Fast Approximate Implicitization of Envelope Curves using Chebyshev Polynomials

Oliver J. D. Barrowclough, Bert Jüttler, and Tino Schulz

Abstract Consider a rational family of planar rational curves in a certain region of interest. We are interested in finding an approximation to the implicit representation of the envelope. Since exact implicitization methods tend to be very costly, we employ an adaptation of approximate implicitization to envelope computation. Moreover, by utilizing an orthogonal basis in the construction process, the computational times can be shortened and the numerical condition improved. We provide an example to illustrate the performance of our approach.

Key words: implicitization, approximation, envelopes, Chebyshev polynomials.

1 Introduction

In geometric applications there are two basic standards for representing curves, namely the *parametric* and the *implicit* descriptions. Both descriptions feature specific advantages and disadvantages that complement each other. For instance, parametric curves allow the simple generation of point samples, while implicit forms support the decision of point location queries. In many applications, such as intersection computations, it is an advantage if both representations are available, and conversion algorithms are therefore of substantial practical interest. The conversion processes are called *parametrization* and *implicitization*, respectively.

Oliver J. D. Barrowclough
SINTEF ICT, Applied Mathematics, P.O. Box 124, Blindern, 0314 Oslo, Norway e-mail: oliver.barrowclough@sintef.no
Bert Jüttler
Johannes Kepler University, Institute of Applied Geometry, Altenberger Str. 69, A-4040 Linz, Austria e-mail: bert.juettler@jku.at
Tino Schulz
GALAAD, INRIA Méditerranée, 2004 Route des Lucioles, 06902 Sophia-Antipolis, France e-mail: tino.schulz@inria.fr

A rational curve may always be implicitized, whereas the opposite is not true [10]. Several techniques for *exact* implicitization exist, e.g. Gröbner bases, moving curves/surfaces, or methods based on resultants, (see e.g. [5]). However, due to their computational complexity, their practical use is often restricted to low-degree curves. Moreover, the variety obtained by exact implicitization may contain unexpected branches and self-intersections.

A valid alternative to exact methods is *approximate implicitization*; cf. [3, 4]. Instead of the exact variety, a low degree approximation is used to represent the shape of the geometric object in a certain region of interest. This technique can be implemented using floating point numbers and thus it offers faster computation, while having very high convergence rates. As shown in [2], the speed-up may be increased even further by using an orthogonal basis in the construction process.

These features make approximate implicitization a promising candidate for an efficient computation of *envelope curves*. Envelopes are used in different contexts in mathematics and applications, ranging from gearing theory and geometrical optics to NC-machining and Computer-Aided Design. In robotics, envelopes are ubiquitous, appearing for instance as singularities or boundaries. The theory of envelopes is covered by the classical literature, and is continuously extended, due to their practical importance [1, 6, 7, 8].

Approximate implicitization has recently been adapted to the computation of envelopes in [9]. As shown there, the idea is feasible and most properties of the original method can be preserved, such as the possibility of obtaining the exact solution. However, the convergence behaviour for higher degrees has not previously been studied and the computations are still fairly expensive, needing integrals of products of high degree polynomials.

The present paper uses the latest results from approximate implicitization to obtain a fast and efficient algorithm for approximating the envelope. This will make the use of implicit methods more attractive and moreover allow us to study the convergence behaviour experimentally. The paper is organized as follows: In Section two we will recall the basics of envelopes of planar curves. After that, the third section shows how approximate implicitization can be used to compute envelopes and derives a fast and efficient algorithm. The performance of our approach is illustrated with an example and discussion in Section four.

2 Envelopes of Rational Families of Curves

Consider the family of rational curves

$$\mathbf{p}(s,t) = \big(x(s,t)/w(s,t), y(s,t)/w(s,t)\big)^T, \quad (s,t) \in I \times J \tag{1}$$

where x, y and w are bivariate polynomials of bidegree (n_1, n_2) with $\gcd(x,y,w) = 1$ and $I, J \subset \mathbb{R}$ are closed intervals. We assume that $w(s,t) \neq 0$ for all $(s,t) \in I \times J$.

Either s or t can be thought of as the time-like parameters, and the remaining parameter t or s is then used to parameterize the curves forming the family.

