Institut für Numerische Simulation
Rheinische Friedrich-Wilhelms-Universität Bonn

Wegelerstraße 6. 53115 Bonn. Germany
phone +49 228 73-3427. fax +49 228 73-7527
www.ins.uni-bonn.de

Maharavo Randrianarivony

IMPLICITIZATION OF HIGH DEGREE TENSOR PRODUCT CAGD ENTITIES

INS Preprint No. 0905

March 2009

IMPLICITIZATION OF HIGH DEGREE TENSOR PRODUCT CAGD ENTITIES

MAHARAVO RANDRIANARIVONY

Abstract. We focus on the implicitization of parametric CAGD entities which are supposed to be of tensor product type. The presented approach is featured by its ability to treat polynomial or rational splines having a great number of control points. For B-splines, we describe a method of exactly obtaining the Fourier coefficients without using quadrature rules. As for NURBS, a preprocessing step is necessary to convert them into an easy structure. The implicitization is obtained from the convolutions of the parametric CAGD entities. To speed the computations, one can use FFT based algorithms.

1. Introduction

Parametric representations [9, 12] are efficient when one wants to trace the corresponding curves or surfaces. They are also very useful for generating meshes in which nodes need to be inserted or shifted. In some other geometric tasks, working with implicit representations [2] is better than working with the parametric ones. For instance, deciding whether a point lies on one side or the other side of a surface is easier for implicit setting. It is therefore advantageous to have both representations simultaneously. That advantage is often observed in geometric operations such as the determination of the curve which is the intersection of two surfaces. Such advantages might also be useful in some numerical methods [10] where one has only a cloud of unorganized points inside a 3D solid. Unfortunately, most of the entities which are found in CAD standards [19, 20, 21] are only provided in parametric setting. Hence, we are interested here in some way to represent those parametric entities in implicit forms.

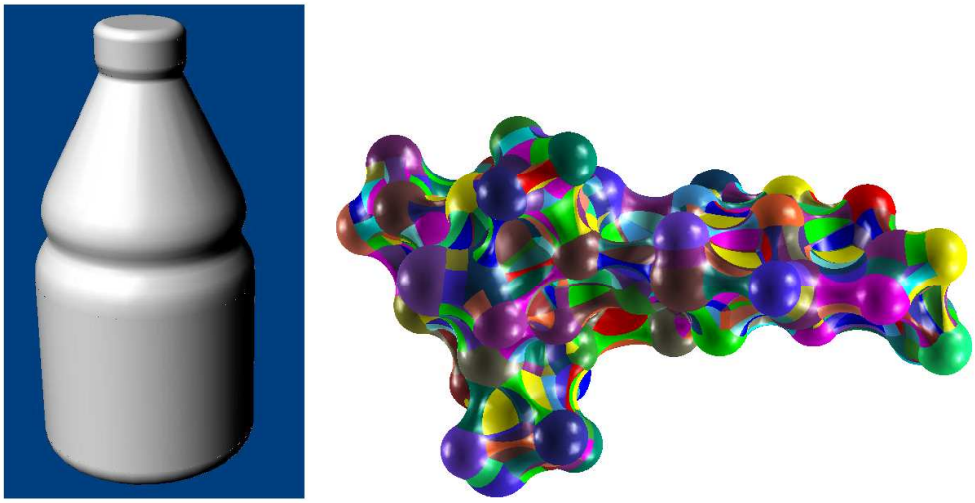
Let us first consider some related works. We do not attempt here to summarize all the existing methods in implicitization which is a century old problem because there are a large number of them. Both symbolic [5] and numerical methods [8] are vastly available. It depends on the subsequent applications and the type of the curves or surfaces to choose the method which best fits. Let us only consider some works which made us design our approach. Traditional implicitization approaches have their roots from elimination theory. Usual implicitization methods [1, 5] based on Bézout or Sylvester

Key words and phrases. implicitization, Fourier transform, tensor product, CAGD.

resultants are analytically elegant but they are flawed by the having to exactly compute the determinant of a large matrix everytime one wants to evaluate the resulting implicit function. That flaw does not make those methods attractive because the entries of the matrix depend on the variables of the resulting implicit functions. Additionally, for practical CAGD models, one usually has piecewise polynomials or piecewise rational functions. Applying the traditional methods to each polynomial or rational piece does not give a good implicitization of the whole curve. J. Winkler has introduced an improvement [22] in implicitization techniques because he extended the classical Bézout forms which require monomial bases to Bernstein bases. Thus, a prior change of bases is not necessary. That improvement was fundamental for that Bernstein bases are more frequent in CAGD and that their formulations are usually more stable than the monomial ones. In [18], implicitization of polynomial and rational curves are done using the moving curves. The work in [23] also uses implicitization method which invokes matrix annihilator but it does not consider CAGD curves. In practice, its method requires least square approximation from point samples to find the Fourier coefficients. Additionally, it did not consider the case of surfaces. For low degree curves and surfaces, there are many efficient techniques which specify the implicitization exactly. In contrast, not much is known about the treatment of the implicitization of high degree CAGD curves and surfaces.

In this paper, we address the problem of finding the implicit form of a tensor product geometric shape. We follow the path of J. Winkler by dealing directly with CAGD entities which are in our case Bézier, B-splines and NURBS. The proposed method works even for polynomial and rational splines with many control points. Since raw data which come from CAD standard or molecular surface as in Fig.1 are not directly in tensor product structure, our former method [16, 17] can be used as preprocessing step before applying the method described here.

This paper is organized as follows. It starts by specifying the problem setting in Section 2. Additionally, we introduce there some notions related to tensor product splines. Since the method of implicitization is based on Fourier analysis, we describe in Section 3.1 the way of obtaining the Fourier coefficients of tensor product B-splines. For B-splines and Bézier entities [15], the Fourier coefficients can be calculated analytically without using quadrature rules. The case of NURBS [13] or similar functions are found in Section 3.2. Unfortunately, as opposed to B-splines, we are not able to exactly obtain the Fourier coefficients of NURBS because of its rationality. We will present a method to efficiently overcome that difficulty. The implicitization method using available Fourier coefficients is described in Section 4. Toward the end of this document, we present some practical results.



(a) (b)

FIGURE 1. (a)Unsegmented CAD model (b)Unsegmented molecular surface in Connolly form.

