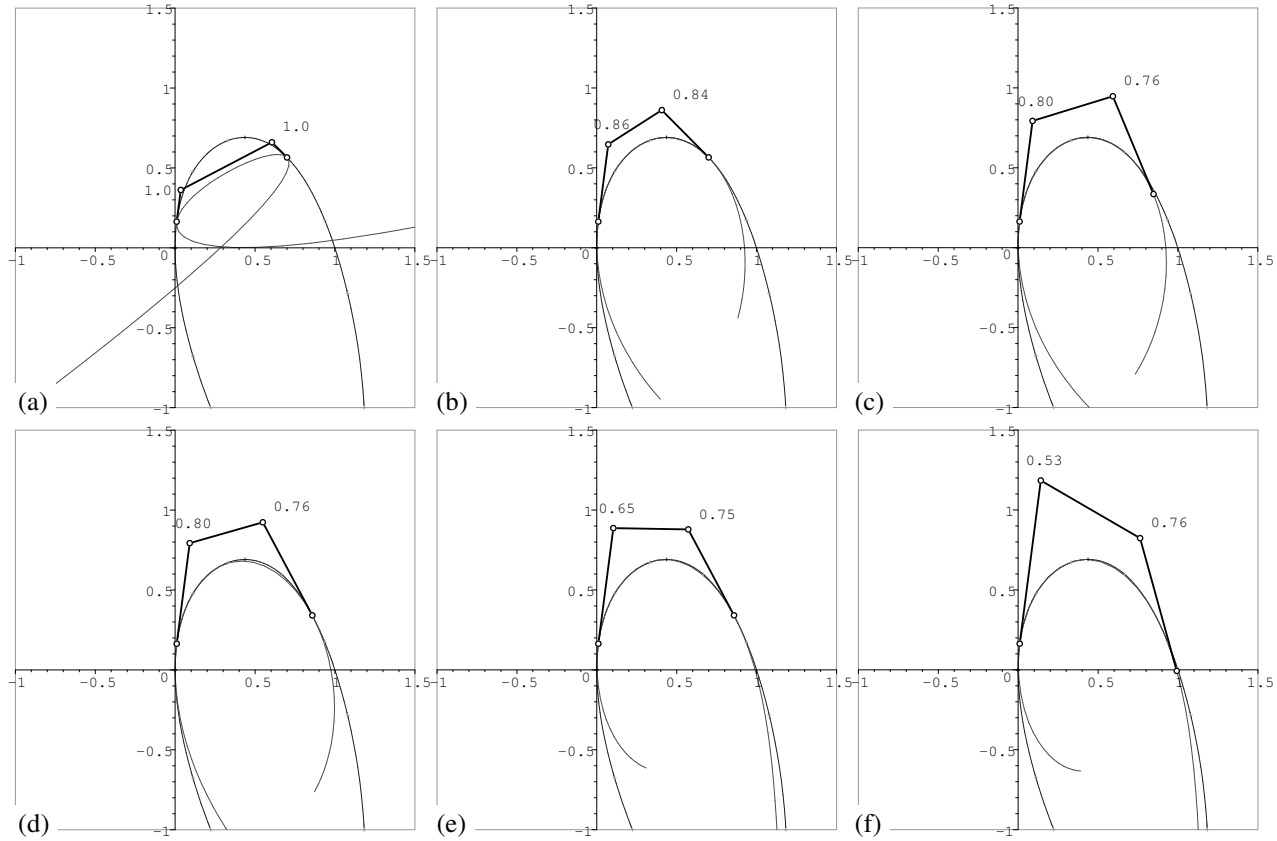

Figure 4: Parameterizing a cubic algebraic curve: (a) initial cubic Hermite curve; (b) optimized segment; (c) extrapolated segment; (d) endpoint projected to the implicit curve; (e) optimization of the extended segment; (f) next extension.