The *envelope* of the mapping $\mathbf{p}$ consists of those points where its Jacobian $\mathbf{J}(s,t)$ becomes singular. We observe that $\det \mathbf{J}(s,t) = h(s,t)/w(s,t)^3$, where

$$h(s,t) = \det \begin{pmatrix} x(s,t) & \partial_s x(s,t) & \partial_t x(s,t) \\ y(s,t) & \partial_s y(s,t) & \partial_t y(s,t) \\ w(s,t) & \partial_s w(s,t) & \partial_t w(s,t) \end{pmatrix}. \tag{2}$$

The function h is called the *envelope function*, since its zero set determines those points in the parameter domain which are mapped to the envelope. Unfortunately, certain parts of the zero set of h may degenerate under the mapping $\mathbf{p}$.

The earlier paper [9] describes how these *improper factors* can be removed from h. This can be done via some gcd computation and gives the *reduced envelope function* $\tilde{h}$. The image of the zero set of $\tilde{h}$ under $\mathbf{p}$ is called *proper part* of the envelope.

Let $q : \mathbb{C}^2 \to \mathbb{C}$ be the polynomial which defines *the implicit equation of the proper part of the envelope* of $\mathbf{p}$. According to Theorem 1 of [9], there exists a real polynomial $\lambda(s,t)$ such that

$$(q \circ \mathbf{p})(s,t) w(s,t)^d = \lambda(s,t) \tilde{h}(s,t)^2. \tag{3}$$

Equation (3) is linear with respect to the coefficients of q and λ. Let

$$q(\mathbf{x}) = \mathbf{c}_q^T \beta(\mathbf{x}) \quad \text{and} \quad \lambda(s,t) = \mathbf{c}_\lambda^T \alpha(s,t), \tag{4}$$

where $\beta(\mathbf{x}) = (\beta_k(\mathbf{x}))_{k=1}^M$ and $\alpha(s,t) = (\alpha_i(s)\alpha_j(t))_{(i,j)=(0,0)}^{(k_1,k_2)}$ are bases of polynomials in $\mathbf{x}$ and s,t of total degree d and bidegree (k_1,k_2) respectively, where $M = \binom{d+2}{2}$. The coefficients of q and λ with respect to these bases form a vector $\mathbf{c} = (\mathbf{c}_q^T, \mathbf{c}_\lambda^T)^T$. We formulate the *problem of approximate envelope implicitization:* Find the coefficients $\mathbf{c}$ which solve the weighted least squares minimization problem

$$\min_{\|\mathbf{c}\|_2=1} \int_{I\times J} \omega(s,t) \left((q\circ\mathbf{p})(s,t) w(s,t)^d - \lambda(s,t) h(s,t)^2 \right)^2 \mathrm{d}(s,t), \tag{5}$$

for a nonnegative weight function ω, and chosen degrees d, k_1 and k_2.

It is important to mention that we use h instead of the exact $\tilde{h}$, since our algorithm uses floating point computations which do not support exact gcd computations. While an exact solution of this simplified problem might produce additional branches, the effect on our low degree approximation will be negligible.

The result of the minimization (5) depends both on the choice of bases of q and λ and on the weight function ω. The standard choice of a triangular Bernstein basis for q and a tensor-product Bernstein basis for λ has been used for the approximations in this paper and also in [9]. However, as a major difference to the approach in [9] where $\omega \equiv 1$, here we use a tensor product Chebyshev weight function on the domain $I \times J$, for the reasons described in the next section.

3 Fast Approximate Implicitization of Envelope Curves

The direct method for finding an approximate implicitization of envelope curves by evaluating high degree integrals is simple, but computationally costly. In addition, the resulting symmetric positive semi-definite matrix can be rather ill conditioned, leading to inaccurate null space computations when using floating point arithmetic. This is similar to the case of approximate implicitization of parametric curves presented in [2]. In that paper, an approach using orthogonal polynomials is presented which greatly improves both the conditioning and the computation time of the problem. In this section we give the details of how to implement the approach to approximate implicitization of envelope curves using Chebyshev polynomials.

3.1 Approximate Implicitization using Chebyshev Polynomials

As described previously, the method works by minimization of the integral (5). Such a problem is aided by expressing the function in a basis orthonormal with respect to the chosen weight function ω. The objective function is expressible in any tensor product polynomial basis of bidegree

$$(L_1, L_2) = (\max(dn_1, k_1 + 2\deg_s(h)), \max(dn_2, k_2 + 2\deg_t(h))).$$

Thus, choosing an orthonormal basis (e.g., tensor-product Chebyshev polynomials), $\mathbf{T}(s,t) = (T_i(s)T_j(t))_{i=0,j=0}^{L_1,L_2}$ written in vector form and using (4), we can write

