ISSN 1392 – 124X INFORMATION TECHNOLOGY AND CONTROL, 2010, Vol. 39, No. 4

DIRECT DEGREE ELEVATION OF NURBS CURVES

Kestutis Jankauskas, Dalius Rubliauskas

Department of Multimedia Engineering, Kaunas University of Technology
Studentų St. 50, LT–51368 Kaunas, Lithuania
e-mail: kestutis.jankauskas@ktu.lt, dalius@soften.ktu.lt

Abstract. In this paper we provide the guidelines for the direct degree elevation of NURBS curves. Through the analysis of linear equation systems of quartic and lower degree splines we derive a direct relation between the knot vector of a spline and the degree elevation coefficients. We also present a direct degree elevation scheme and several algorithms based on the discovered relation. Experimental results indicate that the direct degree elevation algorithms are up to twice more time-efficient than Piegl and Tiller's degree elevation method. This proves the inefficiency of B-spline calculation schemes based on blossoming that involve redundant operations with control points. It also negates Piegl and Tiller's claim about the inefficiency of linear equation system solving method for quartic and lower degrees.

Keywords: degree elevation; degree raising; NURBS.

1. Introduction

NURBS (Non-Uniform Rational B-Spline) is a well-known mathematical model used to represent curves and surfaces. It has become de facto technology in CAD systems because of its compact form and the ability to represent any desired shape. A NURBS curve is composed from a control polygon and the set of basis functions of a specific degree. In some cases a designer has to link two or more B-spline curves of different degree to form a new curve or surface [9]. In both situations the set of input curves must have a common degree [7]. Such a problem can be solved either using degree elevation or reduction. Both solutions should not affect the shape of curves. Although in general, degree reduction is an approximation of an input curve [9]. Therefore, the only way to link several B-spline curves of different degree without affecting their shape is degree elevation.

Degree elevation is considered a fundamental problem [7], although there is no unified and simple approach to it (see Section 2). Degree elevation methods (in some sources referred to as degree raising methods [1, 9]) can be categorized into direct and indirect [9]. Indirect methods, like one provided in [6] and [7], are intuitive and simple to understand. But they may suffer from the lack of precision (because of round-off error [9]) and poorer performance. In this paper we intend to reveal the relation between NURBS curve and its degree elevated version (inspect Section 3). Acquired information allows the composition of direct degree elevation (DDE) scheme and algorithms (see Section 4). In Section 5 we show the effectiveness of the presented algorithms and discuss their potential. The research is concluded in Section 6.

2. Related Work

Let us open this section by quoting Piegl and Tiller [7]. They claim that obvious but very inefficient method to degree elevate NURBS is to solve a system of linear equations. The authors suggest the other method instead that consists of three steps: 1) the extraction of Bézier segments using knot insertion, 2) the degree elevation of Bézier segments, and 3) the removal of unnecessary knots.

We suspected this claim to be disputable because of the research results published in [4]. The research showed that de Boor's knot insertion algorithm is not time-efficient. This happens because of the excess of arithmetic operations with control points. Each operation involving a control point is performed with each of its coordinates. In order to compose a time-efficient algorithm, one must minimize the amount of such operations.

Lee and Park in their paper [5] show that knot insertion as well as de Boor's algorithm relates to the theory of blossoming [2]. As blossoming is the recursive scheme involving arithmetic operations with control points in every iteration, it cannot provide a time-efficient solution. Coincidently, Piegl and Tiller's algorithm is based on the same concept. We have decided to analyze this case by composing an algorithm based on equation solving and compare it to Piegl and Tiller's method.

There are many other approaches to the degree elevation problem. For example, Qin [9] presents the matrix method. Naturally, large matrices require a considerable amount of memory and additional allocation time. Wang in [10] proposes bi-degree basis function approach, although admits its inefficiency because of weight calculation. The most efficient approach we came across was proposed by Huang, Hu and Martin in [3]. It is based on the fact that a curve is uniquely defined by points on curve and its derivatives. It can be applied to NURBS of an arbitrary degree and any knot vector. Moreover, it can elevate the degree by an arbitrary natural number greater than zero. Another set of efficient algorithms is provided by Cohen, Lyche and Schumaker in [1]. The authors give degree elevation methods optimized for linear, quadratic and uniform cubic NURBS. In the following sections we will refer to these methods as to CLS methods.

Performance is not the only factor taken into consideration. According to [3], fast algorithms like that of Prautzsch and Piper's [8] may fail to gain popularity because of the complexity and problematic implementation. Therefore, in this paper we consider the balance between good performance and simplicity.

3. Mathematical Basis

In this section we present the basic concept of NURBS, the mathematical definition of degree elevation problem and the solutions for linear, quadratic, cubic and quartic cases of non-uniform rational B-spline.

3.1. Background of NURBS

As mentioned before, NURBS curve is defined by a set of control points $\mathbf{P}_i$ and a set of basis functions $N_{i,p}(u)$ of certain degree $p \geq 1$, where $i = 0,1...n$. Basis functions are calculated from the so-called knot vector, consisting of $m = n + p + 1$ elements. In our research we use only a clamped normalized knot vector form:

$$\mathbf{U} = \{u_0 = u_1 = ... = u_p = 0 < u_{p+1} \leq u_{p+2} \leq ... \leq u_{n-1} < u_n = .. = u_{n+p-1} = u_{n+p} = 1\}. \quad (1)$$

There are several mathematical representations of NURBS, but the most popular is Cox-de Boor recursion formula. It can be found in the majority of papers referring to NURBS. Although, notations of basis functions, control points and knot values vary greatly. In this paper we refer to the function $N_{i,p}(u)$ as to the basis function of the degree p:

$$N_{i,0}(u) = \begin{cases} 1, \text{if } u_i \leq u < u_{i+1} \\ 0, \text{otherwise} \end{cases}, \quad (2)$$

$$N_{i,p}(u) = \frac{u - u_i}{u_{i+p} - u_i} N_{i,p-1}(u) + \frac{u_{i+p+1} - u}{u_{i+p+1} - u_{i+1}} N_{i+1,p-1}(u). \quad (3)$$

In general case, NURBS is a rational spline, thus each control point $\mathbf{P}_i$ has a positive weight value w_i attached to it. So any point on NURBS curve within the limits of parametric space (in our case $0 \leq u \leq 1$) can be obtained from the equation:

$$\mathbf{C}(u) = \sum_{i=0}^{n-1} N_{i,p}(u) w_i \mathbf{P}_i \Big/ \sum_{i=0}^{n-1} N_{i,p}(u) w_i . \quad (4)$$

