A generalized conversion matrix between non-uniform B-spline and Bézier representations with applications in CAGD

Giulio Casciola, Lucia Romani

Department of Mathematics, University of Bologna, P.zza di Porta San Donato 5, 40127 Bologna, Italy

Abstract

This paper presents the recursive computations of the generalized conversion matrices which allow direct transformations between non-uniform B-spline and Bézier representations of arbitrary degrees, defined over a completely arbitrary interval of the real axis. Although the very general setting assumed, the proposed algorithm turns out to be very easy and practical, and provides interesting applications in computer aided geometric design.

Key words: Matrix representation; Direct transformations; Non-uniform B-spline basis functions; Explicit multi-Bézier form; Clamping and unclamping techniques; Merging spline procedures

1 Introduction

In order to simplify the mathematical treatment of B-splines, during the last two decades several algorithms have been developed to compute the Bézier representation of each non-vanishing segment of a non-uniform B-spline curve (Cohen et al., 1980; Boehm, 1981; Chui and Lai, 1987; Chui, 1988; Grabowski and Li, 1991). While these were exclusively formulated to work out the B-spline-to-Bézier conversion, there are also some algorithms which allow us to convert a B-spline representation into a Bézier representation and vice versa (Boehm, 1977; Sablonniere, 1978; de Casteljau, 1985; Ramshaw, 1987; Prautzsch et al., 2002, pp. 72-73). However, none of these algorithms exploit

Email addresses: `casciola@dm.unibo.it` (Giulio Casciola), `romani@dm.unibo.it` (Lucia Romani).

29 July 2004

the advantages of working with matrix representations. Since the use of matrix forms (largely promoted in CAD/CAM) turns out to be both convenient and practical in representing parametric curves and surfaces, in this paper we propose an alternative conversion method which exploits the matrix notation introduced in Romani and Sabin (2004) to generate direct matrix transformations between non-uniform B-spline and Bézier representations of arbitrary degrees.
While the computation of the conversion matrix between uniform B-spline and Bézier representations (Romani and Sabin, 2004) depends exclusively on the degree of the polynomial representation (and thus can be computed once for all), the conversion matrix between non-uniform B-spline and Bézier representations depends also on the knot configuration and on the parametric domain. As a consequence, the recursive computations presented in Romani and Sabin (2004) necessitate a more general formulation. In this paper we present an algorithm for efficiently computing the conversion matrices in a very general setting. This extended formulation allows us to explain the origin of the fairly criptic initial conditions assumed in Chui's algorithm (Chui and Lai, 1987; Chui, 1988) and to avoid the rather involved use of indices this scheme requires, making our formulation much easier to follow.
Additionally, the possibility of choosing knots in a completely arbitrary configuration, makes our proposal a very powerful procedure for solving problems that very often appear in CAGD.

1.1 Preliminary details

Mathematically, the problem of transforming curve and surface representations is simply a problem of transforming the associated bases.
Let $\{u_{k+j-n},\ \cdots,\ u_k,\ u_{k+1},\cdots,\ u_{k+j+1}\}$ be the arbitrary knot-partition defining the degree-n, $n \geq 1$, non-uniform B-spline basis function $N_{k+j-n,n}(u)$. Then, for any $j = 0, \cdots, n$, let $N_{k+j-n,n}(u)$ denote the $n+1$ basis functions overlapping an arbitrary interval $[a, b]$ of the real axis (not necessarily contained in $[u_k, u_{k+1}]$ as required by the classical procedures).
The key idea behind the B-spline-to-Bézier transformation over the interval $[a, b]$, is to express each $N_{k+j-n,n}(u)$, $j = 0, \cdots, n$ as a linear combination of the degree-n Bernstein-Bézier polynomials $B_{i,n}(u)$, $i = 0, \cdots, n$, by introducing for any $j = 0, \cdots, n$ a set of coefficients $\left\{s_{i,j}^{(n)}\right\}_{i=0,\cdots,n}$ such that

$$N_{k+j-n,n}(u) = \sum_{i=0}^{n} s_{i,j}^{(n)} B_{i,n}(u) \qquad \forall j = 0, \cdots, n \qquad u \in [a, b]. \tag{1}$$

Then, introducing vectors of basis functions, we can rewrite the transformation in (1) in a convenient matrix form as

$$[\,N_{k-n,n}(u) \quad \cdots \quad N_{k,n}(u)\,] = [\,B_{0,n}(u) \quad \cdots \quad B_{n,n}(u)\,]\; S^{(n)}. \tag{2}$$

In this way $S^{(n)} = \left\{s_{i,j}^{(n)}\right\}_{i,j=0,\cdots,n}$ is the $(n+1)\times(n+1)$ matrix by which we can multiply the column vector of control points $C_{k-n},\cdots,C_k$ in the control polygon of a given B-spline curve of degree n, to obtain the Bézier control points $D_0,\cdots,D_n$ of the corresponding span. To compute the piecewise Bézier representation of a complete curve we can build a much larger rectangular matrix by stacking different matrices $S^{(n)}$. Since two adjacent Bézier pieces share a common control point at the junction, it follows that the bottom row of the matrix $S^{(n)}$ associated with the first piece is exactly the same as the top row of the matrix $S^{(n)}$ associated with the following one. This consideration turns out to be useful to compute the $S^{(n)}$ matrices that follow the first. In this paper we take the stacking for granted and focus on the determination of the elements of $S^{(n)}$ for arbitrarily large n by recurrence over the polynomial degree.
Conversely, the Bézier-to-B-spline conversion matrix will be the $(n+1)\times(n+1)$ matrix by which we can multiply the vector of control points $D_0,\cdots,D_n$ of a degree-n Bézier curve, to obtain a sequence of control points $C_{k-n},\cdots,C_k$ of a degree-n B-spline which contains that curve as a span (this time there is no equivalent of stacking because in general a sequence of Bézier curves is not necessarily a B-spline).
In particular, if by introducing for any $i=0,\cdots,n$ a set of coefficients $\left\{r_{j,i}^{(n)}\right\}_{j=0,\cdots,n}$, we express the $n+1$ Bernstein-Bézier polynomials $B_{i,n}(u)$, $i=0,\cdots,n$ defined over the interval $[a,b]$, via the B-spline representation

$$B_{i,n}(u) = \sum_{j=0}^{n} r_{j,i}^{(n)} N_{k+j-n,n}(u) \qquad \forall i=0,\cdots,n \qquad u\in[a,b] \tag{3}$$

and rewrite (3) in matrix form, we have that the matrix $R^{(n)} = \left\{r_{j,i}^{(n)}\right\}_{j,i=0,\cdots,n}$ which satisfies

$$[\,B_{0,n}(u) \quad \cdots \quad B_{n,n}(u)\,] = [\,N_{k-n,n}(u) \quad \cdots \quad N_{k,n}(u)\,]\; R^{(n)}, \tag{4}$$

is exactly the degree-n Bézier-to-B-spline conversion matrix.