2. Problem setting and preliminaries

Suppose that we have M parametric functions X_i which map $\mathbf{t} = (t_1, ..., t_N) \in \mathcal{D} \subset \mathbb{R}^n$ to

$$\mathbf{X}(t_1, ..., t_N) = \begin{bmatrix} X_1(t_1, ..., t_N) \\ X_2(t_1, ..., t_N) \\ \cdots \\ X_M(t_1, ..., t_N) \end{bmatrix} \in \mathbb{R}^M. \tag{2.1}$$

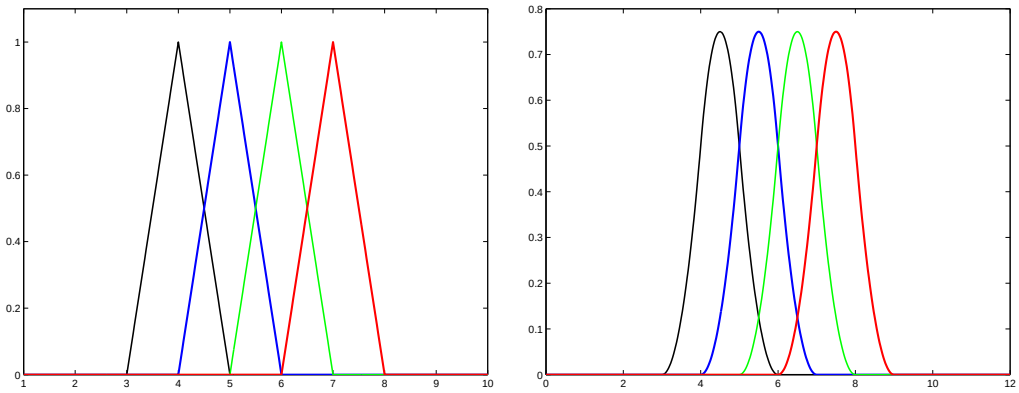
In the case of planar curves, the number of parameters t_i is $N = 1$ and number of image components is $M = 2$. As for surfaces embedded in the 3D space, we have two parametric variables and three image components i.e. $N = 2$ and $M = 3$.

The purpose of this document is to search for an implicit representation $F(X_1, ..., X_M)$ for the above parametric setting. That is, for all $\mathbf{t} \in \mathcal{D}$, the function F verifies

$$F\big(X_1(\mathbf{t}), X_2(\mathbf{t}), \cdots, X_M(\mathbf{t})\big) = 0. \tag{2.2}$$

In this document, we deal with parametric functions X_i that are usually utilized in CAGD. Consider a sequence of non-uniform knot values $(\tau_i) \subset \mathbb{R}$ which are not necessarily pairwise distinct. We recall [7] the divided difference to be $[\tau_i]f := f(\tau_i)$ and

$$[\tau_i, \tau_{i+1}, ..., \tau_p]f := \begin{cases} \big([\tau_{i+1}, ..., \tau_p]f - [\tau_i, ..., \tau_{p-1}]f\big)/(\tau_p - \tau_i) & \text{if} \quad \tau_i \neq \tau_p, \\ f^{(p-i)}(\tau_i)/(p-i)! & \text{if} \quad \tau_i = \tau_p. \end{cases} \tag{2.3}$$



(a) (b)

FIGURE 2. B-spline bases functions on uniform knots: (a)Piecewise linear N_i^1 (b)Piecewise quadratic N_i^2.

The divided difference is related to the higher derivatives by

$$[\tau_i, ..., \tau_{i+\ell}]f = \frac{f^{(\ell)}(\sigma)}{\ell!} \qquad \text{for some} \qquad \sigma \in [\tau_i, \tau_{i+\ell}]. \tag{2.4}$$

In order to introduce the B-spline basis, let us consider any constant integer $k \geq 2$ which specifies the smoothness of the spline and a knot sequence $\tau_0, ..., \tau_{n+k}$ such that $\tau_{i+k} \neq \tau_i$. The usual definition of B-spline basis functions [12, 15] with respect to the knot sequence $(\tau_i)_i$ is

$$N_i^1(t) := \begin{cases} 1 & \text{if} \quad t \in [\tau_i, \tau_{i+1}), \\ 0 & \text{otherwise,} \end{cases} \tag{2.5}$$

$$N_i^k(t) := \left(\frac{t - t_i}{t_{i+k-1} - t_i}\right) N_i^{k-1}(t) + \left(\frac{t_{i+k} - t}{t_{i+k} - t_{i+1}}\right) N_{i+1}^{k-1}(t). \tag{2.6}$$

One can immediately observe that $\text{Supp}(N_i^k) = [\tau_i, \tau_{i+k}]$. By induction, one can prove that the above definition is equivalent to the divided difference using the truncated power functions $(\cdot - t)_+^k$ given by

$$(x - t)_+^k := \begin{cases} (x - t)^k & \text{if} \quad x \geq t, \\ 0 & \text{if} \quad x < t, \end{cases} \tag{2.7}$$

which are illustrated in Fig. 3. More precisely, we have the identity

$$N_i^k(t) = (\tau_{i+k} - \tau_i)[\tau_i, ..., \tau_{i+k}](\cdot - t)_+^k. \tag{2.8}$$

This last relation which is used as definitions in old literature is in fact more appropriate for Fourier analysis than the recurrence relation in (2.6). For the special case of cardinal splines where the knots are uniformly spaced which are supposed to be $\tau_i = i \in \mathbb{Z}$, we

have the property

$$N_{i+1}^k(t) = N_i^k(t-1), \quad \text{hence the designation} \quad N^m := N_0^m. \tag{2.9}$$

Additionally, we have the convolution property

$$N^m = N^{m-1} * N^1. \tag{2.10}$$

To ensure that the B-spline functions are closed, we assume that the knot sequence $\tau_0, ..., \tau_{n+k}$ is periodically provided as follows:

$$\tau_{n+1} = \tau_n + (\tau_1 - \tau_0), \tag{2.11}$$

$$\tau_{n+2} = \tau_{n+1} + (\tau_2 - \tau_1), \tag{2.12}$$

$$... \quad ... \quad ... \quad ... \tag{2.13}$$

$$\tau_{n+k} = \tau_{n+k-1} + (\tau_k - \tau_{k-1}). \tag{2.14}$$

For open B-splines, we suppose there is some practical method of making them closed. A B-spline curve with respect to the above bases is

$$f(t) = \sum_{i=0}^{n} \mathbf{d}_i N_i^k(t) \qquad \text{where} \qquad \mathbf{d}_i \in \mathbb{R}^M. \tag{2.15}$$

The above bases can be extended to tensor product cases. That is, for $\mathbf{i} = (i_1, ..., i_N)$ and $\mathbf{k} = (k_1, ..., k_N)$, we denote $N_{\mathbf{i}}^{\mathbf{k}} := N_{i_1}^{k_1} \otimes \cdots \otimes N_{i_N}^{k_N}$ where

$$(N_{i_1}^{k_1} \otimes \cdots \otimes N_{i_N}^{k_N})(t_1, \cdots, t_N) = N_{i_1}^{k_1}(t_1) \cdots N_{i_N}^{k_N}(t_N). \tag{2.16}$$