The rational B-spline can be easily mapped into homogeneous space to take a non-rational form [7]:

$$\mathbf{C}^w(u) = \sum_{i=0}^{n-1} N_{i,p}(u) \mathbf{P}_i^w . \quad (5)$$

This can be achieved by multiplying each control point $\mathbf{P}_i$ coordinate by the weight value w_i. The weight value becomes an additional coordinate. This means that the rational B-spline positioned in 3D model space can be mapped to 4D homogeneous space by mapping control points [7]:

$$\mathbf{R}^3 \to \mathbf{R}^4 :$$
$$\mathbf{P}_i^w = \{x \cdot w, y \cdot w, z \cdot w, w\} = \{x^w, y^w, z^w, w\}. \quad (6)$$

The superscript w indicates that the entity resides in homogeneous space. Mapping from homogeneous to model space is also simple. It is achieved by dividing coordinates by the weight value:

$$\mathbf{R}^4 \to \mathbf{R}^3 :$$
$$\mathbf{P}_i = \{x^w / w, y^w / w, z^w / w\} = \{x, y, z\}. \quad (7)$$

In fact, we will drop the superscript w for the remainder of this article and treat NURBS curve as if it has a non-rational form. This will not affect the quality of our calculations and the solution will be applicable to the rational case as well (see Subsection 3.3 for the explanation):

$$\mathbf{C}(u) = \sum_{i=0}^{n-1} N_{i,p}(u) \mathbf{P}_i . \quad (8)$$

Now that we have clarified the notation, let us discuss the problem of the degree elevation. Firstly, let us discuss the degree elevation of a Bézier segment, which is a special case of NURBS.

3.2. Bézier Curve Degree Elevation

The (rational) Bézier curve is NURBS defined by a clamped knot vector with a single non-zero knot interval. As we have set the bounds for parametric space to $0 \leq u \leq 1$, the knot vector takes the form of:

$$\mathbf{U} = \{\underbrace{0,0,...,0}_{p+1}, \underbrace{1,1,...,1}_{p+1}\}. \quad (9)$$

In this case, the spline has $n = p+1$ control points and $m = n + p + 1 = 2p + 2$ knots. We are to elevate its degree from p to $\hat{p} = p + 1$. Intuitively, the knot vector of a new spline has the form of:

$$\hat{\mathbf{U}} = \{\underbrace{0,0,...,0}_{p+2}, \underbrace{1,1,...,1}_{p+2}\}. \quad (10)$$

Degree elevated Bézier segment has $\hat{m} = 2p + 4$ knots. Therefore it must have $\hat{n} = \hat{m} - \hat{p} - 1 = p + 2$ control points. Hence, a new Bézier segment has one additional control point ($\hat{n} - n = 1$). Fortunately, it is not difficult to calculate the positions of new control points $\hat{\mathbf{P}}_i$ [7]:

$$\hat{\mathbf{P}}_i = (1 - \alpha_i)\mathbf{P}_i + \alpha_i \mathbf{P}_{i-1}, \quad (11)$$

$$\alpha_i = i / \hat{p}, \; i = 0,..., \hat{p}. \quad (12)$$

In some sources, like [10], this process is called corner cutting because every new control point $\hat{\mathbf{P}}_i$ is positioned on the line between two adjacent $\mathbf{P}_i$ and $\mathbf{P}_{i-1}$, with exception of the first and the last control points. They remain in their former position: $\hat{\mathbf{P}}_0 = \mathbf{P}_0$, $\hat{\mathbf{P}}_{\hat{n}-1} = \mathbf{P}_{n-1}$. Figure 1 illustrates the degree elevation of Bézier spline from cubic to quartic.

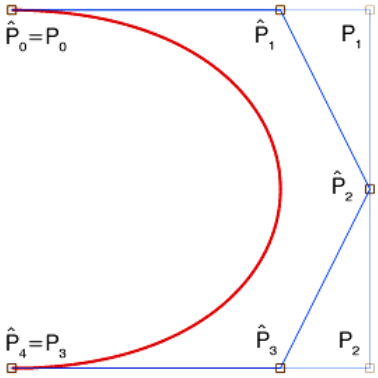


Figure 1. Degree elevation of cubic Bézier spline

Clearly, raising the degree of Bézier spline is very simplistic. The control point repositioning coefficients α_i does not depend on the initial position of control points or on the knot vector. In the remainder of this paper we will refer to (12) as to Bézier coefficients.

Unfortunately, degree elevation of NURBS is a lot more complicated. The following subsections discuss the general concept of degree elevation and special cases of linear, quadratic, cubic, and quartic splines.

3.3. General Concept of NURBS Degree elevation

There are two prerequisites for algorithms that raise the degree of a NURBS curve: input and output curves must match parametrically and geometrically [5, 7]. This means that both splines at any parametric value $0 \le u \le 1$ must produce the point located at the same position in the model space:

$$\mathbf{C}(u) = \hat{\mathbf{C}}(u). \quad (13)$$

Also, the degree elevated spline must retain original differentiability properties [2, 3, 5]. Hence, a new spline must have the same continuity as the original one. This can be achieved by increasing knot multiplicities of the degree elevated spline by one. Let s_l denote a multiplicity of a certain knot, where $l = 0...\eta$ and η is the number of non-zero intervals in the knot vector. Then the original knot vector can be denoted as:

$$\mathbf{U} = \{\underbrace{0,...,0}_{p+1}, \underbrace{u_1^s,...,u_1^s}_{s_1},...,\underbrace{u_{\eta-1}^s,...,u_{\eta-1}^s}_{s_{\eta-1}}, \underbrace{1,...,1}_{p+1}\}. \quad (14)$$

Notice that the knot subscript does not refer to the knot position in a knot vector, but to an index of a unique knot value. This rearrangement of indices does not affect the result of Cox-de Boor formula (3), because the knot vector remains the same. We only grouped equal values. To avoid confusion we mark such knot values with the superscript s. Coincidently, the knot vector of degree elevated NURBS has the form of:

$$\hat{\mathbf{U}} = \{\underbrace{0,...,0}_{p+2}, \underbrace{u_1^s,...,u_1^s}_{s_1+1},...,\underbrace{u_{\eta-1}^s,...,u_{\eta-1}^s}_{s_{\eta-1}+1}, \underbrace{1,...,1}_{p+2}\}. \quad (15)$$