$$(q \circ \mathbf{p})(s,t)w(s,t)^d - \lambda(s,t)h(s,t)^2 = \mathbf{T}(s,t)^T(\mathbf{D}_q\mathbf{c}_q + \mathbf{D}_\lambda\mathbf{c}_\lambda), \tag{6}$$

where the matrices $\mathbf{D}_q$ and $\mathbf{D}_\lambda$ contain coefficients in $\mathbf{T}$. Now, defining a matrix

$$\mathbf{D} = (\mathbf{D}_q, \mathbf{D}_\lambda), \tag{7}$$

we claim that the singular vector corresponding to the smallest singular value of $\mathbf{D}$ solves the minimization problem (5). To see this, we prove the following Theorem:

Theorem 1. *Let the matrix* $\mathbf{D}$ *be defined as in (7). Then we have*

$$\min_{\|\mathbf{c}\|_2=1} \int_{I\times J} \omega((q\circ\mathbf{p})w^d - \lambda h^2)^2 = \min_{\|\mathbf{c}\|_2=1} \|\mathbf{Dc}\|_2^2.$$

Proof. By (6) and (7) we have

$$\int_{I\times J} \omega((q\circ\mathbf{p})w^d - \lambda h^2)^2 = \int_{I\times J} \omega(\mathbf{c}^T\mathbf{D}^T\mathbf{T})(\mathbf{T}^T\mathbf{Dc}) =$$

$$\mathbf{c}^T\mathbf{D}^T\left(\int_{I\times J}\omega\mathbf{T}\mathbf{T}^T\right)\mathbf{Dc} = \mathbf{c}^T\mathbf{D}^T\mathbf{Dc} = \|\mathbf{Dc}\|_2^2. \quad \square$$

Since we have $\min_{\|\mathbf{c}\|_2=1} \|\mathbf{Dc}\|_2 = \sigma_{\min}$, where $\sigma_{\min}$ is the smallest singular value of $\mathbf{D}$, the corresponding right singular vector solves the problem. The problem is, however, better conditioned and can be implemented in a more efficient way than the weak approach [2].

3.2 Implementation of the Chebyshev method

The choice of using Chebyshev polynomials for the orthogonal basis is made mainly for computational reasons; the coefficients can be generated via a fast algorithm. This utilizes an existing method outlined for univariate polynomials in [11], which exploits the discrete orthogonality of Chebyshev polynomials at Chebyshev points.

Here we briefly describe the algorithm for efficient generation of tensor-product Chebyshev coefficients. The univariate Chebyshev points of degree L in the interval $[0,1]$ are given by:

$$t_{j,L} = \left(1-\cos\left(j\pi/L\right)\right)/2, \quad j = 0,\ldots,L.$$

The Chebyshev coefficients of any tensor product polynomial f of bidegree no higher than (L_1,L_2) can then be generated by the following procedure [11]:

- Construct a matrix $\mathbf{f} = (f(t_{i,L_1},t_{j,L_2}))_{i=0,j=0}^{L_1,L_2}$ of values of the function f at the tensor-product Chebyshev points,
- Extend $\mathbf{f}$ to its even counterpart $\hat{\mathbf{f}}$:

$$\begin{aligned}
\hat{f}_{i,j} &= f_{i,j}, \;\; i = 0,\ldots,L_1, \; j = 0,\ldots,L_2, \\
\hat{f}_{L_1+i,j} &= f_{L_1-i,j}, \;\; i = 1,\ldots,L_1-1, \; j = 0,\ldots,L_2, \\
\hat{f}_{i,L_2+j} &= f_{L_1-i,j}, \;\; i = 0,\ldots,L_1, \; j = 1,\ldots,L_2-1, \\
\hat{f}_{L_1+i,L_2+j} &= f_{L_1-i,L_2-j}, \;\; i = 1,\ldots,L_1-1, \; j = 1,\ldots,L_2-1.
\end{aligned}$$

- perform a bivariate fast Fourier transform (FFT) to get $\tilde{\mathbf{f}} = \mathrm{FFT}(\hat{\mathbf{f}})$,
- extract the first (L_1+1,L_2+1) coefficients of $\tilde{\mathbf{f}}$ to get $\mathbf{g} = (\tilde{f}_{i,j})_{i,j=0}^{L_1,L_2}$. The matrix $\mathbf{g}$ then contains the tensor product Chebyshev coefficients of f.