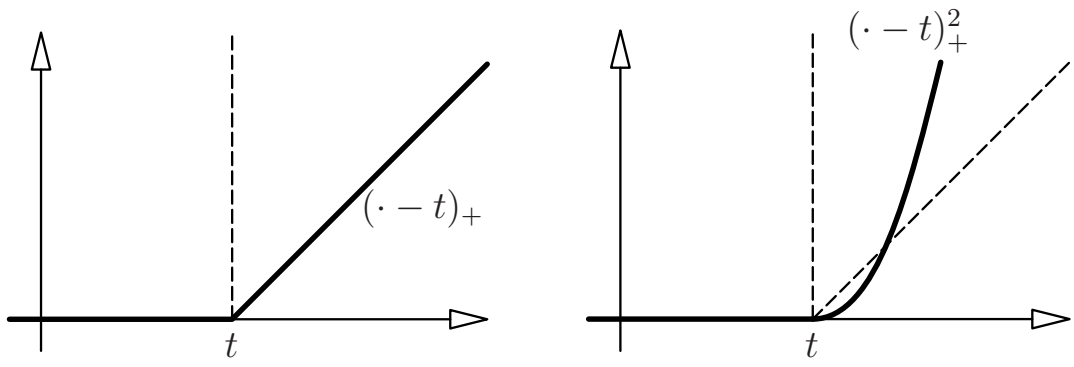
To facilitate the presentation, we suppose that the numbers of control points with respect to all N variables are the same $(n+1)$:

$$f(t_1, ..., t_N) = \sum_{i_1=0}^{n} \cdots \sum_{i_N=0}^{n} \mathbf{d}_{\mathbf{i}} N_{\mathbf{i}}^{\mathbf{k}}(t_1, ..., t_N). \tag{2.17}$$

That simplification assumption holds for the knot sequences too. In fact, more general cases can be treated similarly but the notations would become very complicated. The way of realizing the implicitization invokes some notion from complex Fourier analysis. Therefore, we will reserve the letter j for the complex number $j^2 = -1 \in \mathbb{C}$ throughout this paper.

3. Fourier series of CAGD entities

3.1. **Fourier coefficients of B-splines and Bézier.** In this section, we will compute the Fourier coefficients of B-spline functions having nonuniform knots. It is possible to compute them without using quadrature rules even for the multidimensional case. To that end, we will first treat the case of 1D function. Then, we will deduce the tensor product setting. Suppose that the domain of definition given by the knot sequence



(a) (b)

FIGURE 3. Truncated power functions $(\cdot - t)_+^n$.

(τ_i) is $[-\pi, +\pi]$. Consider now a function f as given in (2.15). Below, we are going to compute the following Fourier coefficients:

$$\widehat{f}(\zeta) := \frac{1}{2\pi} \int_{-\pi}^{\pi} f(t) \exp(-j\zeta t) dt. \tag{3.18}$$

Because of continuity argument and local support of N_i^k, one has the membership of f in L^2 which ensures that it can be reproduced from the inverse relation using the Fourier series:

$$\sum_{\zeta=-\infty}^{\zeta=+\infty} \widehat{f}(\zeta) e^{j\zeta x}. \tag{3.19}$$

An expression with which we deal in section 4 is the following truncated Fourier series

$$\sum_{|\zeta| \leq n} \widehat{f}(\zeta) e^{j\zeta x}. \tag{3.20}$$

For the case of the cardinal splines from (2.10), the computation of the Fourier coefficients can be directly deduced from their property. In fact, from the convolution property, we deduce

$$\widehat{N}^m(\zeta) = \left(\widehat{N}^1(\zeta)\right)^m = \frac{1}{(2\pi j\zeta)^m} \left(1 - e^{-i\zeta}\right)^m. \tag{3.21}$$

Let us now consider B-splines with nonuniform knots. For any fixed ζ, consider the function $\varphi_\zeta(t) := \exp(-j\zeta t)/(-j\zeta)^k$. Its Taylor expansion with integral remainder yields

$$\varphi_\zeta(x) = \sum_{i=0}^{k} \frac{\varphi^{(i)}(-\pi)}{i!} (x+\pi)^i + \frac{1}{k!} \int_{-\pi}^{x} (x-t)^{k-1} \varphi_\zeta^{(k)}(t) dt \tag{3.22}$$

$$= \sum_{i=0}^{k} \frac{\varphi^{(i)}(-\pi)}{i!} (x+\pi)^i + \frac{1}{k!} \int_{-\pi}^{\pi} (x-t)_+^{k-1} \varphi_\zeta^{(k)}(t) dt. \tag{3.23}$$

By taking the divided difference on both sides and by considering (2.4), we obtain

$$[\tau_i, ..., \tau_{i+k}]\varphi_\zeta(x) = \sum_{i=0}^{k} \frac{\varphi^{(i)}(-\pi)}{i!}[\tau_i, ..., \tau_{i+k}](x+\pi)^i + \frac{1}{k!}\int_{-\pi}^{\pi} N_i^k(t)\varphi_\zeta^{(k)}(t)dt. \tag{3.24}$$

In the above equation, in order to see that the order of the integral sign and the divided difference can be swapped, one uses a simple induction with the definition in (2.3). As a result, we obtain

$$[\tau_i, ..., \tau_{i+k}]\varphi_\zeta(x) = \frac{1}{k!}\int_{-\pi}^{\pi} N_i^k(t)\exp(-j\zeta t)dt. \tag{3.25}$$

Or equivalently,

$$\widehat{N}_i^k(\zeta) = \frac{k!}{2\pi(-j\zeta)^k}[\tau_i, ..., \tau_{i+k}]\exp(-j\zeta t). \tag{3.26}$$

As a consequence, the Fourier coefficients of a spline as given in (2.15) are

$$\widehat{f}(\zeta) = \frac{k!}{2\pi(-j\zeta)^k}\sum_{i=0}^{n} \mathbf{d}_i[\tau_i, ..., \tau_{i+k}]\exp(-j\zeta t). \tag{3.27}$$

For the multi-dimensional tensor product case as in relation (2.17), any $\boldsymbol{\zeta} = (\zeta_1, ..., \zeta_N)$ corresponds to the Fourier coefficient

$$\widehat{f}(\boldsymbol{\zeta}) = \frac{1}{(2\pi)^N}\int_{[-\pi,\pi]^N} f(t_1, .., t_N)e^{-j\langle\boldsymbol{\zeta},\mathbf{t}\rangle}dt_1...dt_N, \tag{3.28}$$

in which $\mathbf{t} = (t_1, ..., t_N)$. Thus, by applying a similar argument as above, we obtain the Fourier coefficients for a tensor product B-spline as

$$\widehat{f}(\boldsymbol{\zeta}) = \frac{(k!)^N}{(2\pi)^N\prod_{p=1}^{N}(-j\zeta_p)^k}\sum_{\mathbf{i}=(i_1,...,i_N)} \mathbf{d}_\mathbf{i}\prod_{p=0}^{N}[\tau_{i_p}, ..., \tau_{i_p+k}]\exp(-j\zeta_p t). \tag{3.29}$$

Remark 1. *The above argument can of course be applied to a Bézier curve*

$$\sum_{i=0}^{n} b_i B_i^n(t) \tag{3.30}$$

which is a particular case of B-splines. But independently of that, by using arguments related to hypergeometric functions, it was [4] *proved by Chui et al. that the Fourier coefficients of a Bézier curve are*

$$\exp(-2\pi j\zeta)\sum_{k=0}^{n}(-1)^k\frac{n!}{(n-k)!}\frac{1}{(2\pi j\zeta)^{k+1}}(\nabla^k b_n - \Delta^k b_0) \tag{3.31}$$

where $\nabla b_i := b_i - b_{i-1}$ *and* $\Delta b_i := b_{i+1} - b_i$ *and* $\nabla^k := \nabla^{k-1}\nabla$, $\Delta^k := \Delta^{k-1}\Delta$.