Since non-uniform B-spline basis functions and Bernstein-Bézier polynomials are defined by recursion formulae over the degree, the conversion matrices $S^{(n)}$ and $R^{(n)}$ turn out to be defined by a recurrence over the degree n. In the next two sections the recursion algorithms for generating the B-spline-to-Bézier and the Bézier-to-B-spline conversion matrices of arbitrary degree n are presented. The last section is devoted to the description of some interesting applications of these matrix transformations in CAGD.

2 The B-spline-to-Bézier conversion matrix

Since the non-uniform degree-n B-splines $N_{k+j-n,n}(u)$ with $n \geq 1$ turn out to be defined by the following recursion integral relation (Prautzsch et al., 2002, p. 68)

$$N_{k+j-n,n}(u) = n \int_{-\infty}^{u} \left[\frac{N_{k+(j-1)-(n-1),n-1}(v)}{u_{k+j} - u_{k+j-n}} - \frac{N_{k+j-(n-1),n-1}(v)}{u_{k+j+1} - u_{k+j+1-n}} \right] dv, \quad (5)$$

by substituting every B-spline basis function overlapping the interval $[a, b]$ with its Bézier representation in (1) and by solving the Bézier integral that derives from it, we get for any $n \geq 1$ the following formula on the coefficients

$$s_{i,j}^{(n)} = s_{i-1,j}^{(n)} + (b-a) \left[\frac{s_{i-1,j-1}^{(n-1)}}{u_{k+j} - u_{k+j-n}} - \frac{s_{i-1,j}^{(n-1)}}{u_{k+j+1} - u_{k+j+1-n}} \right] \quad (6)$$
$$i = 1, \cdots, n \quad j = 0, \cdots, n$$

with $s_{i-1,-1}^{(n-1)} = s_{i-1,n}^{(n-1)} = 0$ and $S^{(0)} \equiv s_{0,0}^{(0)} = 1$ since $B_{0,0}(u) = N_{k,0}(u) = 1$ for any $u \in [a, b]$.

Looking at (6) we can see that we need to know the value of $s_{0,j}^{(n)}$, $\forall j = 0, \cdots, n$ in order to determine all the entries $s_{i,j}^{(n)}$ $\forall j = 0, \cdots, n,\ i = 1, \cdots, n$. Following the notation introduced in Romani and Sabin (2004), we refer to $s_{0,j}^{(n)}$, $j = 0, \cdots, n$ as the *integration constants* of the degree-n matrix $S^{(n)}$. In general, the computation of the integration constants cannot be worked out easily, since it requires us to determine the value of $N_{k+j-n,n}(a)$ $\forall j = 0, \cdots, n$. The only exception is the uniform case $u_k = k$, when the interval $[a, b] = [k, k+1]$ (see Romani and Sabin, 2004). In fact, in this special case (see Fig.1 as example), since $N_{k+j-n,n}(k) = N_{k+j+1-n,n}(k+1)$, the integration constants can easily be computed as:

$$s_{0,n}^{(n)} = N_{k,n}(k) = 0,$$
$$s_{0,j}^{(n)} = N_{k+j-n,n}(k) = N_{k+j+1-n,n}(k+1) = s_{n,j+1}^{(n)} \quad \forall j = n-1, \cdots, 0.$$

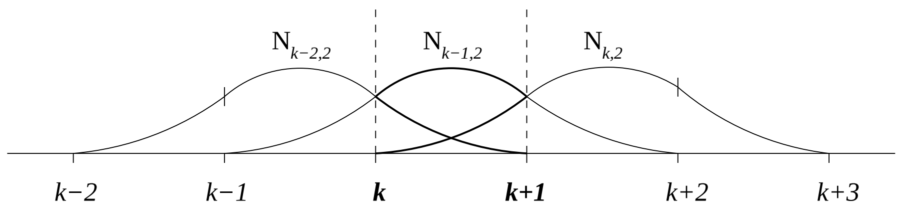

Fig. 1. The uniform quadratic B-spline basis functions overlapping $[a, b] = [k, k+1]$.

In order to overcome the problem of the determination of the integration constants, when $u_k \neq k$ and $[a, b] \neq [k, k+1]$, we introduce a new approach which, instead of the recurrence in (5), exploits the well-known Cox-de Boor recursion formula (Prautzsch et al., 2002, p.61)

$$N_{k+j-n,n}(u) = \frac{u-u_{k+j-n}}{u_{k+j}-u_{k+j-n}} N_{k+j-n,n-1}(u) + \frac{u_{k+j+1}-u}{u_{k+j+1}-u_{k+j+1-n}} N_{k+j+1-n,n-1}(u). \tag{7}$$

By substituting in (7) the Bézier representation in (1) of the $n+1$ B-spline basis functions overlapping the interval $[a,b]$, we obtain the relation

$$\sum_{i=0}^{n} s_{i,j}^{(n)} B_{i,n}(u) = \sum_{i=0}^{n-1} \left[\frac{u-u_{k+j-n}}{u_{k+j}-u_{k+j-n}} s_{i,j-1}^{(n-1)} + \frac{u_{k+j+1}-u}{u_{k+j+1}-u_{k+j+1-n}} s_{i,j}^{(n-1)} \right] B_{i,n-1}(u). \tag{8}$$

Now, by substituting in (8) the recurrence relation on Bernstein polynomials (Prautzsch et al., 2002, p.10)

$$B_{i,n}(u) = \frac{u-a}{b-a}\, B_{i-1,n-1}(u) + \frac{b-u}{b-a}\, B_{i,n-1}(u), \tag{9}$$

we obtain the following expression on the coefficients

$$\frac{u-a}{b-a}\, s_{i+1,j}^{(n)} + \frac{b-u}{b-a}\, s_{i,j}^{(n)} = \frac{u-u_{k+j-n}}{u_{k+j}-u_{k+j-n}}\, s_{i,j-1}^{(n-1)} + \frac{u_{k+j+1}-u}{u_{k+j+1}-u_{k+j+1-n}}\, s_{i,j}^{(n-1)}$$

$$i = 0,\cdots,n-1 \quad j = 0,\cdots,n. \tag{10}$$

Evaluating (10) at $u=a$, we get

$$s_{i,j}^{(n)} = \frac{a-u_{k+j-n}}{u_{k+j}-u_{k+j-n}} s_{i,j-1}^{(n-1)} + \frac{u_{k+j+1}-a}{u_{k+j+1}-u_{k+j+1-n}} s_{i,j}^{(n-1)} \tag{11}$$

$$i = 0,\cdots,n-1 \quad j = 0,\cdots,n,$$

while evaluating (10) at $u=b$, it follows that

$$s_{i+1,j}^{(n)} = \frac{b-u_{k+j-n}}{u_{k+j}-u_{k+j-n}} s_{i,j-1}^{(n-1)} + \frac{u_{k+j+1}-b}{u_{k+j+1}-u_{k+j+1-n}} s_{i,j}^{(n-1)} \tag{12}$$