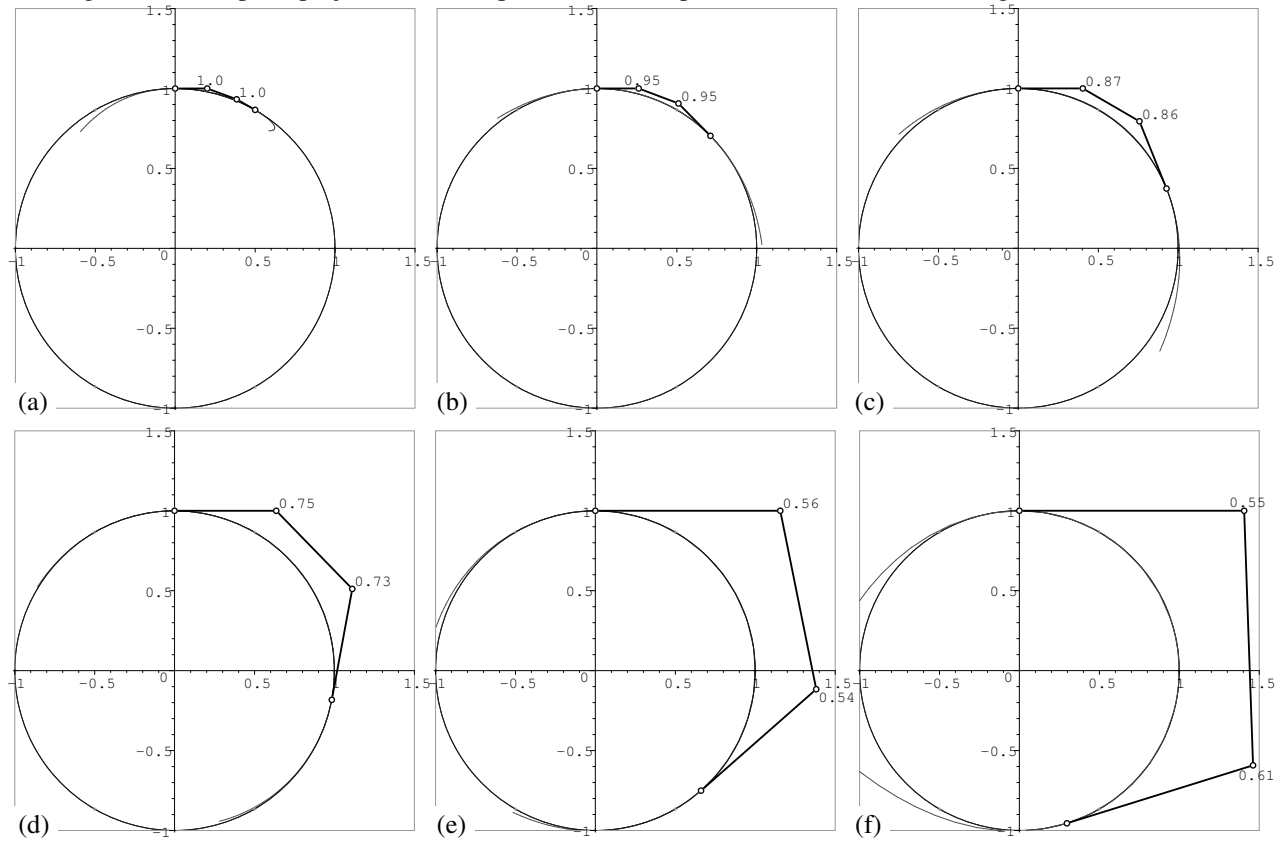

Figure 5: Parameterizing a circle: (a) initial cubic Hermite curve; (b–f) several steps of the curve extension.