3.2. **NURBS and similar parametric functions.** In this section, we consider functions such as NURBS. It is very difficult to exactly compute the Fourier coefficients of such functions because they are highly non-linear and rational:

$$f(t_1,...,t_N)=\frac{\sum_{i_1=0}^{n}\cdots\sum_{i_N=0}^{n}\omega_{\mathbf{i}}\mathbf{d}_{\mathbf{i}}N_{\mathbf{i}}^{\mathbf{k}}(t_1,...,t_N)}{\sum_{i_1=0}^{n}\cdots\sum_{i_N=0}^{n}\omega_{\mathbf{i}}N_{\mathbf{i}}^{\mathbf{k}}(t_1,...,t_N)}. \tag{3.32}$$

In order to circumvent that difficulty, we propose below an approach of treating them efficiently. The following method is well adapted to NURBS and functions whose higher derivatives can be exactly calculated without any recourse to numerical approximations. Suppose we have such a function f and μ uniform samples $(a_i)_{i=0}^{\mu-1}$. In addition, we have the function values and the higher derivative values corresponding to the samples:

$$v_i^{(0)}:=f(a_i),\quad v_i^{(1)}:=f'(a_i),\quad v_i^{(2)}:=f^{(2)}(a_i),\quad\cdots\quad,\quad v_i^{(m)}:=f^{(m)}(a_i). \tag{3.33}$$

We want to determine an interpolant $\mathcal{H}(f)\in\mathcal{C}^m$ such that

$$\mathcal{H}(f)^{(\ell)}(a_i)=f^{(\ell)}(a_i)\qquad\forall\,\ell=0,1,...,m \tag{3.34}$$

and the Fourier coefficients of $\mathcal{H}(f)$ are exactly computable. Before describing the determination of $\mathcal{H}(f)$, let us first see the approximation order. Let the stepsize be $h:=\max_i|a_i-a_{i+1}|$. Due to the Taylor expansion, we have for any $x\in[a_i,a_{i+1}]$

$$f(x) = \sum_{k=0}^{m}\frac{(x-a_i)^k}{k!}f^{(k)}(a_i)+\mathcal{O}(h^m) \tag{3.35}$$

$$= \sum_{k=0}^{m}\frac{(x-a_i)^k}{k!}\mathcal{H}(f)^{(k)}(a_i)+\mathcal{O}(h^m)=\mathcal{H}(f)(x)+\mathcal{O}(h^m), \tag{3.36}$$

so that we obtain in the maximum norm the following approximation error:

$$\|f-\mathcal{H}(f)\|_\infty=\mathcal{O}(h^m). \tag{3.37}$$

Due to the uniformity of the samples, we suppose in the sequel that there is some $t_0\in[0,1)$ such that $a_i=t_0+i$. Now, we choose any t_1, ..., t_m such that $t_0\le t_1\le\cdots\le t_m<1$ and we define for $p=0,...,m$

$$G_p(x,z)=\sum_{\ell=-\infty}^{\infty}N^{p+1}(\ell+x)z^\ell\qquad\forall\,z\in\mathbb{C} \tag{3.38}$$

where the cardinal spline N^{p+1} from (2.10) is used. Although that involves an infinite sum, only a few terms are relevant because of finite support of N_p. Consider now the

column vector

$$\mathbf{g}(x,u) := \begin{bmatrix} g_0(x,u) \\ g_1(x,u) \\ \cdots \\ g_m(x,u) \end{bmatrix} \qquad \text{where} \qquad g_p(x,u) := G_p(x, e^{-ju}). \tag{3.39}$$

For each $q = 0, ..., m$, we have

$$\begin{aligned} g_q(t_0 + i, 2\pi p/\mu) &= \sum_{\ell=-\infty}^{\infty} N^{q+1}(\ell + t_0 + i)(e^{-2\pi pj/\mu})^{\ell} \\ &= \sum_{\ell=-\infty}^{\infty} N^{q+1}(\ell + t_0)(e^{-2\pi pj/\mu})^{\ell-i} = \omega^{-pi} g_q(t_0, 2\pi pj/\mu), \end{aligned}$$

where $\omega = e^{-2\pi j/\mu}$. As a consequence, we obtain

$$\mathbf{g}(t_0 + i, 2\pi p/\mu) = \omega^{-pi}\, \mathbf{g}(t_0, 2\pi p/\mu). \tag{3.40}$$

Introduce the following matrices where $q = 1, ..., m-1$

$$\begin{aligned} \mathbf{B}_q(u) &:= \Big([t_0]\mathbf{g}(\cdot,u), ..., [t_0, ..., t_{q-1}]\mathbf{g}(\cdot,u), [t_0, ..., t_{q+1}]\mathbf{g}(\cdot,u), ..., [t_0, ..., t_m]\mathbf{g}(\cdot,u)\Big), \\ \mathbf{B}_0(u) &:= \Big([t_0, t_1]\mathbf{g}(\cdot,u), ..., [t_0, ..., t_m]\mathbf{g}(\cdot,u)\Big), \\ \mathbf{B}_m(u) &:= \Big([t_0]\mathbf{g}(\cdot,u), ..., [t_0, ..., t_{m-1}]\mathbf{g}(\cdot,u)\Big). \end{aligned}$$

From those matrices, we define the next determinants

$$F_q(t,u) := \det\Big(\mathbf{g}(t,u), \mathbf{B}_q(u)\Big) \qquad \forall\, q = 0, ..., m. \tag{3.41}$$

The function F_q has the property that for $\ell \neq q$ and for any shifting integer i, we have

$$[t_0, ..., t_\ell] F_q(\cdot + i, 2\pi p/\mu) = 0 \qquad \forall\, p = 0, ..., \mu - 1. \tag{3.42}$$

Indeed, from relation (3.40), we obtain

$$\begin{aligned} [t_0, ..., t_\ell] F_q(\cdot + i, 2\pi p/\mu) &= \det\Big([t_0, ..., t_\ell]\mathbf{g}(\cdot + i, 2\pi p/\mu),\ \mathbf{B}_q(2\pi p/\mu)\Big) \\ &= \omega^{-ji} \det\Big([t_0, ..., t_\ell]\mathbf{g}(\cdot, 2\pi p/\mu),\ \mathbf{B}_q(2\pi p/\mu)\Big) = 0 \end{aligned}$$

because $[t_0, ..., t_\ell]\mathbf{g}(\cdot, 2\pi p/\mu)$ must be amongst the columns of the matrix $\mathbf{B}_q(2\pi p/\mu)$.