$$i = 0,\cdots,n-1 \quad j = 0,\cdots,n.$$

Therefore, the entries of the B-spline-to-Bézier conversion matrix $S^{(n)}$, $n \geq 1$, over the interval $[a,b]$, can be defined $\forall j = 0,\cdots,n$, either by the recurrence

$$s_{i,j}^{(n)} = \frac{a-u_{k+j-n}}{u_{k+j}-u_{k+j-n}} s_{i,j-1}^{(n-1)} + \frac{u_{k+j+1}-a}{u_{k+j+1}-u_{k+j+1-n}} s_{i,j}^{(n-1)} \qquad \forall i = 0,\cdots,n-1$$

$$(13)$$

$$s_{n,j}^{(n)} = \frac{b-u_{k+j-n}}{u_{k+j}-u_{k+j-n}} s_{n-1,j-1}^{(n-1)} + \frac{u_{k+j+1}-b}{u_{k+j+1}-u_{k+j+1-n}} s_{n-1,j}^{(n-1)}$$

or by

$$\begin{aligned}
s_{0,j}^{(n)} &= \frac{a-u_{k+j-n}}{u_{k+j}-u_{k+j-n}} s_{0,j-1}^{(n-1)} + \frac{u_{k+j+1}-a}{u_{k+j+1}-u_{k+j+1-n}} s_{0,j}^{(n-1)} \\
s_{i+1,j}^{(n)} &= \frac{b-u_{k+j-n}}{u_{k+j}-u_{k+j-n}} s_{i,j-1}^{(n-1)} + \frac{u_{k+j+1}-b}{u_{k+j+1}-u_{k+j+1-n}} s_{i,j}^{(n-1)} \ \forall i = 0, \cdots, n-1
\end{aligned} \tag{14}$$

with $s_{i,-1}^{(n-1)} = s_{i,n}^{(n-1)} = 0$ and $S^{(0)} \equiv s_{0,0}^{(0)} = 1$.

Remark 1 *Note that $s_{0,j}^{(n)}$, $j = 0, \cdots, n$ computed by the first expression in (14) are exactly the integration constants we need in (6). From now on we will refer to the algorithm given by (6), with integration constants precomputed by (14), using the term* column procedure. *This algorithm extends the computational scheme given by Chui and Lai (1987), Chui (1988) for computing the Bézier representation of non-uniform B-splines with knots chosen in geometric progression, to the more general setting of B-splines with completely arbitrary knot configurations and completely arbitrary parametric domains.*

Remark 2 *Conversely, since both (13) and (14) give us the possibility of working out each row of the conversion matrix $S^{(n)}$ independently of the others, we will define them as the* row procedures.

While the column procedure is computationally more efficient for determining the entries of $S^{(n)}$ in the uniform case $u_k = k$, when $[a, b] = [k, k+1]$ (see Romani and Sabin, 2004), in the non-uniform case over an arbitrary interval $[a, b]$, we can deduce by our complexity analysis and implementation, that the row procedures turn out to be less time-consuming. Additionally, there is also a way to formulate the recursions for the entries (13) (and analogously (14)) by means of the following matrix recursion between the matrices $S^{(n)}$ and $S^{(n-1)}$:

$$S^{(n)} = \left[\begin{array}{rcc} n\left\{\right. & \overbrace{\boxed{S^{(n-1)}L^{(n-1)}}}^{n} & \overbrace{\boxed{\mathbf{0}}}^{1} \\ 1\left\{\right. & \underbrace{\boxed{0}}_{1} & \underbrace{\boxed{\mathbf{s}_{n-1}^{(n-1)}M^{(n-1)}}}_{n} \end{array}\right] \tag{15}$$

where

$\mathbf{0} = [\underbrace{0\ 0\ \dots\ 0}_{n}]^t$ *is a column vector with n zeros, $\mathbf{s}_{n-1}^{(n-1)} = \{s_{n-1,j}^{(n-1)}\}_{j=0,\cdots,n-1}$ is the last row of $S^{(n-1)}$ and $L^{(n-1)}$, $M^{(n-1)}$ are respectively the following $n \times n$ bidiagonal matrices:*

$$L^{(n-1)} = \begin{bmatrix} \frac{u_{k+1}-a}{u_{k+1}-u_{k+1-n}} & \frac{a-u_{k+1-n}}{u_{k+1}-u_{k+1-n}} & \cdots & 0 \\ \vdots & \ddots & & \vdots \\ & \frac{u_{k+j}-a}{u_{k+j}-u_{k+j-n}} & \frac{a-u_{k+j-n}}{u_{k+j}-u_{k+j-n}} & \\ \vdots & & \ddots & \vdots \\ 0 & \cdots & \frac{u_{k+n-1}-a}{u_{k+n-1}-u_{k-1}} & \frac{a-u_{k-1}}{u_{k+n-1}-u_{k-1}} \\ 0 & \cdots & 0 & \frac{u_{k+n}-a}{u_{k+n}-u_k} \end{bmatrix},$$

$$M^{(n-1)} = \begin{bmatrix} \frac{b-u_{k-n+1}}{u_{k+1}-u_{k-n+1}} & 0 & 0 & \cdots & 0 \\ \vdots & \ddots & & & \vdots \\ & \frac{u_{k+n-j+1}-b}{u_{k+n-j+1}-u_{k-j+1}} & \frac{b-u_{k-j+1}}{u_{k+n-j+1}-u_{k-j+1}} & & \\ \vdots & & \ddots & & \vdots \\ 0 & \cdots & \frac{u_{k+n-1}-b}{u_{k+n-1}-u_{k-1}} & \frac{b-u_{k-1}}{u_{k+n-1}-u_{k-1}} & 0 \\ 0 & \cdots & 0 & \frac{u_{k+n}-b}{u_{k+n}-u_k} & \frac{b-u_k}{u_{k+n}-u_k} \end{bmatrix}.$$

3 The Bézier-to-B-spline conversion matrix

Although the Bézier-to-B-spline conversion matrix $R^{(n)}$ coincides with the inverse of $S^{(n)}$, we now want to exploit the *row procedure* approach in order to generate $R^{(n)}$, for any $n \geq 1$, without using the entries of $S^{(n)}$.
By substituting the B-spline representation (3) of the $n+1$ Bernstein polynomials defined over the interval $[a, b]$ in the recurrence relation of Bernstein polynomials (9), it follows that

$$\sum_{j=0}^{n} r_{j,i}^{(n)} N_{k+j-n,n}(u) = \sum_{j=0}^{n-1} \left[\frac{u-a}{b-a} r_{j,i-1}^{(n-1)} + \frac{b-u}{b-a} r_{j,i}^{(n-1)} \right] N_{k+j+1-n,n-1}(u). \quad (16)$$