Although all authors agree upon the form of the knot vector, their approach to the calculation of a control polygon differs greatly. We will refer to the calculation of degree elevated spline control points as to elevation of control points (the control polygon). It is known that elevation of the control point is a convex scheme [7]. This means that the control point $\hat{\mathbf{P}}_j$ is located inside the polygon of adjacent control points $\mathbf{P}_i$:

$$\hat{\mathbf{P}}_j = \sum_{i=0}^{n-1} \gamma_{i,j} \mathbf{P}_i, \; j = 0...\hat{n}, \quad (16)$$

where $\gamma_{i,j}$ is a positive real value that submits to the restriction:

$$\sum_{i=0}^{n-1} \gamma_{i,j} = 1. \quad (17)$$

Notice that $\hat{\mathbf{P}}_j$ is acquired by summing the multiplications of $\gamma_{i,j}$ and $\mathbf{P}_i$. Both arithmetic operations are performed with each coordinate of $\mathbf{P}_i$. So, $\hat{\mathbf{P}}_j$ resides in the same multidimensional model space as $\mathbf{P}_i$. As the solution does not depend on the initial control polygon, any algorithm that can obtain correct $\gamma_{i,j}$ values for 1D model space is correct in cases of

2D, 3D, and 4D model spaces as well. Because of this, we can map NURBS to homogeneous space using (6) before the elevation and ignore NURBS rationality in our analysis of the relation between original and degree elevated spline. All control points must be mapped back to the model space after the elevation using (7).

Currently, there is no unified method that calculates $\gamma_{i,j}$ values. Existing direct and indirect methods vary in complexity and performance. We intend to reveal the relation between $\mathbf{C}(u)$ and $\hat{\mathbf{C}}(u)$ by solving the linear equation system obtained from (13). Plugging (8) into (13) gives:

$$\sum_{i=0}^{n-1} N_{i,p}(u)\mathbf{P}_i = \sum_{j=0}^{\hat{n}-1} \hat{N}_{j,\hat{p}}(u)\hat{\mathbf{P}}_j \,, \tag{18}$$

where $\hat{n}$ is the number of elevated control points, $\hat{N}_{j,\hat{p}}(u)$ is the basis function of NURBS, whose degree was elevated from p to $\hat{p} = p+1$. Actually, only $p+1$ or less of adjacent basis functions have a positive value. Other functions equal zero, therefore can be eliminated (the local support property [7]):

$$\sum_{i=k-p}^{k} N_{i,p}(u)\mathbf{P}_i = \sum_{j=\hat{k}-\hat{p}}^{\hat{k}} \hat{N}_{j,\hat{p}}(u)\hat{\mathbf{P}}_j, \tag{19}$$
$$u_k < u < u_{k+1}, \hat{u}_{\hat{k}} < u < \hat{u}_{\hat{k}+1},$$

$$\sum_{i=k-p}^{k-s_l} N_{i,p}(u)\mathbf{P}_i = \sum_{j=\hat{k}-\hat{p}}^{\hat{k}-\hat{s}_l} \hat{N}_{j,\hat{p}}(u)\hat{\mathbf{P}}_j \,, \tag{20}$$
$$u = u_k = \hat{u}_{\hat{k}},\ l = 1...\eta-1,$$

where $l = \hat{k} - k - 1$, $\hat{s}_l = s_l + 1$, $\hat{u}_{\hat{k}}$ denotes knot from elevated knot vector and s_l denotes knot multiplicity. The equation (19) states that curve segment $u_k < u < u_{k+1}$ is affected by a set of control points from $\mathbf{P}_{k-p}$ to $\mathbf{P}_k$ before the elevation and the set of control points from $\hat{\mathbf{P}}_{\hat{k}-\hat{p}}$ to $\hat{\mathbf{P}}_{\hat{k}}$ after the elevation. The equation (20) defines a special case, when parametric value equals the knot value $u = u_k = \hat{u}_{\hat{k}}$. This special curve point is affected by the same number of control points before and after the elevation (because we set knot multiplicities from (14) to (15) to maintain continuity). This number equals:

$$n_l = p - s_l + 1 \,. \tag{21}$$

Table 1 helps to visualize the situation. As u, p, and $\hat{p}$ values are constant within the limits of the equations (19) and (20), we use N_i instead of $N_{i,p}(u)$ and $\hat{N}_j$ instead of $\hat{N}_{j,\hat{p}}(u)$ for the sake of compactness. Also, we use the acronym PN in the text instead of the multiplication of the control point and the basis function.

Table 1. PN influence on certain segments of NURBS curve

l, k, $\hat{k}$	$l=0$, $k=p$, $\hat{k}=\hat{p}$				$l=1$, $k=p+s_1$, $\hat{k}=\hat{p}+\hat{s}_1$					$l=\eta-1$, $k=n$, $\hat{k}=\hat{n}$			
$\mathbf{P}_i N_i$	$\mathbf{P}_0 N_0$		$\mathbf{P}_p N_p$		$\mathbf{P}_{sl} N_{sl}$		$\mathbf{P}_{p+s_1} N_{p+s_1}$			$\mathbf{P}_{n-p} N_{n-p}$		$\mathbf{P}_n N_n$	
$\hat{\mathbf{P}}_j \hat{N}_j$	$\hat{\mathbf{P}}_0 \hat{N}_0$	…	$\hat{\mathbf{P}}_p \hat{N}_p$	$\hat{\mathbf{P}}_{\hat{p}} \hat{N}_{\hat{p}}$	$\hat{\mathbf{P}}_{\hat{s}_1} \hat{N}_{\hat{s}_1}$	…	$\hat{\mathbf{P}}_{p+\hat{s}_1} \hat{N}_{p+\hat{s}_1}$	$\hat{\mathbf{P}}_{\hat{p}+\hat{s}_1} \hat{N}_{\hat{p}+\hat{s}_1}$	…	$\hat{\mathbf{P}}_{\hat{n}-\hat{p}} \hat{N}_{\hat{n}-\hat{p}}$	…	$\hat{\mathbf{P}}_{\hat{n}-1} \hat{N}_{\hat{n}-1}$	$\hat{\mathbf{P}}_{\hat{n}} \hat{N}_{\hat{n}}$
$\alpha_{l,\hat{p}-\hat{k}+j}$	$\alpha_{0,0}$		$\alpha_{0,p}$	$\alpha_{0,\hat{p}}$	$\alpha_{1,0}$		$\alpha_{1,p}$	$\alpha_{1,\hat{p}}$		$\alpha_{\eta-1,0}$		$\alpha_{\eta-1,p}$	$\alpha_{\eta-1,\hat{p}}$

There are η non-zero knot intervals. Each of them can be indentified either by k in the knot vector $\mathbf{U}$, by $\hat{k}$ in the knot vector $\hat{\mathbf{U}}$ or by l in the knot multiplicity vector $\mathbf{S}$:

$$\mathbf{S} = \{s_0 = p+1, s_1, ..., s_{\eta-1}, s_\eta = p+1\} \,. \tag{22}$$

Every interval (the segment of a curve residing in the model space) has a set of non-zero basis functions. They define the amount of control point influence on a curve within the bounds of a given segment. The second row in Table 1 indicates PN pairs that contribute to the shape of the curve segment before the elevation. The third row refers to PN pairs after the elevation.