We will introduce now some functions ψ_0, ψ_1,..., ψ_m. For each $q = 0, 1, ..., m$, we define

$$\psi_q(t) := \frac{1}{\mu} \sum_{p=0}^{\mu-1} \frac{F_q(t, 2\pi p/\mu)}{[t_0, ..., t_q] F_q(\cdot, 2\pi p/\mu)}. \tag{3.43}$$

As a consequence, we deduce for any shifting parameter $i \in \mathbb{Z}$

$$[t_0, ..., t_\ell]\psi_q(\cdot + i) = 0 \qquad \text{for any} \quad \ell \neq q. \tag{3.44}$$

On the other hand, we proceed as above in order to obtain

$$[t_0,...,t_q]F_q(\cdot+i,2\pi p/\mu)=\omega^{-ji}[t_0,...,t_q]F_q(\cdot,2\pi p/\mu). \tag{3.45}$$

Hence, we obtain the next divided difference

$$[t_0,...,t_q]\psi_q(\cdot+i) = \frac{1}{\mu}\sum_{p=0}^{\mu-1}\frac{[t_0,...,t_q]F_q(\cdot+i,2\pi p/\mu)}{[t_0,...,t_q]F_q(\cdot,2\pi p/\mu)} \tag{3.46}$$

$$= \frac{1}{\mu}\sum_{p=0}^{\mu-1}\omega^{-pi}=\delta_{0,i}. \tag{3.47}$$

The higher degree interpolant will be defined as the following spline

$$\mathcal{H}(f)(x):=\sum_{p=0}^{\mu-1}\sum_{q=0}^{m}v_p^{(q)}\psi_q(x-p). \tag{3.48}$$

If we apply relations (3.44) and (3.47) to the function $\mathcal{H}(f)$ of (3.48), we obtain for any $\ell=0,...,m$

$$\begin{aligned}[t_0,...,t_\ell]\mathcal{H}(f)(\cdot+i) &= \sum_{p=0}^{\mu-1}\sum_{q=0}^{m}v_p^{(q)}\,[t_0,...,t_\ell]\psi_q(\cdot+i-p)\\ &= \sum_{p=0}^{\mu-1}v_p^{(\ell)}\,[t_0,...,t_\ell]\psi_\ell(\cdot+i-p)=\sum_{p=0}^{\mu-1}v_p^{(\ell)}\delta_{0,i-p}=v_i^{(\ell)}.\end{aligned}$$

That is to say, if we choose $t_0=\cdots=t_m$, we obtain from (2.4) the following higher order interpolation properties:

$$\mathcal{H}(f)^{(\ell)}(a_i)=\mathcal{H}(f)^{(\ell)}(t_0+i)=v_i^{(\ell)} \qquad \forall\,\ell=0,...,m. \tag{3.49}$$

Since the function $\mathcal{H}(f)$ is a linear combination of the cardinal splines (2.9), it is immediate to compute its Fourier coefficients by using relation (3.21).

4. Implicitizing tensor product functions

This section is occupied by the description of the implicitization process. Before giving the detail, let us introduce some preliminary notions. We will use the multi-index notation $\boldsymbol{\alpha}=(\alpha_1,...,\alpha_N)$ where each α_i could be positive, negative or zero. Additionally, we denote $|\boldsymbol{\alpha}|=\alpha_1+\cdots+\alpha_N$. We will use also the next two index sets

$$\mathcal{J}_n := \{\boldsymbol{\alpha}=(\alpha_1,...,\alpha_N): \quad |\boldsymbol{\alpha}|\leq n\}, \tag{4.50}$$

$$\mathcal{K}_n := \{\boldsymbol{\alpha}=(\alpha_1,...,\alpha_N): \quad |\boldsymbol{\alpha}|\leq n, \quad \alpha_i\geq 0\}. \tag{4.51}$$

For two multi-indices $\boldsymbol{\alpha}=(\alpha_1,...,\alpha_N)$ and $\boldsymbol{\beta}=(\beta_1,...,\beta_N)$, we define

$$\boldsymbol{\alpha}-\boldsymbol{\beta}:=(\alpha_1-\beta_1,...,\alpha_N-\beta_N). \tag{4.52}$$

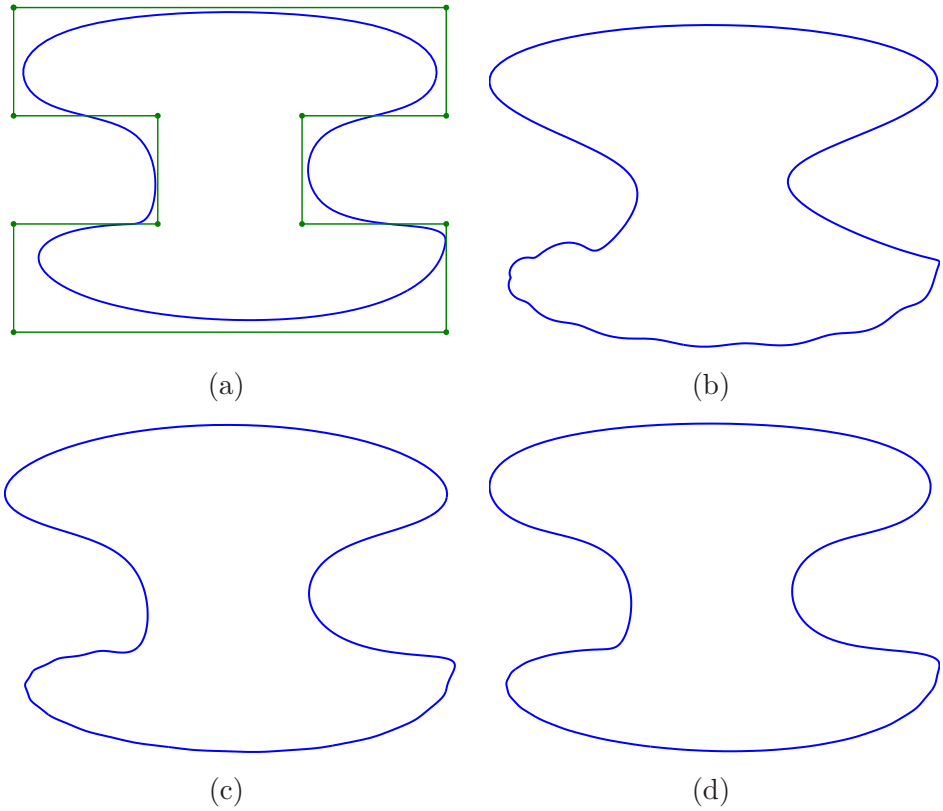


FIGURE 4. Low frequency coefficients capture the geometric shape (a)Original NURBS curve and its control polygon, (b)$n = 13$, (c)$n = 18$, (d)$n = 25$.