Now, substituting in (16) the recurrence relation on B-spline basis functions (7), we obtain the following expression on the coefficients

$$\frac{u-u_{k+j+1-n}}{u_{k+j+1}-u_{k+j+1-n}} r_{j+1,i}^{(n)} + \frac{u_{k+j+1}-u}{u_{k+j+1}-u_{k+j+1-n}} r_{j,i}^{(n)} = \frac{u-a}{b-a} r_{j,i-1}^{(n-1)} + \frac{b-u}{b-a} r_{j,i}^{(n-1)}$$

$$j = 0, \cdots, n-1 \quad i = 0, \cdots, n. \quad (17)$$

Evaluating (17) at $u = u_{k+j+1-n}$, we get

$$r_{j,i}^{(n)} = \frac{u_{k+j+1-n} - a}{b-a} r_{j,i-1}^{(n-1)} + \frac{b - u_{k+j+1-n}}{b-a} r_{j,i}^{(n-1)} \quad j = 0, \cdots, n-1 \; i = 0, \cdots, n, \tag{18}$$

while evaluating (17) at $u = u_{k+j+1}$, it follows that

$$r_{j+1,i}^{(n)} = \frac{u_{k+j+1} - a}{b-a} r_{j,i-1}^{(n-1)} + \frac{b - u_{k+j+1}}{b-a} r_{j,i}^{(n-1)} \quad j = 0, \cdots, n-1 \quad i = 0, \cdots, n. \tag{19}$$

Therefore, the entries of the Bézier-to-B-spline conversion matrix $R^{(n)}$, $n \geq 1$, over the interval $[a, b]$, can be defined for any $i = 0, \cdots, n$ either by the recurrence

$$\begin{aligned} r_{j,i}^{(n)} &= \frac{u_{k+j+1-n} - a}{b-a} r_{j,i-1}^{(n-1)} + \frac{b - u_{k+j+1-n}}{b-a} r_{j,i}^{(n-1)} \; \forall j = 0, \cdots, n-1 \\ r_{n,i}^{(n)} &= \frac{u_{k+n} - a}{b-a} r_{n-1,i-1}^{(n-1)} + \frac{b - u_{k+n}}{b-a} r_{n-1,i}^{(n-1)} \end{aligned} \tag{20}$$

or by

$$\begin{aligned} r_{0,i}^{(n)} &= \frac{u_{k+1-n} - a}{b-a} r_{0,i-1}^{(n-1)} + \frac{b - u_{k+1-n}}{b-a} r_{0,i}^{(n-1)} \\ r_{j+1,i}^{(n)} &= \frac{u_{k+j+1} - a}{b-a} r_{j,i-1}^{(n-1)} + \frac{b - u_{k+j+1}}{b-a} r_{j,i}^{(n-1)} \; \forall j = 0, \cdots, n-1 \end{aligned} \tag{21}$$

with $r_{j,-1}^{(n-1)} = r_{j,n}^{(n-1)} = 0$ and $R^{(0)} \equiv r_{0,0}^{(0)} = 1$.

Remark 3 *For any* $i = 0, \cdots, n$ $r_{0,i}^{(n)}$ *computed by (21) are the* integration constants *for the* column procedure

$$r_{j,i}^{(n)} = r_{j-1,i}^{(n)} + \frac{u_{k+j} - u_{k+j-n}}{b-a} \left[r_{j-1,i-1}^{(n-1)} - r_{j-1,i}^{(n-1)} \right] \quad j = 1, \cdots, n \quad i = 0, \cdots, n \tag{22}$$

with $r_{j-1,-1}^{(n-1)} = r_{j-1,n}^{(n-1)} = 0$ *and* $R^{(0)} \equiv r_{0,0}^{(0)} = 1$*, that can easily be obtained by substituting (3) in the integral relation of Bernstein polynomials*

$$B_{i,n}(u) = \frac{n}{b-a} \int_{-\infty}^{u} \left[B_{i-1,n-1}(v) - B_{i,n-1}(v) \right] \, dv \qquad i = 0, \cdots, n \quad u \in [a, b] \tag{23}$$

and by solving the resulting B-spline integral.

As in the non-uniform B-spline-to-Bézier conversion, from our complexity analysis and implementation it follows that the determination of $R^{(n)}$ results less time-consuming using the row procedures, especially by using the formulae in (21). The complexity order of (21) consists of $O(n^3)$ floating point operations ($O(\frac{2}{3}n^3)$ multiplications/divisions and $O(\frac{1}{3}n^3)$ additions/subtractions); the cost is the same as computing the inverse of $S^{(n)}$, but we do not need to know $S^{(n)}$.

4 Geometric applications of the conversion matrices

While in practice, applications of the uniform conversion matrices are very restrictive (see Romani and Sabin, 2004), the generalized conversion matrices proposed in sections 2-3 can be exploited for solving many problems that often appear in CAGD. Indeed, the possibility of choosing knots in a completely arbitrary configuration, makes our proposal a very powerful procedure for working out all the classical algebraic operations on B-spline curves and surfaces (degree-elevation, inner product, etc). In subsections 4.1 and 4.2 we propose respectively to exploit the generalized conversion matrices for computing precise conversions between clamped/unclamped B-splines and for providing approximate conversions of Bézier curves/surfaces that almost join C^0, into Bézier curves/surfaces joined C^k continuously. While in these situations classical approaches turn out to be very tedious, the procedures we are going to propose provide an easy and practical solution.

4.1 Clamping and unclamping techniques

Up to now we have assumed the very general terminology "non-uniform B-splines" to underline that we are working with completely arbitrary knot-configurations. To be more precise, this freedom allows us to define clamped/unclamped and uniform/non-uniform B-splines. In particular, clamped/unclamped refers to whether or not the first and last knot values in the B-spline knot vector are repeated with multiplicity equal to degree plus one, and uniform/non-uniform refers to the knot spacing. To be uniform and unclamped, every knot must be a simple knot and all knots spans must be of equal lengths (this is the case treated in Romani and Sabin (2004)). To be uniform and clamped, only internal knots must be simple and only the internal knot spans must be of equal length. In comparison to unclamped curves, clamped B-splines pass through the first and the last control points and are tangent to the first and the last line segments of the control polygon. If we denote the knot vector by $\{u_0, u_1, ..., u_m\}$, a degree-n Bézier curve on $[u_n, u_{m-n}]$ is an example of clamped B-spline with $n+1$ knots clamped at u_n and u_{m-n}, but

with no internal knots.
In the following paragraphs we provide matrix transformations for precise conversion of an unclamped B-spline into a clamped one and vice versa.