Let us presume that knot multiplicities are $s_l = p$, $l = 1...\eta-1$. Then we have NURBS composed of Bézier segments. In such case, we can apply (11) and (12) to degree elevate a spline, just like Piegl and Tiller suggested in [6] and [7]. Then all $\alpha_{l,\hat{p}-\hat{k}+j}$ values given in the fourth row of Table are Bézier coefficients. But in general, NURBS is not composed of Bézier segments.

Normally, the influence of a basis function spreads throughout several segments, so a single control point affects several segments of a curve. This is the reason why NURBS is continuous at segment "welding" points-knots. In fact, NURBS curve is C^{p-s_l} continuous at knots [4] and each knot interval can be different in size (non-uniformity). Because of these inter-segment relations calculation of alpha coefficients is not as simple as in Bézier case. Henceforth, we will refer to alpha values in Table 1 and to gamma in the equation (16) as to elevation coefficients.

Theoretically, elevation coefficients do not depend on the position of control points. They depend on the degree and the knot vector of NURBS. It is much easier to grip the nature of relation between $\mathbf{C}(u)$ and $\hat{\mathbf{C}}(u)$ by examining degree-specific examples. The next subsection discusses the example of NURBS degree elevation from linear to quadratic.

3.4.Degree Elevation of Linear NURBS

We are to elevate NURBS from $p=1$ to $\hat{p}=2$. Let the knot vector of the original spline take the form of:

$$\mathbf{U}=\{0,0,u_1^s,u_2^s,...,u_{\eta-1}^s,1,1\}. \quad (23)$$

It is important to note that knot multiplicities s_l cannot exceed the degree (with the exception of the first and the last knot), otherwise a certain control point will have no affect on the curve:

$$1\le s_l\le p,\ l=1...\eta-1. \quad (24)$$

So, in the linear case, (23) is the only valid form of a clamped normalized knot vector. Let us adapt Table 1 to the linear case (see Table 2) and compose the knot vector of the degree elevated spline:

$$\mathbf{U}=\{0,0,0,u_1^s,u_1^s,u_2^s,u_2^s...,u_{\eta-1}^s,u_{\eta-1}^s,1,1,1\}. \quad (25)$$

Table 2. PN influence on certain segments of the linear NURBS curve

$l=0,\ k=1,\ \hat{k}=2$			$l=1,\ k=2,\ \hat{k}=4$			
$\mathbf{P}_0N_0$	$\mathbf{P}_1N_1$		$\mathbf{P}_1N_1$	$\mathbf{P}_2N_2$		…
$\hat{\mathbf{P}}_0\hat{N}_0$	$\hat{\mathbf{P}}_1\hat{N}_1$	$\hat{\mathbf{P}}_2\hat{N}_2$	$\hat{\mathbf{P}}_2\hat{N}_2$	$\hat{\mathbf{P}}_3\hat{N}_3$	$\hat{\mathbf{P}}_4\hat{N}_4$	
$\alpha_{0,0}$	$\alpha_{0,1}$	$\alpha_{0,2}$	$\alpha_{1,0}$	$\alpha_{1,1}$	$\alpha_{1,2}$	

For the sake of simplicity and compact look, we introduce several useful notations. Let us mark the difference between knots values u_{k+1} and u_k as $D_{l,1}$. So, $D_{l,1}$ denotes the size of the l'th non-zero knot interval. It is convenient to use this notation in order to capture the size of L adjacent non-zero knot intervals:

$$D_{l,L}=u_K-u_k,\ K=k+1+\sum_{i=l+1}^{i<l+L}s_i. \quad (26)$$

Another useful notation for the ratio of knot intervals is taken from [4]:

$$A_{i,j}(u)=(u-u_i)/(u_{i+j}-u_i),\quad k-j\le i\le k,\ 0\le j\le p. \quad (27)$$

The last two notations are used to collapse the subtraction from 1 operation:

$$\overline{A}_{i,p}(u)=1-A_{i,p}(u),\quad k-j\le i\le k,\ 0\le j\le p. \quad (28)$$

$$\overline{\alpha}_{i,j}=1-\alpha_{i,j},\ 0\le i\le\eta-1,\ 0\le j\le\hat{p}. \quad (29)$$

Let us go back to the degree elevation of linear NURBS. Setting u to zero collapses equation (20) to $\mathbf{P}_0N_0=\hat{\mathbf{P}}_0\hat{N}_0$, $N_0=\hat{N}_0=1$. Therefore, we get:

$$\mathbf{P}_0=\hat{\mathbf{P}}_0. \quad (30)$$

This relation is correct for an arbitrary degree because of the clamping. An analogous relation can be declared for the last control point:

$$\mathbf{P}_n=\hat{\mathbf{P}}_{\hat{n}}. \quad (31)$$

Let us move forward into the first non-zero interval $0<u<u_1^s$. Now the equation (19) can be rewritten as:

$$\mathbf{P}_0N_0+\mathbf{P}_1N_1=\hat{\mathbf{P}}_0\hat{N}_0+\hat{\mathbf{P}}_1\hat{N}_1+\hat{\mathbf{P}}_2\hat{N}_2. \quad (32)$$

This equation can be decomposed into the system of linear equations. In order to do this, one must formulate the relation between $\mathbf{P}_i$ and $\hat{\mathbf{P}}_j$. It can be accomplished by applying the corner cutting scheme given in (11) for each knot interval separately. Hence, $\hat{\mathbf{P}}_j$ relates to $\mathbf{P}_i$ in the following fashion:

$$\hat{\mathbf{P}}_0=\overline{\alpha}_{0,0}\mathbf{P}_0, \quad (33)$$

$$\hat{\mathbf{P}}_1=\alpha_{0,1}\mathbf{P}_0+\overline{\alpha}_{0,1}\mathbf{P}_1, \quad (34)$$

$$\hat{\mathbf{P}}_2=(\alpha_{0,2}\mathbf{P}_1+\overline{\alpha}_{0,2}\mathbf{P}_2)D_{0,1}/D_{0,2}+(\alpha_{1,0}\mathbf{P}_0+\overline{\alpha}_{1,0}\mathbf{P}_1)D_{1,1}/D_{0,2}. \quad (35)$$