By using relation (3.19), it is possible to approximate the original shape by the truncated series (3.20). That expression captures the initial shape efficiently. Further, the approximation can be done with any desired accuracy and the shape is recovered very quickly. As an illustration, we can see in Fig. 4 the images of the representation of a curve by the truncated series. We can compare the original shape in Fig. 4(a) and the approximated ones in Figs. 4(b)–4(d).

In this section, we suppose that the initial parametric functions X_1,...,X_M from relation (2.1) are of the form given in (3.20). That is to say, we suppose that each component function is expressed as

$$X_p(u_1, ..., u_N) = \sum_{|\boldsymbol{\alpha}| \leq n} C_p(\alpha_1, ..., \alpha_N)(e^{ju_1})^{\alpha_1} ... (e^{ju_N})^{\alpha_N}. \tag{4.53}$$

By using the notation $z_i := e^{ju_i}$, we have a multivariate rational function in $\mathbf{z} := (z_1, ..., z_N)$:

$$X_p = \sum_{|\boldsymbol{\alpha}|\leq n} C_p(\alpha_1, ..., \alpha_N) z_1^{\alpha_1} ... \, z_N^{\alpha_N} = \sum_{|\boldsymbol{\alpha}|\leq n} C_p(\boldsymbol{\alpha}) \mathbf{z}^{\boldsymbol{\alpha}}. \tag{4.54}$$

We will use the convolution of two sequences $\mathbf{A} = \big(A(\boldsymbol{\alpha})\big)_{\boldsymbol{\alpha}\in\mathcal{J}_{n_1}}$ and $\mathbf{B} = \big(B(\boldsymbol{\alpha})\big)_{\boldsymbol{\alpha}\in\mathcal{J}_{n_2}}$ as $\mathbf{D} := \mathbf{A} * \mathbf{B} = \big(D(\boldsymbol{\alpha})\big)_{\boldsymbol{\alpha}\in\mathcal{J}_{n_1+n_2}}$ where

$$D(\boldsymbol{\alpha}) := \sum_{\boldsymbol{\beta}} A(\boldsymbol{\beta}) B(\boldsymbol{\alpha} - \boldsymbol{\beta}) \qquad \forall\, \boldsymbol{\alpha} \in \mathcal{J}_{n_1+n_2}. \tag{4.55}$$

As a consequence, any product $X_p X_q$ can be expressed using the convolution as follows

$$X_p X_q = \sum_{\boldsymbol{\alpha}} (C_p * C_q)(\boldsymbol{\alpha}) \mathbf{z}^{\boldsymbol{\alpha}}. \tag{4.56}$$

For each $\mathbf{m} = (m_1, ..., m_M)$ such that $|\mathbf{m}| \leq Q$ which is a parameter discussed below, we introduce the notation

$$\mathcal{M}_{\mathbf{m},\boldsymbol{\alpha}} := \big(\underbrace{C_1 * \cdots * C_1}_{m_1} * \cdots * \underbrace{C_M * \cdots * C_M}_{m_M} \big)(\boldsymbol{\alpha}) \tag{4.57}$$

so that we obtain

$$\prod_{i=1}^{M} X_i^{m_i} = \sum_{|\boldsymbol{\alpha}|\leq nQ} \mathcal{M}_{\mathbf{m},\boldsymbol{\alpha}} \, \mathbf{z}^{\boldsymbol{\alpha}}. \tag{4.58}$$

As a consequence, we obtain a constant complex matrix $\mathcal{M} := \big(\mathcal{M}_{\mathbf{m},\boldsymbol{\alpha}}\big)_{\substack{\mathbf{m}\in\mathcal{K}_Q \\ \boldsymbol{\alpha}\in\mathcal{I}_{nQ}}}$. Let us designate $\mathbf{Y} := \big(\prod_{i=1}^{M} X_i^{m_i}\big)_{\mathbf{m}\in\mathcal{K}_Q}$ and $\mathbf{Z} := (\mathbf{z}^{\boldsymbol{\alpha}})_{\boldsymbol{\alpha}\in\mathcal{J}_{nQ}}$. Hence, relation (4.58) amounts to

$$\mathbf{Y} = \mathcal{M}\mathbf{Z}. \tag{4.59}$$

For a sufficiently large value of Q, there exists $\mathbf{v} = (v_{\mathbf{m}})_{\mathbf{m}\in\mathcal{K}_Q} \subset \mathbb{C}$ which is a left annihilator of the complex matrix $\mathcal{M}$. That is, $\mathbf{v}\mathcal{M} = 0$ or equivalently

$$\sum_{\mathbf{m}\in\mathcal{K}_Q} v_{\mathbf{m}} \mathcal{M}_{\mathbf{m},\boldsymbol{\alpha}} = 0 \qquad \forall\, \boldsymbol{\alpha} \in \mathcal{J}_{nQ}. \tag{4.60}$$

That means, relation (4.59) provides the implicit function corresponding to the parametrization (2.1) as follows

$$\mathcal{F}(X_1, ..., X_M) := \mathbf{v}\mathbf{Y} = \sum_{\mathbf{m}\in\mathcal{K}_Q} v_{\mathbf{m}} \prod_{i=0}^{M} X_i^{m_i} = 0. \tag{4.61}$$

Since the initial parametric functions X_i are real valued, so are the products $\prod_{i=0}^{M} X_i^{m_i}$. Therefore, we deduce from (4.61)

$$\sum_{\mathbf{m}\in\mathcal{K}_Q} \mathcal{R}e(v_{\mathbf{m}}) \prod_{i=0}^{M} X_i^{m_i} + j\, \mathcal{I}m(v_{\mathbf{m}}) \prod_{i=0}^{M} X_i^{m_i} = 0. \tag{4.62}$$

Thus, the final implicitization is

$$F(X_1, ..., X_M) := \sum_{\mathbf{m}\in\mathcal{K}_Q} u_{\mathbf{m}} \prod_{i=0}^{M} X_i^{m_i} \qquad \text{where} \quad u_{\mathbf{m}} := \mathcal{R}e(v_{\mathbf{m}}) \in \mathbb{R}. \tag{4.63}$$

The above implicitization method has several advantages. First, the function F is a multivariate polynomial. Additionally, the coefficients $u_{\mathbf{m}} \in \mathbb{R}$ can be computed once for all and they can be stored for future reference. The implicitization method presented here works even for curves with a large number of control points. Finally, the implicit representation in (4.63) can be efficiently evaluated by means of such techniques as Hörner scheme.