4.1.1 The clamping matrices

Let $\{u_0, u_1, ..., u_m\}$ be the knot vector associated with an unclamped B-spline of degree n defined by $m-n$ control points C_i, $i = 0, ..., m-n-1$. Clamping the curve at u_n and u_{m-n} is nothing more than inserting the knots u_n and u_{m-n} until they have multiplicity $n+1$, and then discarding the knots and control points lying outside the clamped region. The classical procedures (like the knot insertion formula (Boehm, 1981)) are also valid as long as one does not index knots outside the range 0 to m (which does not happen when clamping at u_n and u_{m-n}), but they turn out to be very tedious. On the contrary, using the conversion matrices defined in sections 2-3, we can extract the curve between the parameters u_n and u_{m-n} in a very easy way, that is by a simple matrix multiplication with the column vector of the spline control points. In particular, let us denote by $S^{(n)}_{\Delta,I}$ and $R^{(n)}_{I,\Delta}$ the conversion matrices that transform the degree-n B-spline defined over the knot-partition Δ into the degree-n Bézier curve defined over the interval I, and vice versa.
Then, the control point vector $C^* = [C^*_0, C^*_1, ..., C^*_n, C_{n+1}, ..., C_{m-n-1}]^T$ which defines the original curve clamped at its left end u_n, can be computed in the following way:

$$\begin{bmatrix} C^*_0 \\ C^*_1 \\ \vdots \\ C^*_n \end{bmatrix} = P^{(n)}_{u_n} \begin{bmatrix} C_0 \\ C_1 \\ \vdots \\ C_n \end{bmatrix} \tag{24}$$

where

$$P^{(n)}_{u_n} = R^{(n)}_{I_n,\Delta_n}\ S^{(n)}_{\Delta_0,I_n}$$

is the so-called *clamping matrix at* u_n, defined by $I_n = [u_n, u_{n+1}]$, $\Delta_0 = \{u_0, u_{n+1}, ..., u_{2n+1}\}$ and $\Delta_n = \{\overline{u}_n, u_{n+1}, ..., u_{2n+1}\}$, with $\overline{u}_n$ denoting the n+1-fold knot u_n.
Analogously, the control point vector $C^* = [C_0, ..., C_{m-2n-2}, C^*_{m-2n-1}, C^*_{m-2n}, ..., C^*_{m-n-1}]^T$ which defines the original curve clamped at its right end u_{m-n}, can be computed in the following way:

$$\begin{bmatrix} C^*_{m-2n-1} \\ C^*_{m-2n} \\ \vdots \\ C^*_{m-n-1} \end{bmatrix} = P^{(n)}_{u_{m-n}} \begin{bmatrix} C_{m-2n-1} \\ C_{m-2n} \\ \vdots \\ C_{m-n-1} \end{bmatrix} \tag{25}$$

where

$$P^{(n)}_{u_{m-n}} = R^{(n)}_{I_{m-n},\Delta_{m-n}}\ S^{(n)}_{\Delta_m,I_{m-n}}$$

is the so-called *clamping matrix at* u_{m-n}, defined by $I_{m-n} = [u_{m-n-1}, u_{m-n}]$, $\Delta_m = \{u_{m-2n-1}, u_{m-2n}, ..., u_m\}$ and $\Delta_{m-n} = \{u_{m-2n-1}, u_{m-2n},\ ..., u_{m-n-1}, \overline{u}_{m-n}\}$, with $\overline{u}_{m-n}$ denoting the $n+1$-fold knot u_{m-n}.

Figure 2 shows an example of curve clamping at both the ends. The original degree-4 B-spline defined over the unclamped uniform knot-partition $\{-4, -3, -2, -1, 0, 1, 2, 3, 4, 5, 6, 7, 8\}$ is shown in Figure 2 left, whereas the original curve clamped at both its ends $u_4 = 0$, $u_8 = 4$, is pictured in Figure 2 right.

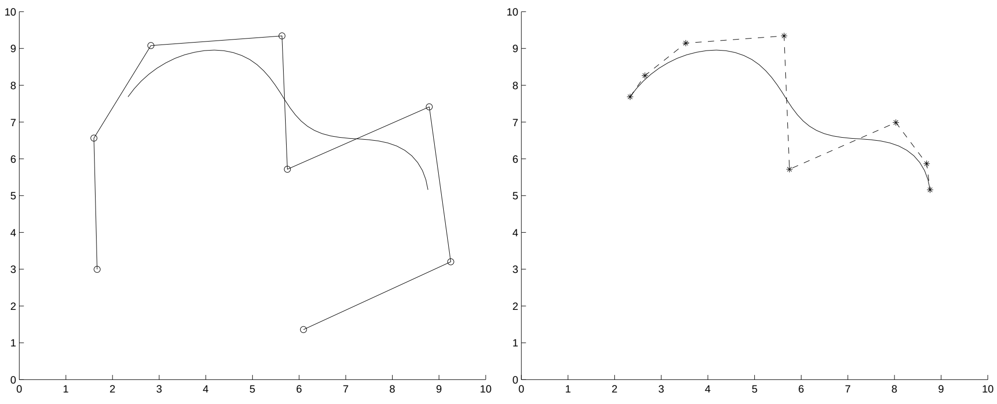

Fig. 2. An unclamped B-spline (left) - The original curve clamped at its ends (right).

In this example the left end clamping matrix is

$$P^{(4)}_0 = \frac{1}{24}\begin{bmatrix} 1 & 11 & 11 & 1 & 0 \\ 0 & 8 & 14 & 2 & 0 \\ 0 & 0 & 18 & 6 & 0 \\ 0 & 0 & 0 & 24 & 0 \\ 0 & 0 & 0 & 0 & 24 \end{bmatrix}$$

while the right end one is

$$P^{(4)}_4 = \frac{1}{24}\begin{bmatrix} 24 & 0 & 0 & 0 & 0 \\ 0 & 24 & 0 & 0 & 0 \\ 0 & 6 & 18 & 0 & 0 \\ 0 & 2 & 14 & 8 & 0 \\ 0 & 1 & 11 & 11 & 1 \end{bmatrix}.$$

4.1.2 The unclamping matrices

On the other hand, unclamping is essentially knot removal. However, we present here an algorithm which computes the new control points at each end all in place, just by a simple matrix multiplication. In fact, if $C = [C_0, C_1, ..., C_n, ..., C_{m-n-1}]^T$ denotes the original control point vector, the control point vector $C^* = [C_0^*, C_1^*, ..., C_n^*, C_{n+1}, ..., C_{m-n-1}]^T$ which defines the original curve unclamped at its left end u_n, can be computed in the following way:

$$\begin{bmatrix} C_0^* \\ C_1^* \\ \vdots \\ C_n^* \end{bmatrix} = Q_{u_n}^{(n)} \begin{bmatrix} C_0 \\ C_1 \\ \vdots \\ C_n \end{bmatrix} \tag{26}$$

where

$$Q_{u_n}^{(n)} = R_{I_n,\Delta_0}^{(n)} \; S_{\Delta_n,I_n}^{(n)}$$

is the so-called *unclamping matrix at* u_n,defined by $I_n = [u_n, u_{n+1}]$, $\Delta_0 = \{u_0, u_{n+1}, ..., u_{2n+1}\}$ and $\Delta_n = \{\overline{u}_n, u_{n+1}, ..., u_{2n+1}\}$, with $\overline{u}_n$ denoting the n+1-fold knot u_n.
Analogously, the control point vector $C^* = [C_0, ..., C_{m-2n-2}, C_{m-2n-1}^*, C_{m-2n}^*, ..., C_{m-n-1}^*]^T$ which defines the original curve unclamped at its right end u_{m-n}, can be computed in the following way:

$$\begin{bmatrix} C_{m-2n-1}^* \\ C_{m-2n}^* \\ \vdots \\ C_{m-n-1}^* \end{bmatrix} = Q_{u_{m-n}}^{(n)} \begin{bmatrix} C_{m-2n-1} \\ C_{m-2n} \\ \vdots \\ C_{m-n-1} \end{bmatrix} \tag{27}$$

where

$$Q_{u_{m-n}}^{(n)} = R_{I_{m-n},\Delta_m}^{(n)} \; S_{\Delta_{m-n},I_{m-n}}^{(n)}$$

is the so-called *unclamping matrix at* u_{m-n}, defined by $I_{m-n} = [u_{m-n-1}, u_{m-n}]$, $\Delta_m = \{u_{m-2n-1}, u_{m-2n}, ..., u_m\}$ and $\Delta_{m-n} = \{u_{m-2n-1}, u_{m-2n}, ..., u_{m-n-1}, \overline{u}_{m-n}\}$, with $\overline{u}_{m-n}$ denoting the $n+1$-fold knot u_{m-n}.

Figure 3 shows an example of curve unclamping. The original degree-3 B-spline defined over the clamped knot-vector $\{0, 0, 0, 0, 1, 2, 3, 4, 4, 4, 4\}$ is shown in Figure 3 left, whereas the original curve unclamped at the left end $u_3 = 0$ is pictured in Figure 3 right.

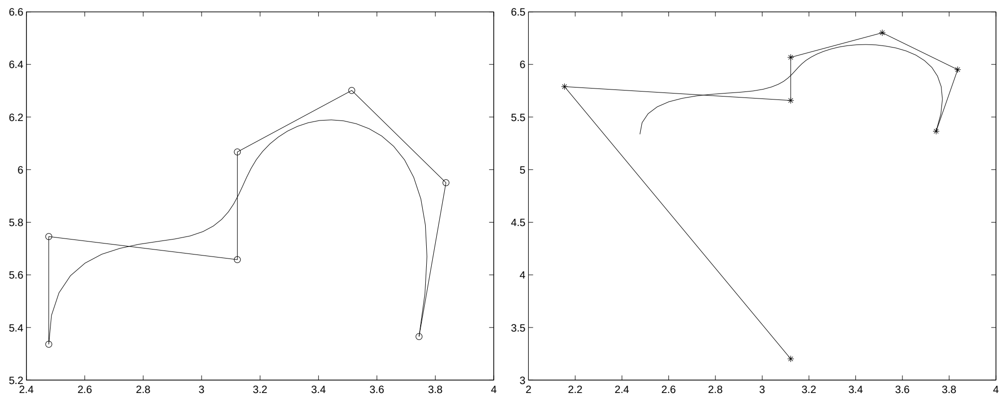
Fig. 3. A clamped B-spline (left) - The original curve unclamped at its left end (right).

In this example the unclamping matrix $Q_0^{(3)}$ is the 4×4 matrix

$$Q_0^{(3)} = \frac{1}{2} \begin{bmatrix} 12 & -12 & 2 & 0 \\ 0 & 3 & -1 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 2 \end{bmatrix}.$$

Remark 4 *Clearly, the extension of these results to surfaces is straightforward. A surface is clamped/unclamped in the u (v) direction by applying the same clamping/unclamping matrix to the rows (columns) of control points. Thus, in these situations, the advantage of working with the proposed transformation matrices is really evident.*

4.2 Merging spline procedures

In this subsection we introduce a procedure for modifying the control points of two assigned degree-n Bézier curves almost C^0, in order they can join C^k-contiuously, for any $k < n$. The name *merging spline procedure* derives from the fact that to join the Bézier curves together, we make their spline representations become part of a unique degree-n spline that approximates them.

4.2.1 Merging spline of a pair of Bézier curves

Without any loss of generality let us assume that $d^0(u) = \sum_{i=0}^n D_{n-i}^0 \; B_{i,n}(u)$ and $d^1(u) = \sum_{i=0}^n D_i^1 \; B_{i,n}(u), \; u \in [0,1]$, are two polynomial curves of degree n given by their Bézier representations, which almost join C^0. In order to make $d^0(u)$ and $d^1(u)$ join exactly C^k-continuously (for a given $k < n$) at the common parameter value 0, we propose a strategy for opportunely modifying their Bézier point sets without changing the original curves significantly. The key idea is exploiting the Bézier-to-B-spline conversion in order to transform $d^0(u)$ and $d^1(u)$ into two B-spline representations $c^0(u) = \sum_{i=0}^{n} C^0_{n-i}\ N_{i,n}(u)$ and $c^1(u) = \sum_{i=0}^{n} C^1_i\ N_{i,n}(u)$, defined over the same knot-partition

$$\{\underbrace{-1,\ -1,\ \cdots,\ -1,\ -1}_{k+1},\ \underbrace{0,\ 0,\ \cdots,\ 0,\ 0}_{n-k},\ \underbrace{1,\ 1,\ \cdots,\ 1,\ 1}_{n+1}\}.$$

In this way, considering $c^0(u-1)$, we are able to represent the two splines over the consecutive intervals $[-1,0]$ and $[0,1]$ on the extended common knot-partition

$$\{\underbrace{-1,\ -1,\ \cdots,\ -1,\ -1}_{n+1},\ \underbrace{0,\ 0,\ \cdots,\ 0,\ 0}_{n-k},\ \underbrace{1,\ 1,\ \cdots,\ 1,\ 1}_{n+1}\}.$$

Thus, defining a unique spline $c(u) = \sum_{i=0}^{2n-k} C_i\ N_{i,n}(u)$ having

$$C_i = \begin{cases} C^0_{n-i} & i = 0, \cdots, n-k-1 \\ \frac{1}{2}(C^0_{n-i} + C^1_{i-n+k}) & i = n-k, \cdots, n \\ C^1_{i-n+k} & i = n+1, \cdots, 2n-k \end{cases}$$

it follows, from the standard spline theory, that $c(u)$ represents the required C^k-join at the common parameter value 0 between the two curves $c^0(u-1)$ and $c^1(u)$ with the first $k+1$ control points slightly modified, since the original Bézier curves are assumed to be almost C^0.

To illustrate this we present a concrete example. Let $d^0(u)$ and $d^1(u)$, $u \in [0,1]$, two assigned sextic Bézier curves almost C^0, we want to modify in order to make them join exactly C^3-continuously (Fig. 4, left). Following the algorithm given above, the two Bézier curves $d^0(u)$ and $d^1(u)$ have to be converted into their B-spline representation $c^0(u)$ and $c^1(u)$ over the knot-partition $\{-1,\ -1,\ -1,\ -1,\ 0,\ 0,\ 0,\ 1,\ 1,\ 1,\ 1,\ 1,\ 1,\ 1\}$ (Fig. 4, right).