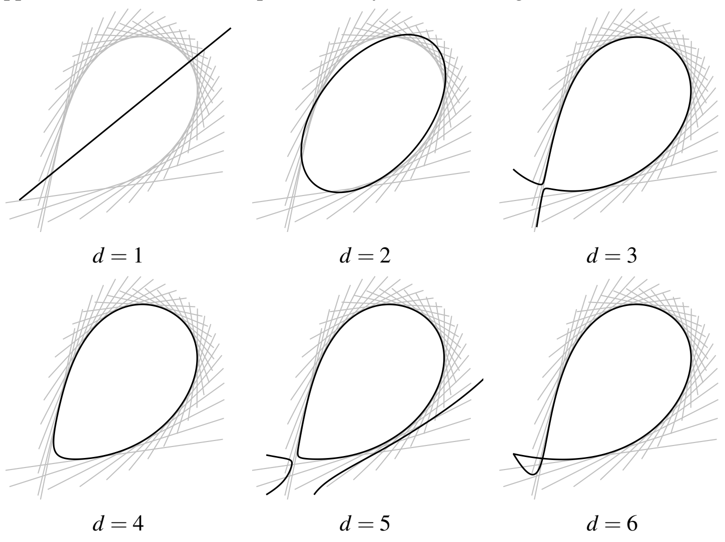
The algorithm for approximate implicitization proceeds by applying the above procedure to the functions $\{w^d(\beta_k \circ \mathbf{p})\}_{k=1}^{M}$ and $\{-h^2\alpha_l\}_{l=1}^{L_1L_2}$, and arranging the coefficients in matrices $\mathbf{D}_q$ and $\mathbf{D}_\lambda$ according to the definition (7). The efficiency of the method is due to it being based on point sampling and FFT. Moreover, the sampling can be done entirely in parallel making the method highly suitable for implementation on heterogeneous architectures.

4 Numerical results

In this section we present an example of the method along with both computation times and estimations for the convergence rates. In order to generate reliable data, we have chosen a degree six family of lines which has a rational envelope. We can thereby use a parametrization of the envelope to compute the algebraic error of the approximations. The family of lines is pictured in Figure 1, along with the envelope approximations up to the exact implicitization at degree six. For these approximations we take $k_1 = \max(0, dn_1 - 2\deg_s(h))$ and $k_2 = \max(0, dn_2 - 2\deg_t(h))$, since this is also the minimum needed for the exact solution.

It can be seen that with increased degree the approximations converge quickly. It is possible that with higher degrees, extra branches may appear in the region of interest. For example, the approximation of degree five has an extra branch close to the envelope curve. However, such artifacts could be avoided using a suitable collection of low-degree approximations (see [9] for an adaptive algorithm).

Fig. 1 Approximations of the envelope of a family of lines for degrees $d = 1, \ldots, 6$.



In Table 1 we show the computation times for the above approximations. The algorithm has been implemented in the Python programming language using the NumPy library for the built in FFT and singular value decomposition (SVD) algorithms. The results are computed on a 3.4Ghz Intel Core i7-2600 with 8GB RAM.

Instead of increasing the polynomial degree d, one may also improve the quality of the approximations by subdivision; the envelope is then approximated by a piecewise implicit representation. It is thus of interest to see how the approximation improves as the region $\Omega = I \times J$ is reduced.

Table 1 Computation times and number of matrix coefficients for the examples in Fig. 1.

Degree d	1	2	3	4	5	6
# coefficients	196	975	2964	7000	14136	25641
Time (s)	0.02	0.04	0.11	0.23	0.45	0.80

Consider a region $\Omega_i = I_i \times J_i$, of diameter 2^{-i} centered on a point (s_0, t_0) in

$$\mathscr{H} = \{(s,t) \in \Omega : h(s,t) = 0\}.$$

For an approximation $q_{d,i}$ of degree d to over the region Ω_i, we define the maximum algebraic error to be

$$\varepsilon_{d,i} = \max_{(s,t)\in\mathscr{H}\cap\Omega_i} |q_{d,i}(\mathbf{p}(s,t))|,$$

where the coefficients $\mathbf{c}_{q_{d,i}}$ of $q_{d,i}$ have been renormalized to $\|\mathbf{c}_{q_{d,i}}\| = 1$, in order to give meaningful results. Given two approximations $q_{d,i}$, and $q_{d,i+1}$, on subsequent subdivision regions Ω_i and Ω_{i+1}, we define the convergence rate to be $r_{d,i} = \log_2(\varepsilon_{d,i}/\varepsilon_{d,i+1})$. Table 2 shows values of $\varepsilon_{d,i}$ and $r_{d,i}$ for four successive subdivisions of the example in Figure 1 and degrees d up to four. Values of $\varepsilon_{d,i}$ below machine precision have been omitted.