5. Efficient computation

In this section, we want to consider the practical perspective of the above method. For the computer implementation of the multivariate case, the matrix $\mathcal{M}$ in relation (4.59) is not readily applicable to the search of annihilator because the entries are enumerated by the multi-indices. In order to overcome that problem, we need a simple enumeration of the multi-indices (for instance lexicographic) to obtain a matrix which we still denote by $\mathcal{M}$ such that the entries $\mathcal{M}_{pq}$ are indexed by $p \in I$ and $q \in J$ where

$$I := \big\{1, 2, ..., \mathrm{Card}(\mathcal{K}_Q)\big\}, \qquad J := \big\{1, 2, ..., \mathrm{Card}(\mathcal{I}_{nQ})\big\}. \tag{5.64}$$

Let us now briefly discuss about the determination of the left annihilator $\mathbf{v}$ of a complex matrix $\mathcal{M}$ by means of a singular value decomposition (SVD). A left annihilator of $\mathcal{M}$ corresponds to a right annihilator of $\mathcal{M}^T$. That means, the problem amounts to finding a nontrivial element of $\ker(\mathcal{M}^T)$. Suppose we have an SVD decomposition of $\mathcal{M}^T$:

$$\mathcal{M}^T = U\Sigma V^* \tag{5.65}$$

where U and V are orthonormal square matrices and Σ is a diagonal matrix containing the singular values of $\mathcal{M}^T$. A basis of $\ker(\mathcal{M}^T)$ consists of the column vectors of V which correspond to the zero singular values in Σ. Since we need only one annihilator, there is no need to compute a complete singular value decomposition. It is sufficient to find a few singular values closest to zero.

Now, let us turn our attention to the fast computation of the entities in the higher degree spline interpolation by means of FFT. The next method becomes more advantageous as the number of samples μ grows. For each fixed x and q, one needs an efficient way to evaluate $g_q(x, 2\pi p/\mu)$ for all $p = 0, ..., \mu - 1$.

$$g_q(x, 2\pi p/\mu) = \sum_{\ell=-\infty}^{\infty} N^{q+1}(x+\ell)(e^{-2\pi jp/\mu})^{\ell} \tag{5.66}$$

$$= \sum_{k=0}^{\mu-1} \sum_{\ell=-\infty}^{\infty} N^{q+1}(x-\ell\mu+k)(e^{-2\pi jp/\mu})^{k-\ell\mu} \tag{5.67}$$

$$= \sum_{k=0}^{\mu-1} a_k e^{-2\pi jkp/\mu}, \tag{5.68}$$

where $a_k := \sum_{\ell=-\infty}^{\infty} N^{q+1}(x-\ell\mu+k)e^{2\pi jp\ell}$. Note that a_k can be very efficiently computed by using the de Boor algorithm because only very few terms of it are non-zero. In fact, we have

$$a_k := \sum_{\ell=\left\lceil \frac{x-q-1}{\mu} \right\rceil - 1}^{\lceil x/\mu \rceil} N^{q+1}(x-\ell\mu+k)e^{2\pi jp\ell} \tag{5.69}$$

where the domain of summation becomes tighter as the number of samples μ becomes larger. All the expressions in (5.68) are in fact DFT which can be quickly computed by using the FFT method such as the Cooley-Tuckey algorithm [6] that requires only $\mathcal{O}(\mu \log(\mu))$ operations.

6. Numerical results and discussion

In this section, we would like to consider some numerical results related to the above implicitization. First of all, we consider the errors which are due to truncation of the Fourier series which we saw in (3.20). Since the Fourier coefficients of the Splines linearly depend on those of their bases, we consider the error of truncation of the bases functions N_i^k. Below, we consider the following expression for a fixed i

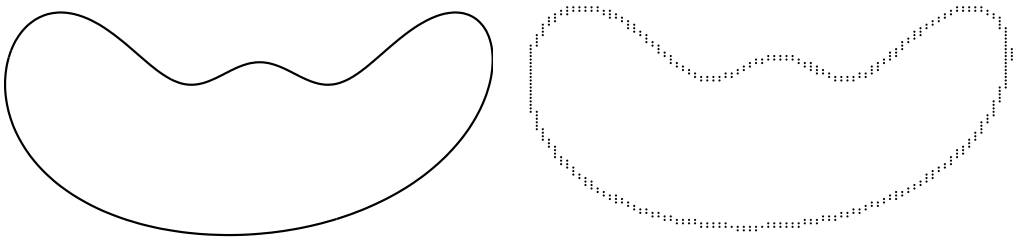
$$\varepsilon_{n,k}(x) := \left| \widehat{N}_i^k(x) - \sum_{|\zeta| \leq n} \widehat{N}_i^k(\zeta) e^{j\zeta x} \right|, \qquad \varepsilon_{n,k} := \max_{x_k \in I} \varepsilon_{n,k}(x_k). \tag{6.70}$$

We made tests for different ranges of the parameter k which specifies the bases smoothness and the size of the truncation threshold n. Above, the set I represents a discrete finite set on the domain of definition of N_i^k. In Table 6.1, we find the resulting values of $\varepsilon_{n,k}$ for different values of n and k.

It can be observed that the error of approximation is not so good when the smoothness is low. For the case $k = 2$, we have the piecewise linear bases functions from Fig. 2(a)

n	$k=2$	$k=3$	$k=4$	$k=5$	$k=6$
5	1.96e-02	3.44e-03	1.10e-03	1.25e-03	9.84e-03
10	6.91e-03	5.09e-03	2.75e-04	4.11e-04	6.51e-03
20	9.64e-03	2.49e-03	2.95e-05	4.26e-05	1.30e-03
25	1.04e-02	1.69e-03	7.18e-06	1.54e-05	6.73e-04
35	1.02e-02	9.66e-04	8.95e-07	1.06e-06	7.35e-05
45	9.58e-03	2.51e-04	1.24e-06	1.01e-07	2.92e-06
50	9.52e-03	1.50e-04	7.94e-07	1.01e-07	6.27e-07
60	8.92e-03	2.97e-05	3.41e-07	1.72e-07	4.63e-07
70	8.79e-03	1.97e-05	1.02e-07	1.11e-07	4.79e-07

TABLE 6.1. Error due to truncation: non-smooth curves are more difficult to approximate.