Notice that $\hat{\mathbf{P}}_2\hat{N}_2$ contributes to both intervals: $D_{0,1}$ and $D_{1,1}$. In order to maintain the restriction of the convex scheme (see the equation (17)), it is necessary to multiply elevation coefficients from $D_{0,1}$ by ratio $D_{0,1}/(D_{0,1}+D_{1,1})$ and coefficients from $D_{1,1}$ by $D_{1,1}/(D_{0,1}+D_{1,1})$. Keep in mind that $D_{0,1}+D_{1,1}=D_{0,2}$. Hence, we get the expression (35). Grouping elements of (35) by $\mathbf{P}_i$ yields:

$$\hat{\mathbf{P}}_2=\alpha_{1,0}\frac{D_{1,1}}{D_{0,2}}\mathbf{P}_0+\left(\alpha_{0,2}\frac{D_{0,1}}{D_{0,2}}+\overline{\alpha}_{1,0}\frac{D_{1,1}}{D_{0,2}}\right)\mathbf{P}_1+\overline{\alpha}_{0,2}\frac{D_{0,1}}{D_{0,2}}\mathbf{P}_2. \quad (36)$$

As $N_2=0$ at $0<u<u_1^s$, we can ignore $\mathbf{P}_2$ group in (36). From (30) and (33), we can tell that $\overline{\alpha}_{0,0}=1$, thus $\alpha_{0,0}=0$. So, plugging (33), (34) and (36) into (32) gives:

$$\mathbf{P}_0N_0+\mathbf{P}_1N_1=\left(\hat{N}_0+\alpha_{0,1}\hat{N}_1+\frac{\alpha_{1,0}D_{1,1}}{D_{0,2}}\hat{N}_2\right)\mathbf{P}_0+\left(\overline{\alpha}_{0,1}\hat{N}_1+\frac{\alpha_{0,2}D_{0,1}+\overline{\alpha}_{1,0}D_{1,1}}{D_{0,2}}\hat{N}_2\right)\mathbf{P}_1. \quad (37)$$

It is easy to decompose (37) into the system of $p+1$ linear equations. In this case, there are two equations in the system. The first is extracted from $\mathbf{P}_0$ group, and the second is extracted from $\mathbf{P}_1$ group:

$$\begin{cases} N_0 = \hat{N}_0 + \alpha_{0,1}\hat{N}_1 + \alpha_{1,0}D_{1,1}\hat{N}_2 / D_{0,2} \\ N_1 = \overline{\alpha}_{0,1}\hat{N}_1 + (\alpha_{0,2}D_{0,1} + \overline{\alpha}_{1,0}D_{1,1})\hat{N}_2 / D_{0,2} \end{cases}. \quad (38)$$

The basis functions can be decomposed using Cox-de Boor formula (3) and converted into the form of (26). The first of equations in (38) can be expressed as:

$$\begin{aligned} &D_{0,2}(D_{0,1} - d) = 2\alpha_{0,1}D_{0,2}(D_{0,1} - d) + \alpha_{1,0}dD_{1,1}. \\ &d = u - u_k. \end{aligned} \quad (39)$$

Now we have a single equation with two unknown coefficients, but we can use two different values of d to obtain solutions for $\alpha_{0,1}$ and $\alpha_{1,0}$. If (39) is interpreted as (40), then coefficients can be calculated from (41) and (42):

$$C = aA + bB, \quad (40)$$

$$a = \frac{C_1B_2 - C_2B_1}{A_1B_2 - A_2B_1}, \quad (41)$$

$$b = \frac{A_1C_2 - A_2C_1}{A_1B_2 - A_2B_1}. \quad (42)$$

Due to given $a = \alpha_{0,1}$, $b = \alpha_{1,0}$, $A_1 = 2D_{0,2}(D_{0,1} - d_1)$, $B_1 = d_1D_{1,1}$, $C_1 = D_{0,2}(D_{0,1} - d_1)$, $A_2 = 2D_{0,2}(D_{0,1} - d_2)$, $B_2 = d_2D_{1,1}$, and $C_2 = D_{0,2}(D_{0,1} - d_2)$, the solutions are $\alpha_{0,1} = 1/2$ and $\alpha_{1,0} = 0$. Solving the second equation in (38) yields $\alpha_{0,2} = 1$. In the linear case it is not necessary to repeat the calculations with the remaining intervals, because each interval produces the same result (inspect Table 3).

Table 3. Alpha elevation coefficient values for the linear NURBS curve

$\alpha_{0,0}$	$\alpha_{0,1}$	$\alpha_{0,2}$	$\alpha_{1,0}$	$\alpha_{1,1}$	$\alpha_{1,2}$	…
0	1/2	1	0	1/2	1	

Notice that alpha values are Bézier coefficients (see Subsection 0). Inserting these values into (33), (34) and (36) produces $\hat{\mathbf{P}}_0 = \mathbf{P}_0$, $\hat{\mathbf{P}}_1 = (\mathbf{P}_0 + \mathbf{P}_1)/2$ and $\hat{\mathbf{P}}_2 = \mathbf{P}_1$. This corresponds to (12) and there is no accident. Clamped linear NURBS is composed of Bézier segments and is C^0 continuous at knots. Hence, Piegl and Tiller's algorithm [6, 7] does not produce unnecessary knots that need to be removed.

In the cases of quadratic, cubic and quartic NURBS we intend to use the same corner cutting scheme for each non-zero knot interval and calculate elevation coefficient values by solving the linear equation system, obtained either from (19) or from (20). Also, the values of the first and the last elevation coefficient in the corner cutting scheme are valid for an arbitrary degree:

$$\alpha_{l,0} = 0, \quad (43)$$

$$\alpha_{l,\hat{p}} = 1. \quad (44)$$

3.5. Degree Elevation of Quadratic NURBS

The elevation from $p = 2$ to $\hat{p} = 3$ slightly differs from the linear case. Since knot multiplicities can be set either to $s_l = 1$ or to $s_l = 2$ (inspect (24)), the knot vector has more than one valid form. For now, let us consider $s_l = 1$:

$$\mathbf{U} = \{0,0,0,u_1^s, u_2^s, ..., u_{\eta-1}^s, 1,1,1\}. \quad (45)$$

By following the procedure provided in Subsection 0 we are able to extract the linear equation system from the first non-zero knot interval $u_2 < u < u_3$ (or $0 < u < u_1^s$):

$$\begin{cases} N_0 = \hat{N}_0 + \alpha_{0,1}\hat{N}_1 \\ N_1 = 2\hat{N}_1/3 + (\alpha_{0,2}D_{0,1} + D_{1,1})\hat{N}_2 / D_{0,2} + \\ \qquad + \alpha_{1,1}\hat{N}_3D_{1,1} / D_{0,2} \\ N_2 = \overline{\alpha}_{0,2}D_{0,1}\hat{N}_2 / D_{0,2} + (D_{0,1} + \overline{\alpha}_{1,1}D_{1,1})\hat{N}_3 / D_{0,2} \end{cases}. \quad (46)$$

A portion of equation system solutions is provided in Table 4.