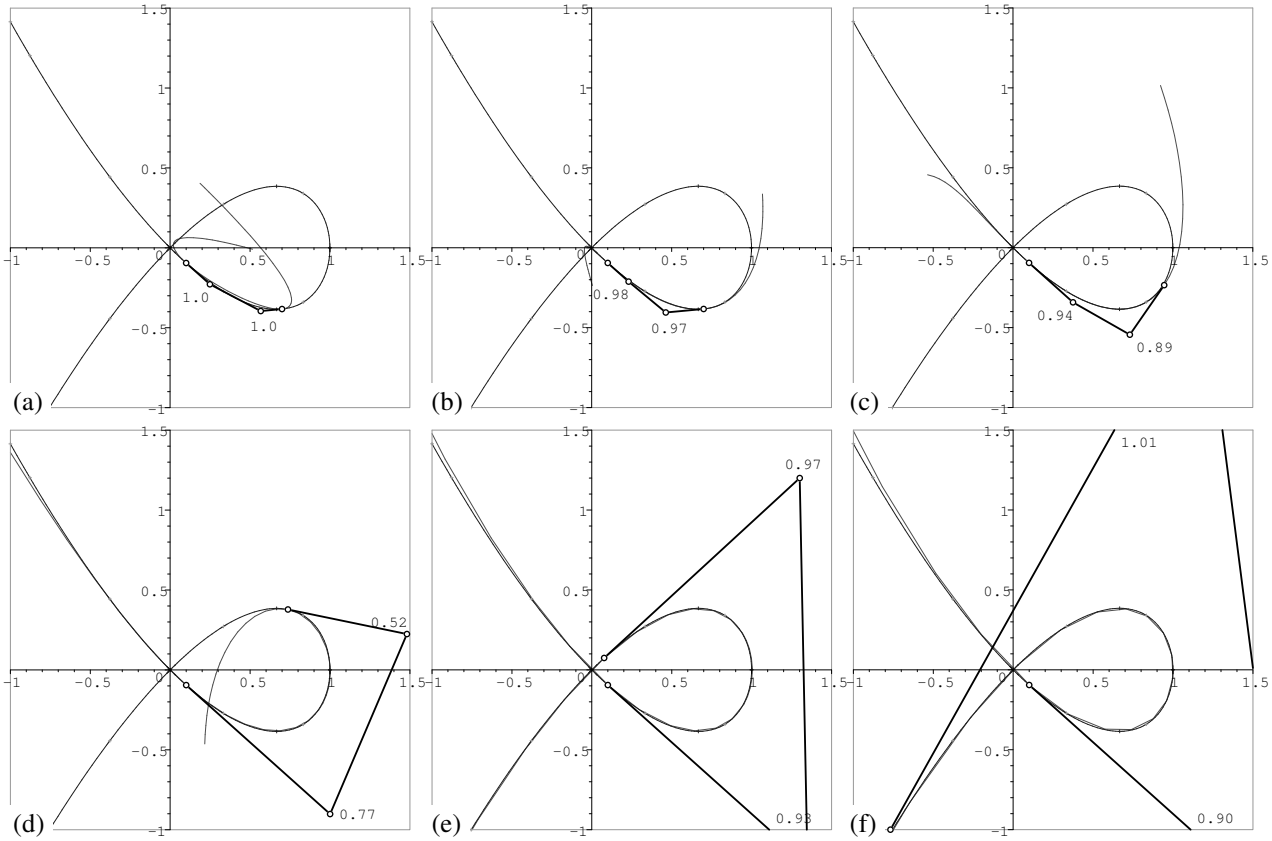

Figure 6: Example of parameterization of implicitized cubic curve with s a singular point: (a) initial cubic Hermite curve; (b)-(f) several steps of the extension.