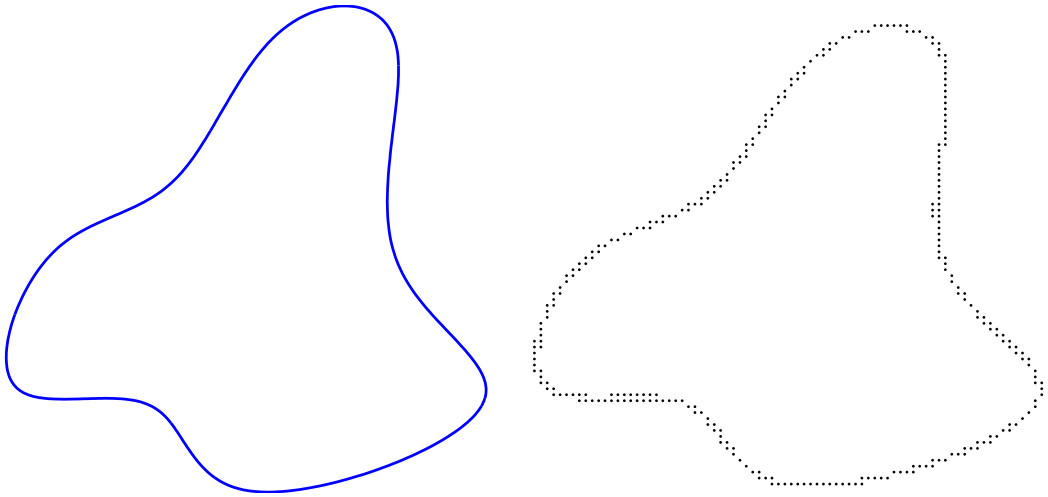
(a) (b)

FIGURE 5. (a) Parametric curve (b) Points (X_1, X_2) such that $|F(X_1, X_2)| < \varepsilon$.

and one needs large truncation threshold n to have good accuracy. In practice, most splines correspond to at least $k = 4$ which corresponds to smoothness $\mathcal{C}^2$. That is for example the case for the famous cubic splines [12] which are used in most data interpolation of real CAD programs.

Another test in which we are interested is the application of the above method to the implicitization of curves. For that, we have considered two parametric curves. After finding the implicitization using the above method, we sort from a set of samples those points (X_1, X_2) such that their values by the implicit functions verify $F(X_1, X_2) < \varepsilon$. The results of such tests can be found in Fig. 5 and Fig. 6.

To close this paper, let us discuss briefly about its limitations. The method presented here is for periodic knots. Therefore, to make it more practical, it must be combined with a process which closes an open spline. Furthermore, as in most implicitization methods, we cannot theoretically exclude the existence of branch points – i.e. those points which annhilate the implicit functions but which are not member of the initial



(a) (b)

FIGURE 6. (a) Parametric curve (b) Points (X_1, X_2) such that $|F(X_1, X_2)| < \varepsilon$.

splines – although they do not happen often in practice. To circumvent those problem, one can restrict to the convex hull of the control points. Points outside the convex hull must have nothing to do with the spline. If that is not yet sufficient, one can split the convex hull recursively, but such a description is beyond the scope of this paper. On the other hand, the approach which we have presented above cannot be immediately used for surfaces having holes or trimmed boundaries. If one wants to treat such surfaces, one has to convert them first into sets of tensor product patches. Afterwards, one can apply the presented method here to the resulting tensor product patches. Our earlier technique in [16, 17] serves well as a preprocessing step before using the above approach.

REFERENCES

[1] P. BIKKER, A. UTESHEV: *On the Bézout construction of the resultant*. J. Symbolic Computation **28** (1999) 45–88.

[2] J. BLOOMENTHAL: *Introduction to implicit surfaces*. Morgan Kaufmann Publishers (1997).

[3] R. BRACEWELL: *The Fourier transform and its application*. McGraw-Hill International Book Company XVIII (1983).

[4] C. CHUI, T. HE, Q. JIANG: *Fourier transform of Bernstein-Bézier polynomials*. J. Math. Anal. Appl. **325** (2007) 294–304.

[5] E. CHIONH, T. SEDEBERG: *On the minors of the implicitization Bézout matrix for a rational plane curve*. Computer Aided Geometric Design **18**, No. 1 (2001) 21–36.

[6] J. COOLEY, J. TUCKEY: *An algorithm for the machine calculation of complex Fourier series*. Math. Comput. **19** (1965) 297–301.

[7] C. DE BOOR: *Divided Differences*. Surv. Approx. Theory **1** (2005) 46–69.

[8] T. DOKKEN: *Approximate implicitization*. Mathematical methods for curves and surfaces (Oslo), (2001) 81–102.

[9] G. FARIN: *Curves and surfaces for Computer Aided Geometric Design: a practical guide*, Academic Press, Boston (1997).

[10] M. GRIEBEL, A. SCHWEITZER: *A Particle-Partition of Unity Method-Part II: Efficient Cover Construction and Reliable Integration*. SIAM J. Sci. Comp. **23**, No. 5 (2002) 1655–1682.

[11] H. HARBRECHT, M. RANDRIANARIVONY: *From Computer Aided Design to wavelet BEM*. To appear in Journal Comput. Vis. Sci. (2008).

[12] J. HOSCHEK, D. LASSER: *Grundlagen der geometrischen Datenverarbeitung*, Teubner, Stuttgart (1989).

[13] L. PIEGL: *On NURBS: a survey, Computer Graphics & Application* **11**, No. 1 (1991) 55–71.

[14] G. PLONKA: *An efficient algorithm for periodic Hermite spline interpolation with shifted nodes*. Numer. Algorithms **5** (1993) 51–62.

[15] H. PRAUTZSCH, W. BOEHM, M. PALUSZNY: *Bézier and B-Spline techniques*, Springer, Berlin (2002).

[16] M. RANDRIANARIVONY: *Geometric processing of CAD data and meshes as input of integral equation solvers*. PhD thesis, Technische Universität Chemnitz, 2006.

[17] M. RANDRIANARIVONY: *Molecular surface decomposition using geometric techniques*. In: Proc. Conf. Bildverarbeitung für die Medizine, Berlin, 197–201 (2008).

[18] T. SEDEBERG, R. GOLDMAN, H. DU: *Implicitizing rational curves by the method of moving algebraic curves*. J. Symbol. Comp. **23**, No. 2-3 (1997) 153–175.

[19] U. S. PRODUCT DATA ASSOCIATION: *Initial Graphics Exchange Specification. IGES 5.3*, Trident Research Center, SC (1996).

[20] M. VOGEL, P. BUNTE: *Pro/ENGINEER und Pro/MECHANICA*, Carl Hanser Verlag, München (2001).

[21] U. WEISSFLOG: *Product data exchange; Design and implementation of IGES processors*. In: Product data interfaces in CAD, CAM applications, edit. Encarnacao, Schuster, Vöge, Springer, Berlin 116–125 (1986).

[22] J. WINKLER: *A resultant matrix for scaled Bernstein polynomials*. Linear algebr. applic. **319**, No. 1-3 (2000) 179–191.

[23] H. YALCIN, M. UNEL, W. WOLOVICH: *Implicitization of parametric curves by matrix annihilation. Int. J. Comput. Vis.* **54**, No. 1-3 (2003) 105–115.

Maharavo Randrianarivony, Institut für Numerische Simulation, Universität Bonn, Wegelerstr. 6, 53115 Bonn, Germany.

E-mail address: randrian@ins.uni-bonn.de