Then, by introducing the new set of control points

$$C_i = \begin{cases} C^0_{6-i} & i = 0, \cdots, 2 \\ \frac{1}{2}(C^0_{6-i} + C^1_{i-3}) & i = 3, \cdots, 6\,, \\ C^1_{i-3} & i = 7, \cdots, 9 \end{cases}$$

we define on the knot-partition

$$\{-1,\ -1,\ -1,\ -1,\ -1,\ -1,\ -1,\ 0,\ 0,\ 0,\ 1,\ 1,\ 1,\ 1,\ 1,\ 1,\ 1\}$$

a unique spline $c(u)$ (Fig. 5, left) that restricted to the consecutive intervals [-1,0] and [0,1] gives $c^0(u-1)$ and $c^1(u)$ slightly modified respectively. If $c(u)$ is converted back into two Bézier form curves, they join exactly C^3-continuously (Fig. 5, right).

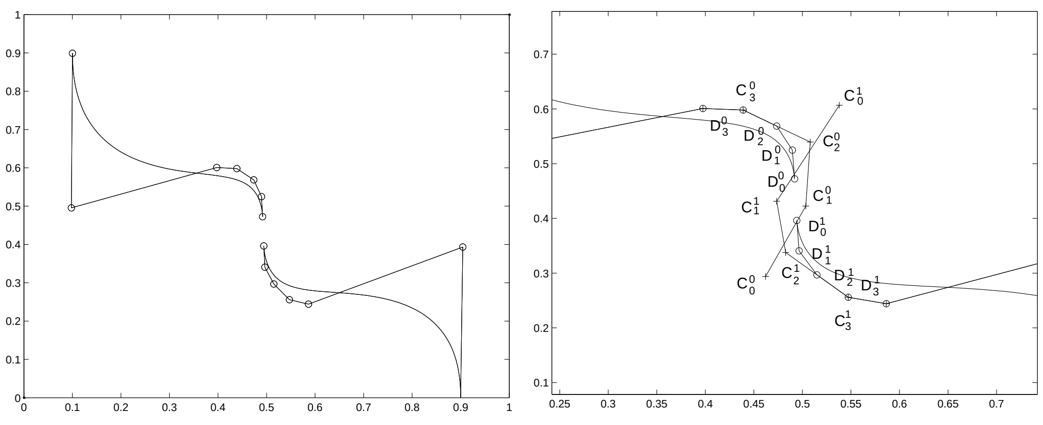

Fig. 4. Bézier (left) and B-spline (right) representation of sextic curves almost C^0.

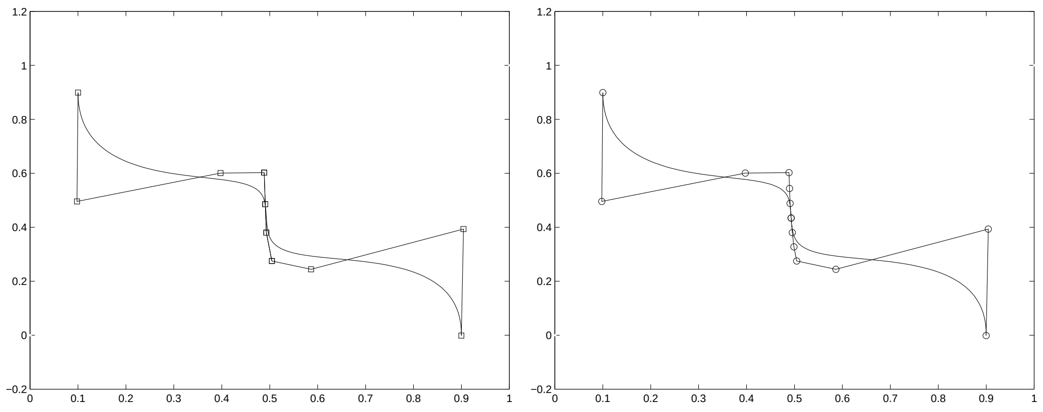

Fig. 5. B-spline (left) and Bézier (right) representation of the two curves C^3-joined.

Remark 5 *Note that (21), as well as being the cheapest procedure for computing the Bézier-to-B-spline transformation, also turns out to be the most convenient when we only have to convert some control points of the Bézier representation, as happens in the example above. In fact, since to join C^k continuously two given degree-n Bézier curves, it is sufficient to modify only the first $k+1$ control points of their Bézier representations, we can limit ourselves to exploiting the row procedure (21) to compute only the first k rows of the Bézier-to-B-spline conversion matrix $R^{(n)}$ (and the first k rows of the B-spline-to-Bézier conversion matrix $S^{(n)}$, if we wish to convert back the two C^k-joined curves into the Bézier form). This is because the last $n+1-k$ control points always remain the same (Fig. 4, right).*
In particular, in the example above, the conversion matrices $R^{(6)}$ and $S^{(6)}$ associated with the knot-partition $\{-1,\ -1,\ -1,\ -1,\ 0,\ 0,\ 0,\ 1,\ 1,\ 1,\ 1,\ 1,\ 1,\ 1\}$ are:

$$R^{(6)} = \begin{bmatrix} 8 & -12 & 6 & -1 & 0 & 0 & 0 \\ 0 & 4 & -4 & 1 & 0 & 0 & 0 \\ 0 & 0 & 2 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{and} \quad S^{(6)} = \begin{bmatrix} \frac{1}{8} & \frac{3}{8} & \frac{3}{8} & \frac{1}{8} & 0 & 0 & 0 \\ 0 & \frac{1}{4} & \frac{1}{2} & \frac{1}{4} & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

Note that, thanks to the assumed notation in the definition of the original Bézier curves, the two conversion matrices are exactly the same for both the curves. Thus it is sufficient to compute them once.

4.2.2 Merging spline of four Bézier patches around a vertex

The procedure given in subsection 4.2.1 can be trivially extended to the tensor-product bivariate case for achieving the C^k-join (for a given $k < n$) of two degree-(n, n) Bézier patches $d^h(u, v) = \sum_{i=0}^{n} \sum_{j=0}^{n} D^h_{i,j}\ B_{j,n}(u)\ B_{i,n}(v)$, $u, v \in [0, 1]$, $h = 0, 1$ that join almost C^0 across their common boundary. Since the tensor-product bivariate case is just a tensor-product of the univariate results, the conversion of a Bézier patch into a B-spline surface is given exactly by applying the univariate matrix $R^{(n)}$ to the rows and columns of the control point matrix.