Table 2 Maximum algebraic error $\varepsilon_{d,i}$, of the approximations of the example in Fig.1, together with approximate convergence rates $r_{d,i}$.

		Implicit Degree d							
		1		2		3		4	
		$\varepsilon_{d,i}$	$r_{d,i}$	$\varepsilon_{d,i}$	$r_{d,i}$	$\varepsilon_{d,i}$	$r_{d,i}$	$\varepsilon_{d,i}$	r_i
Diameter 2^{-i}	1	1.69e-1	-	6.23e-3	-	1.16e-4	-	3.96e-6	-
	1/2	1.67e-1	0.02	3.46e-4	4.170	2.62e-6	5.467	2.66e-10	13.86
	1/4	3.04e-2	2.458	1.52e-5	4.511	1.50e-9	10.77	1.34e-14	14.27
	1/8	6.52e-3	2.223	5.02e-7	4.915	2.87e-12	9.028	n/a	n/a
	1/16	1.41e-3	2.213	1.58e-8	4.989	5.63e-15	8.993	n/a	n/a

As can be seen from Table 2, the error $\varepsilon_{d,i}$ decreases both with increased degree and increased levels of subdivision. The values of $r_{d,i}$, suggest that the convergence rates for $d = 1, 2, 3$ and 4 are approximately two, five, nine and 14 respectively. This corresponds directly to the number of degrees of freedom in approximating with lines, conics, cubics and quartics and is hence as high a convergence as we can expect, supporting our choices for the degrees (k_1, k_2). The results in Table 2 are typical of rational examples we have tested.

It should be noted that in general, envelope curves are not rational. Thus, this example, whilst showing that high convergence rates are attainable, cannot conclude that this is always the case. However, from studying additional examples, our experience shows that convergence behaviour is good in the general setting.

5 Conclusion

We have presented a new implementation of approximate implicitization of envelope curves using Chebyshev polynomials. We have detailed the computation times and convergence behaviour of a specific example, thereby demonstrating the feasibility of our approach. This paper also motivates theoretical work on convergence rates as a direction for future research.

Acknowledgements The research leading to these results has received funding from the European Community's Seventh Framework Programme FP7/2007-2013 under grant agreement n° PITN-GA-2008-214584 (SAGA), and from the Research Council of Norway (IS-TOPP). It was also supported by the Doctoral Program "Computational Mathematics" (W1214) at the Johannes Kepler University of Linz.

References

[1] Abdel-Malek, K., Yang, J., Blackmore, D., Joy, K.: Swept volumes: foundation, perspectives, and applications. International Journal of Shape Modeling **12**(1), 87–127 (2006)
[2] Barrowclough, O., Dokken, T.: Approximate implicitization using linear algebra. Journal of Applied Mathematics (2012). doi:10.1155/2012/293746
[3] Dokken, T.: Approximate implicitization. Mathematical methods for curves and surfaces, Oslo 2000 pp. 81–102 (2001)
[4] Dokken, T., Thomassen, J.: Weak approximate implicitization. In: IEEE International Conference on Shape Modeling and Applications, 2006. SMI 2006, pp. 204–214 (2006)
[5] Hoffmann, C.: Implicit curves and surfaces in CAGD. IEEE Computer Graphics and Applications **13**(1), 79–88 (1993)
[6] Kim, Y., Varadhan, G., Lin, M., Manocha, D.: Fast swept volume approximation of complex polyhedral models. Computer-Aided Design **36**(11), 1013–1027 (2004)
[7] Peternell, M., Pottmann, H., Steiner, T., Zhao, H.: Swept volumes. Computer-Aided Design Appl **2**, 599–608 (2005)
[8] Rabl, M., Jüttler, B., Gonzalez-Vega, L.: Exact envelope computation for moving surfaces with quadratic support functions. In: Lenarčič, Wenger (eds.) Adv. in Robot Kinematics: Analysis and Design, pp. 283 – 290. Springer (2008)
[9] Schulz, T., Jüttler, B.: Envelope computation in the plane by approximate implicitization. Appl. Algebra in Eng., Comm. and Comp. **22**, 265–288 (2011)
[10] Sendra, J., Winkler, F., Perez-Diaz, S.: Rational algebraic curves. Springer (2007)
[11] Trefethen, L.N.: Spectral methods in MATLAB. SIAM, Philadelphia, PA, USA (2000)