Table 4. Alpha elevation coefficient values for the quadratic NURBS curve

$\alpha_{0,0}$	$\alpha_{0,1}$	$\alpha_{0,2}$	$\alpha_{0,3}$	$\alpha_{1,0}$	$\alpha_{1,1}$	$\alpha_{1,2}$	$\alpha_{1,3}$	…
0	1/3	2/3	1	0	1/3	2/3	1	

Notice, that alpha values are Bézier coefficients and can be calculated using (12). Despite this fact, the quadratic case is different from the linear case. As interval ratios $D_{0,1}/D_{0,2}$ and $D_{1,1}/D_{0,2}$ are not eliminated, positions of $\hat{\mathbf{P}}_2$ and $\hat{\mathbf{P}}_3$ are obtained from:

$$\hat{\mathbf{P}}_2 = \left(\alpha_{0,2}\frac{D_{0,1}}{D_{0,2}} + \frac{D_{1,1}}{D_{0,2}}\right)\mathbf{P}_1 + \left(\overline{\alpha}_{0,2}\frac{D_{0,1}}{D_{0,2}}\right)\mathbf{P}_2, \quad (47)$$

$$\hat{\mathbf{P}}_3 = \left(\alpha_{1,1}\frac{D_{1,1}}{D_{0,2}}\right)\mathbf{P}_1 + \left(\frac{D_{0,1}}{D_{0,2}} + \overline{\alpha}_{1,1}\frac{D_{1,1}}{D_{0,2}}\right)\mathbf{P}_2. \quad (48)$$

Equations for $\hat{\mathbf{P}}_0$ and $\hat{\mathbf{P}}_1$ are the same as in the linear case (see (33) and (34)). In order to make expressions (47) and (48) more compact, we introduce another type of elevation coefficients:

$$\beta_{l,\hat{p}-\hat{k}+j} = \alpha_{l,\hat{p}-\hat{k}+j}\frac{D_{l,1}}{\hat{u}_{j+\hat{p}+1} - \hat{u}_j}, \quad (49)$$

$$\tilde{\beta}_{l,\hat{p}-\hat{k}+j} = \overline{\alpha}_{l,\hat{p}-\hat{k}+j}\frac{D_{l,1}}{\hat{u}_{j+\hat{p}+1} - \hat{u}_j}, \quad (50)$$

where $\hat{u}_j$ represents a knot from $\hat{\mathbf{U}}$, $j = 0...\hat{n}$. Plainly speaking, the coefficient beta is acquired by multiplying the coefficient alpha with the same indices multiplied by the knot-interval ratio. The numerator of the ratio indicates the size of the knot interval to which coefficient alpha belongs and the denominator denotes the size of all knot intervals where $N_j \neq 0$. Pay attention to the fact that $\tilde{\beta} \neq 1 - \beta$, when the knot

interval ratio is less than 1. Now (47) and (48) can be rewritten as:

$$\hat{\mathbf{P}}_2 = \left(\beta_{0,2} + \tilde{\beta}_{1,0}\right)\mathbf{P}_1 + \left(\tilde{\beta}_{0,2}\right)\mathbf{P}_2 = \gamma_{1,2}\mathbf{P}_1 + \gamma_{2,2}\mathbf{P}_2, \quad (51)$$

$$\hat{\mathbf{P}}_3 = \left(\beta_{1,1}\right)\mathbf{P}_1 + \left(\beta_{0,3} + \tilde{\beta}_{1,1}\right)\mathbf{P}_2 = \gamma_{1,3}\mathbf{P}_1 + \gamma_{2,3}\mathbf{P}_2. \quad (52)$$

If the knot interval form **U** has double knots at both ends, this makes a respective segment of the curve a Bézier segment. This case is the same as the linear case. Let us move on to the case of cubic NURBS.

3.5. Degree Elevation of Cubic NURBS

The degree elevation of cubic NURBS introduces another new feature. Let $s_l = 1$, $l = 1...\eta - 1$. Then extraction of the linear equation system within the bounds of the first interval $u_3 < u < u_4$ yields:

$$\begin{cases} N_0 = \hat{N}_0 + \alpha_{0,1}\hat{N}_1 \\ \ldots \\ N_3 = D_{0,1}\hat{N}_4 / D_{0,3} + \bar{\alpha}_{1,2}D_{1,1}\hat{N}_4 / D_{0,3} + \\ \quad + \bar{\alpha}_{0,3}D_{0,1}\hat{N}_3 / D_{0,2} \end{cases}. \quad (53)$$

We would like to underline the fact that the first and the last equations in the system always contain two or less coefficients. We will refer to these equations as to short equations in the remaining sections. Short equations are easy to solve using the form of (40) and then applying (41) and (42). Moreover, in the cubic case short equations are sufficient to determine all elevation coefficients. Some of alphas are provided in Table 5.

Table 5. Alpha elevation coefficient values for the cubic NURBS curve

$\alpha_{0,0}$	$\alpha_{0,1}$	$\alpha_{0,2}$	$\alpha_{0,3}$	$\alpha_{0,4}$	$\alpha_{1,0}$	$\alpha_{1,1}$	$\alpha_{1,2}$	$\alpha_{1,3}$	$\alpha_{1,4}$	$\alpha_{2,0}$	$\alpha_{2,1}$	$\alpha_{2,2}$	…
0	1/4	2/4	…	1	0	1/4	2/4	…	1	0	…	2/4	