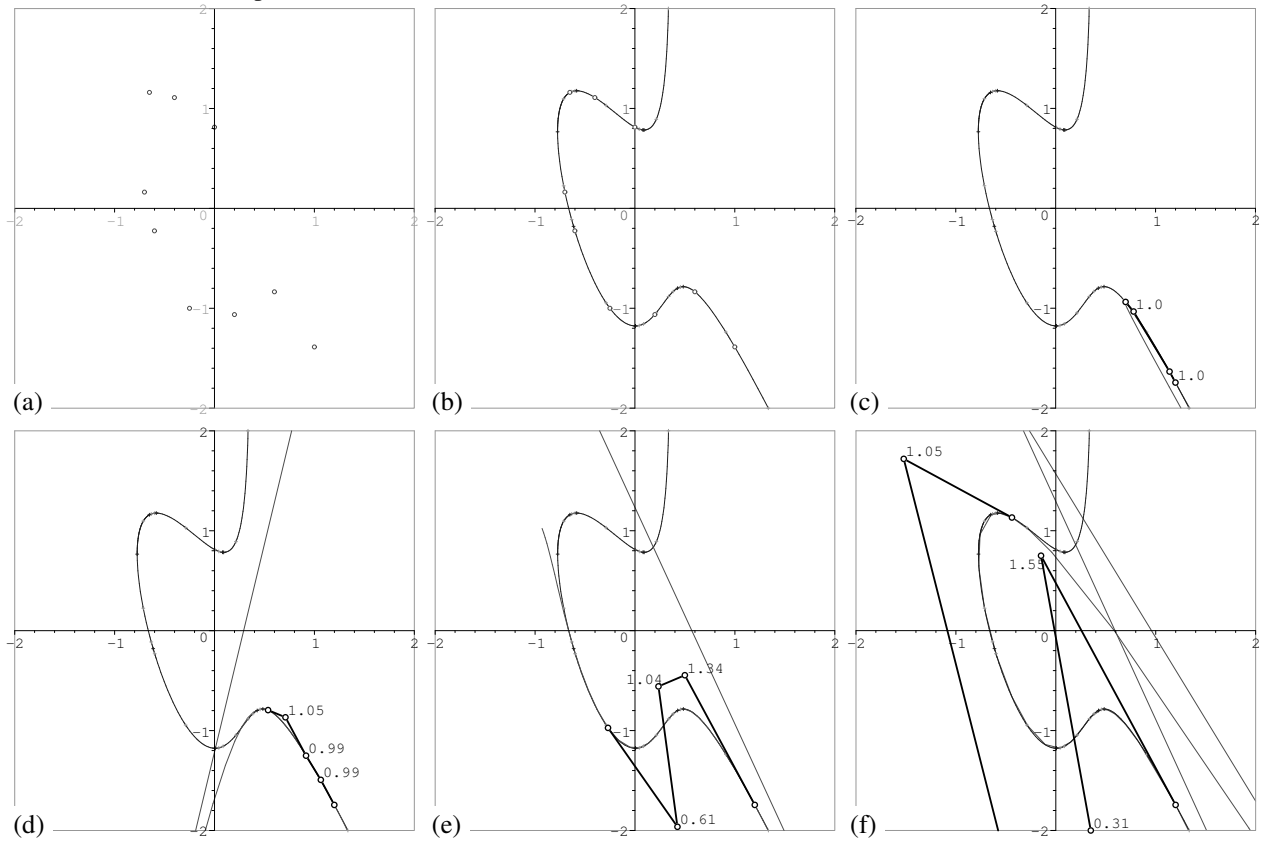

Figure 7: Parameterization of non–rational curve. (b) cubic algebraic curve which interpolates the 9 points in (a); (c) cubic Hermite curve, (d)-(f) several steps of the extension

the plane. These points can be interpolated by a cubic algebraic curve $f = 0$, where

$$f(x,y) = \sum_{i+j+k=3} B^3_{ijk}(x,y,1-x-y)b_{ijk}. \qquad (36)$$

More precisely, the system of equations

$$f(\mathbf{p}_i) = 0 \quad \text{for} \quad i = 1,\ldots,9 \qquad (37)$$

has – in the generic case – a unique solution (up to a constant factors). Using our approximate parameterization technique, we can get the parameterization of the approximation. The resulting implicit curve can then be approximately parameterized. This method has been used to generate the curve shown in Figure 7.

On the one hand, this method can be seen as a possible approach to bypass the parameterization problem for curve fitting (and similarly for surfaces). On the other hand, the algebraic fit has to be carefully examined; for instance, the given points may end up on different branches of the algebraic curve, or the curve may exhibit unwanted singularities. This problem can be resolved by using techniques such as the one described in [Jüttler and Wurm 2003], which produce algebraic spline curves approximating both a sequence of points and associated gradient information.