Consequently, a solution to the problem of smoothly joining four degree-(n, n) Bézier patches $d^h(u, v)$, $h = 0, \cdots, 3$, enumerated anticlockwise around a given point (Fig. 6), follows directly from applying a Bézier-to-B-spline conversion to each patch $d^h(u, v)$ (Fig. 7) over the knot-partition

$$\{\underbrace{-1, \cdots, -1}_{k+1}, \underbrace{0, \cdots, 0}_{n-k}, \underbrace{1, \cdots, 1}_{n+1}\} \times \{\underbrace{-1, \cdots, -1}_{k+1}, \underbrace{0, \cdots, 0}_{n-k}, \underbrace{1, \cdots, 1}_{n+1}\} \tag{28}$$

and determining a C^k-join for the three following pairs of adjacent spline patches $c^{01}(u, v) := (\ c^0(u, v),\ c^1(u, v)\ )$, $c^{23}(u, v) := (\ c^2(u, v),\ c^3(u, v)\ )$, $c^{0123}(u, v) := (\ c^{01}(u, v),\ c^{23}(u, v)\ )$, over the following extended common knot-partition

$$\{\underbrace{-1, \cdots, -1}_{n+1}, \underbrace{0, \cdots, 0}_{n-k}, \underbrace{1, \cdots, 1}_{n+1}\} \times \{\underbrace{-1, \cdots, -1}_{n+1}, \underbrace{0, \cdots, 0}_{n-k}, \underbrace{1, \cdots, 1}_{n+1}\}. \tag{29}$$

In this way, a single spline $c^{0123}(u, v)$, approximating the four original ones, remains defined. More precisely, $c^{0123}(u, v)$ represents a spline over the knot-partition (29) approximating $c^{01}(u, v)$ and $c^{23}(u, v)$, while the latter represent two splines over (29) approximating $c^0(u, v), c^1(u, v)$ and $c^2(u, v), c^3(u, v)$, respectively.

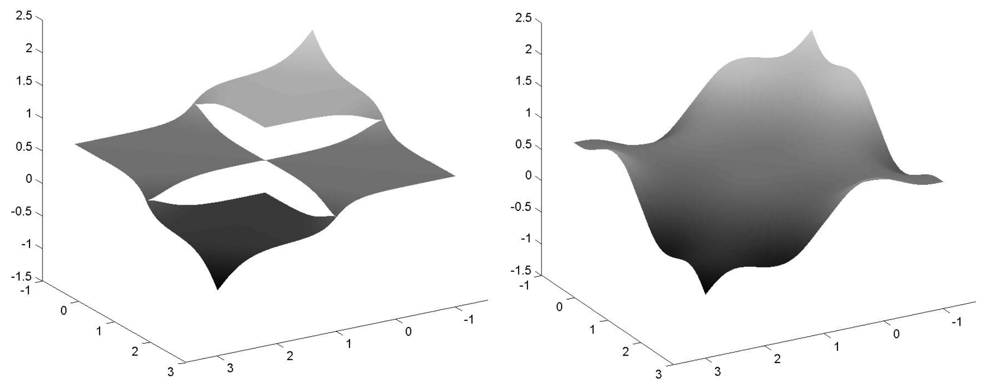

Fig. 6. Four degree-(7,7) Bézier patches almost C^0 (left) and their C^3-join (right).

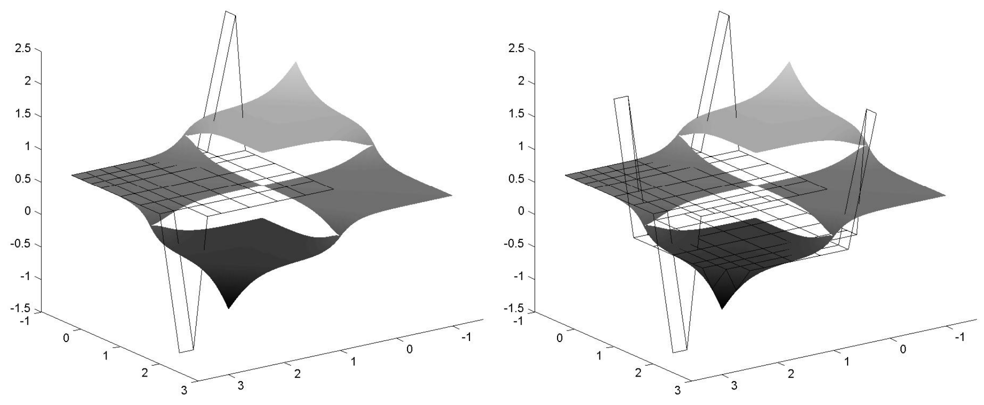

Fig. 7. Bézier-to-B-spline conversion of the 1^{st} (left), 1^{st} and 2^{nd} (right) patches.

Remark 6 *Note that, thanks to the assumed notation, the Bézier-to-B-spline conversion matrix that allows us to represent the four patches on a common knot-partition, is exactly the same for all the patches (that is, for the rows and columns of their control point matrices).*

5 Conclusions

In this paper we have introduced the very general matrix conversion formulae that provide a practical tool for transforming arbitrary non-uniform degree-n B-spline representations into Bézier form and vice versa. While in practice, applications of the uniform conversion matrices are very restrictive (see Romani and Sabin, 2004), the generalized conversion matrices proposed in this paper can be exploited for solving many problems that often appear in CAGD, in a really simple way. Indeed, the possibility of choosing knots in a completely arbitrary configuration, makes our proposal a very powerful procedure for working out all the classical algebraic operations on B-spline curves and surfaces (degree-elevation, inner product, etc). Additionally, as explained in the last section, interesting applications of these general matrix conversions concern also clamping/unclamping and merging spline procedures, since in these situations classical approaches turn out to be very tedious.

Acknowledgements

This work has been supported by MIUR-Cofin 2004 and FIRB 2004.

References

Boehm, W., 1977. Uber die Konstruktion von B-spline-Kurven. Computing 18, 161-166.

Boehm, W., 1981. Generating the Bézier points of B-spline curves and surfaces. Computer Aided Design 13(6), 365-366.

Chui, C.K., Lai, M.J., 1987. Computation of box-splines and B-splines on triangulations of nonuniform rectangular partitions. Approx. Theory Appl. 3, 37-62.

Chui, C.K., 1988. Multivariate Splines. In: CBMS Lectures Series, vol. 54. SIAM, Philadelphia.

Cohen, E., Lyche, T., Riesenfeld, R., 1980. Discrete B-splines and subdivision techniques in computer aided geometric design and computer graphics. Computer Graphics and Image Processing 14(2), 87-111.

de Casteljau, P., 1985. Formes á pôles. In: Mathematiques et CAO, vol. 2. Hermés, Paris.

Grabowski, H., Li, X., 1991. General matrix representation for NURBS curves and surfaces for interfaces. In: Hoschek, J. (Ed.), Freeform Tools in CAD Systems - A Comparison. Teubner, pp. 219-232.

Prautzsch, H., Boehm, W., Paluszny, M., 2002. Bézier and B-spline techniques. Springer-Verlag, Berlin.

Ramshaw, L., 1987. Blossoming: A connect-the-dots approach to splines, Technical Report 19, Digital System Research Center, Palo Alto.

Romani, L., Sabin, M.A., 2004. The conversion matrix between uniform B-spline and Bézier representations. Computer Aided Geometric Design 21(6), 549-560.

Sablonniere, P., 1978. Spline and Bézier polygons associated with a polynomial spline curve. Computer Aided Design 10, 257-261.