Certain values like $\alpha_{0,3}$, $\alpha_{1,3}$, and $\alpha_{2,1}$ are not constant and do depend on the size of knot intervals:

$$\alpha_{0,3} = 1 - \frac{D_{0,2}}{4D_{0,3}} = 1 - \frac{N_{3,3}(u_4)}{4\hat{N}_{4,4}(\hat{u}_6)}, \quad (54)$$

$$\alpha_{1,3} = 1 - \frac{D_{1,2}}{4D_{1,3}} = 1 - \frac{N_{4,3}(u_5)}{4\hat{N}_{6,4}(\hat{u}_8)}, \quad (55)$$

$$\alpha_{2,1} = \frac{D_{1,2}}{4D_{0,3}} = \frac{N_{2,3}(u_6)}{4\hat{N}_{4,4}(\hat{u}_{10})}. \quad (56)$$

Using this pattern in conjunction with the fact that there are two short equations with $\alpha_{l,1}$ and $\alpha_{l,\hat{p}}$ for each non-zero knot interval, we can claim that:

$$\alpha_{l,1} = \frac{1}{\hat{p}}\frac{N_{k-p,p}(u_k)}{\hat{N}_{\hat{k}-\hat{p},\hat{p}}(\hat{u}_{\hat{k}})}, \quad (57)$$

$$\bar{\alpha}_{l,p} = \frac{1}{\hat{p}}\frac{N_{k-s_l,p}(u_k)}{\hat{N}_{\hat{k}-\hat{s}_l,\hat{p}}(\hat{u}_{\hat{k}})}, \quad (58)$$

where $u_k = \hat{u}_{\hat{k}}$, $l = \hat{k} - k - 1$ and $l = 1...\eta - 1$. These relations are valid for the arbitrary degree and this is the reason why they are very important. So, in the cubic case the first alpha is 0, the middle alpha is 1/2, and the last alpha is 1. The remaining alphas can be obtained from (57) and (58).

3.6. Degree Elevation of Quartic NURBS

It is not difficult to see that the complexity of degree elevation grows altogether with continuity at knots. The quartic NURBS maintains C^3 continuity at knots with the multiplicity of 1. So, let us assume that all knot multiplicities are $s_l = 1$ for all $l = 1...\eta - 1$. Equations (43), (44), (57), and (58) allow us to acquire all alphas except $\alpha_{l,2}$ and $\alpha_{l,3}$. Those values can be calculated from short equations extracted at knot values $u = u_k = \hat{u}_{\hat{k}}$ with the exception of $\alpha_{0,3}$ and $\alpha_{\eta-1,2}$. Let us take the example of

$$\alpha_{1,2} : \alpha_{1,2} = \frac{D_{0,3}N_2(u_6) - \alpha_{2,1}D_{2,1}\hat{N}_5(\hat{u}_9)}{D_{1,1}\hat{N}_4(\hat{u}_9)} - \frac{D_{2,1}}{D_{1,1}}. \quad (59)$$

This equation has a fairly large number of elements. Also, it turns out that short equations at $u = u_k$ define the relation between $\alpha_{l,1}$ and $\alpha_{l-1,\hat{s}_l}$ as well as between $\bar{\alpha}_{l-1,p}$ and $\bar{\alpha}_{l,\hat{p}-\hat{s}_l}$, where $\hat{s}_l = s_l + 1$. Notice the sliding index that depends on s_l. Because of this, there are cases when $\alpha_{l,2}$ and $\alpha_{l,3}$ cannot be obtained from short equations. So, we must turn to long equations.

Recall that our final objective is to acquire gamma elevation coefficients to calculate $\hat{\mathbf{P}}_j$ (see (16)). This is the essential idea of the direct degree elevation. In fact, if all alphas are known, we can easily calculate betas (see (49) and (50)) and then gammas (inspect examples (51) and (52)), but that is a redundant process.

In preceding subsections we provided alpha values to illustrate the difference between the degree elevation of Bézier segment and NURBS segment. Henceforth, we will focus our attention to the direct acquisition of gamma elevation coefficients. Let us take the example of $\gamma_{3,3}$:

$$\gamma_{3,3} = \tilde{\beta}_{0,3} = \bar{\alpha}_{0,3}D_{0,1} / D_{0,2}. \quad (60)$$

Coefficient $\gamma_{3,3}$ is obtained directly from the long equation of the group $\mathbf{P}_3$ at $u_4 < u < u_5$ and $u = u_5$:

$$\gamma_{3,3} = \frac{\left(N_3(u) - \gamma_{3,5}\hat{N}_5(u)\right)\hat{N}_4(\hat{u}_7) - \left(N_3(u_5) - \gamma_{3,5}\hat{N}_5(\hat{u}_7)\right)\hat{N}_4(u)}{\hat{N}_3(u)\hat{N}_4(\hat{u}_7) - \hat{N}_3(\hat{u}_7)\hat{N}_4(u)}. \quad (61)$$

Now we will illustrate the methodology that simplifies the solution process:

1. The basis functions must be expressed by the means of ratios (27) and (28). In this specific case there are four different ratios. Let us denote them by $A_1 = A_{4,1}(u)$, $A_2 = A_{4,2}(u)$, $A_3 = A_{4,3}(u)$, and $A_4 = A_{4,4}(u)$. So, the list of necessary basis functions consists of:

$$N_3(u) = A_1 A_2 A_3 (\overline{A}_1 + \overline{A}_2 + \overline{A}_3 + \overline{A}_4), \quad (62)$$

$$\hat{N}_3(u) = A_1^2 A_2 (3\overline{A}_1^2 + 4\overline{A}_1\overline{A}_2 + 3\overline{A}_2^2), \quad (63)$$

$$\hat{N}_4(u) = A_1^2 A_2^2 (2\overline{A}_1 + 2\overline{A}_2 + \overline{A}_3), \quad (64)$$

$$\hat{N}_5(u) = A_1^2 A_2^2 A_3, \quad (65)$$

$$N_3(u_5) = A_2 A_3 (3A_1 - A_2 - A_3 - A_4) / A_1^3, \quad (66)$$

$$\hat{N}_3(\hat{u}_7) = 3A_2 (A_1 - A_2)^2 / A_1^3, \quad (67)$$

$$\hat{N}_4(\hat{u}_7) = A_2^2 (3A_1 - 2A_2 - A_3) / A_1^3, \quad (68)$$

$$\hat{N}_5(\hat{u}_7) = A_2^2 A_3 / A_1^3. \quad (69)$$

2. Gamma in the right side of the equation is obtained from the series of the short equations. In this case $\gamma_{3,5} = \beta_{1,3} + \tilde{\beta}_{2,1}$ is obtained from short equations extracted from (20) at $u = u_5$ and $u = u_6$. The gamma value must be expressed by the means of (27) and (28):

$$\gamma_{3,5} = 1 - \frac{(A_1 - A_3)(A_2 - A_3) + A_4(2A_1 + 2A_2 + A_3)}{5A_1A_2}. \quad (70)$$