4 Conclusions

We have presented a method for approximate parameterization of an implicitly defined curve. The method, which is based on non–linear optimization, produces a piecewise rational Bézier curve. Currently, our implementation leads to G^1 curves, but the generalization to higher orders of differentiability is possible. As the main new feature, the method is able to adjust the weights of the control points, in order to obtain the best possible result. Future research will concentrate on applications, e.g., for tracing surface/surface–intersections, and on the generalization to the case of surfaces.

Acknowledgements The research of the first author was partly supported by the Fifth Framework Programme of the EU via project GAIA II (IST 2001-35512). The second author gratefully acknowledges the support by the Austrian Science Fund (FWF) through the Special research programme (SFB) F013 at Linz.

References

BAJAJ, C., AND ROYAPPA, A. 2000. Parameterization in finite precision. *Algorithmica 27*, 1, 100–114.

BAJAJ, C., AND XU, G. 1997. Spline approximations of real algebraic surfaces. *J. Symb. Comput. 23*, 315–333.

BLOOMENTHAL, J., Ed. 1997. *Introduction to implicit surfaces*. Morgan Kaufmann, San Francisco.

CORLESS, R., GIESBRECHT, M., KOTSIREAS, I., AND WATT, S. 2001. Numerical implicitization of parametric hypersurfaces with linear algebra. In *AISC'2000 Proceedings*, Springer, vol. 1930 of *LNAI*.

COX, D., LITTLE, J., AND O'SHEA, D. 1997. *Ideals, varieties and algorithms*. Springer, New York.

DOKKEN, T., GALLIGO, A., GONZALEZ-VEGA, L., JÜTTLER, B., MASSABO, A., AND PIENE, R., 2002–2005. Intersection algorithms for geometry based it–aplications using approximate algebraic methods. EU Project IST 2001–35512 (5th framework programme).

DOKKEN, T. 2001. Approximate implicitization. In *Math. methods for curves and surfaces*, T. Lyche et al., eds. Vanderbilt University Press, Nashville, 81–102.

FLETCHER, R. 1990. *Practical methods of optimization*. Wiley, Chichester.

HOSCHEK, J., AND LASSER, D. 1993. *Fundamentals of Computer Aided Geometric Design*. AK Peters, Wellesley Mass.

JÜTTLER, B., AND FELIS, A. 2002. Least–squares fitting of algebraic spline surfaces. *Advances in Computational Mathematics 17*, 135–152.

JÜTTLER, B., AND WURM, E. 2003. Approximate implicitization via curve fitting. In *Symposium on geometry processing*, Eurographics / ACM Siggraph, L. Kobbelt, P. Schröder, and H. Hoppe, Eds., 240–247.

JÜTTLER, B., SCHICHO, J., AND SHALABY, M. 2003. Spline implicitization of planar curves. In *Curve and Surface Fitting: St. Malo 2002*, Nashboro Press, Brentwood, T. Lyche et al, eds., 225–234.

KIM, T.-W. 1996. *Design of rational curves and developable surfaces*. PhD thesis, Arizona State University, Tempe. available from UMI Dissertation Services, Ann Arbor MI.

SCHICHO, J. 1998. Rational parametrization of surfaces. *J. Symb. Comp. 26*, 1, 1–30.

SCHNEIDER, F.-J. 1992. *Interpolation, Approximation und Konvertierung mit rationalen B-Spline Kurven und Flächen*. PhD thesis, TH Darmstadt. Published at Shaker–Verlag, Aachen (www.shaker.de).

SEDERBERG, T., AND CHEN, F. 1995. Implicitization using moving curves and surfaces. *Computer Graphics 29*, 301–308. SIGGRAPH'95.

SEDERBERG, T., ZHENG, J., KLIMASZEWSKI, K., AND DOKKEN, T. 1999. Approximate implicitization using monoid curves and surfaces. *Graphical Models and Image Processing 61*, 177–198.

VANDERBEI, R., 2004. LOQO. www.princeton.edu/~rvdb. A quadratic programming solver.