3. Gamma and basis function values must be plugged into the long equation and solved. In this case plugging (62) – (70) into (61) produces huge expression. Luckily it collapses to:

$$\gamma_{3,3} = \frac{2A_3}{5A_1}\frac{\overline{A}_2^2 + 2\overline{A}_2\overline{A}_3 - 3\overline{A}_1^2}{\overline{A}_2^2 + 2\overline{A}_2\overline{A}_3 - 3\overline{A}_1^2} = \frac{2}{5}\frac{A_3}{A_1} = \frac{2}{5}\frac{D_{0,1}}{D_{0,3}}. \quad (71)$$

We know that $\gamma_{3,3} = \tilde{\beta}_{0,3} = \overline{\alpha}_{0,3} D_{0,1} / D_{0,2}$, so we can easily calculate $\overline{\alpha}_{0,3}$:

$$\overline{\alpha}_{0,3} = \frac{2}{5}\frac{D_{0,2}}{D_{0,3}}. \quad (72)$$

So actually, coefficient $\gamma_{3,3}$ we have obtained from (61) is the reduced form of:

$$\gamma_{3,3} = \frac{2}{5}\frac{D_{0,1}}{D_{0,2}}\frac{D_{0,2}}{D_{0,3}}. \quad (73)$$

Now let us take a look at the series of gammas at the left clamped end of the knot vector in Table 6.

Table 6. Gamma elevation coefficient values for the quartic NURBS curve

$\gamma_{0,0}$	$\gamma_{1,1}$	$\gamma_{2,2}$	$\gamma_{3,3}$	$\gamma_{4,4}$	
$\frac{5}{5}\frac{D_{0,1}}{D_{0,1}}\frac{1}{1}$	$\frac{4}{5}\frac{D_{0,1}}{D_{0,1}}\frac{1}{1}$	$\frac{3}{5}\frac{D_{0,1}}{D_{0,2}}\frac{D_{0,2}}{D_{0,2}}$	$\frac{2}{5}\frac{D_{0,1}}{D_{0,2}}\frac{D_{0,2}}{D_{0,3}}$	$\frac{1}{5}\frac{D_{0,1}}{D_{0,3}}\frac{D_{0,2}}{D_{0,4}}$	...

Each gamma coefficient in Table 6 is combined from three ratios. The first one corresponds to Bézier coefficient in (12). The second one illustrates the knot interval ratio we used in (49) and (50). The third one is obtained from the basis function ratio similar to the ratios we used in (57) and (58): $N_{4,4}(u_5) / \hat{N}_{5,5}(u_5)$ for $\gamma_{4,4}$, $N_{4,3}(u_5) / \hat{N}_{5,4}(u_5)$ for $\gamma_{3,3}$, $N_{4,2}(u_5) / \hat{N}_{5,3}(u_5)$ for $\gamma_{2,2}$, and so on. It is easy to spot that the degree of basis functions is smaller when the difference between $\hat{k}$ and j is greater. According to this pattern, we are able to compose a generic form of $\gamma_{i,j}$. Let the active knot interval be indentified by a triplet: k position in $\mathbf{U}$, $\hat{k}$ position in $\hat{\mathbf{U}}$, and l position in $\mathbf{S}$. For all $k - s_l < i \le k$ and $\hat{k} - \hat{s}_l < j \le \hat{k}$ there is a valid relation:

$$\gamma_{i,j} = \frac{\hat{k} - j}{\hat{p}}\frac{\hat{u}_{\hat{k}+1} - \hat{u}_{\hat{k}}}{\hat{u}_{j+\hat{p}+1} - \hat{u}_j}\frac{N_{k,p-\hat{k}+j}(u_k)}{\hat{N}_{\hat{k},\hat{p}-\hat{k}+j}(\hat{u}_{\hat{k}})}, \quad (74)$$
$$j < \hat{k},\ i = j - l,$$

where $u_k = \hat{u}_{\hat{k}}$. In the case of $j = \hat{k}$, gamma can be obtained from the short equation:

$$\gamma_{i,j} = \frac{N_{k,p}(u_k) - \gamma_{i,j-1}\hat{N}_{\hat{k}-1,\hat{p}}(\hat{u}_{\hat{k}})}{\hat{N}_{\hat{k},\hat{p}}(\hat{u}_{\hat{k}})}, \quad (75)$$
$$j = \hat{k},\ i = k.$$

These relations are the very essence of the direct degree elevation of NURBS. They tell how to obtain a portion of elevation coefficients at an arbitrary knot interval. In the quartic case there are maximum three non-zero gammas. The equations (74) and (75) allow acquisition of the last non-zero gamma. The first non-zero gamma can be obtained from the following relation and the short equation:

$$\gamma_{i,j} = \frac{\hat{p} - \hat{k} + j}{\hat{p}}\frac{\hat{u}_{\hat{k}+1} - \hat{u}_{\hat{k}}}{\hat{u}_{j+\hat{p}+1} - \hat{u}_j}\frac{N_{k-p,\hat{k}-j}(u_k)}{\hat{N}_{\hat{k}-\hat{p},\hat{k}-j+1}(\hat{u}_{\hat{k}})}, \quad (76)$$
$$j > \hat{k} - \hat{p},\ i = j - l - 1,$$

$$\gamma_{i,j} = \frac{N_{k-p,p}(u_k) - \gamma_{i,j+1}\hat{N}_{\hat{k}-\hat{p}+1,\hat{p}}(\hat{u}_{\hat{k}})}{\hat{N}_{\hat{k}-\hat{p},\hat{p}}(\hat{u}_{\hat{k}})}, \quad (77)$$
$$j = \hat{k} - \hat{p},\ i = k - p.$$

We would like a reader to pay attention to the fact that all gammas for a single $\hat{\mathbf{P}}_j$ may be obtained from different knot intervals. Thus triplet k, $\hat{k}$, and l will have different values for (74) and (76), as well as for (75) and (77). The acquisition of the middle gamma is

the most difficult. Therefore we suggest the usage of convex scheme restriction (17).

Information gathered in this section confirms that elevation coefficients depend on the degree and the knot vector of NURBS. According to the expressions from (74) to (77), there is no need to solve the equation system obtained from (19). So, information acquired from (20) is sufficient to determine all necessary gammas. Furthermore, basis functions from (20) are significantly simpler and their ratios can be downgraded easily. As it is quite difficult to understand the simplicity of the concept from expressions filled with various indices, we present a direct degree elevation scheme in Subsection 4.1.

4. DDE Algorithms and Implementation

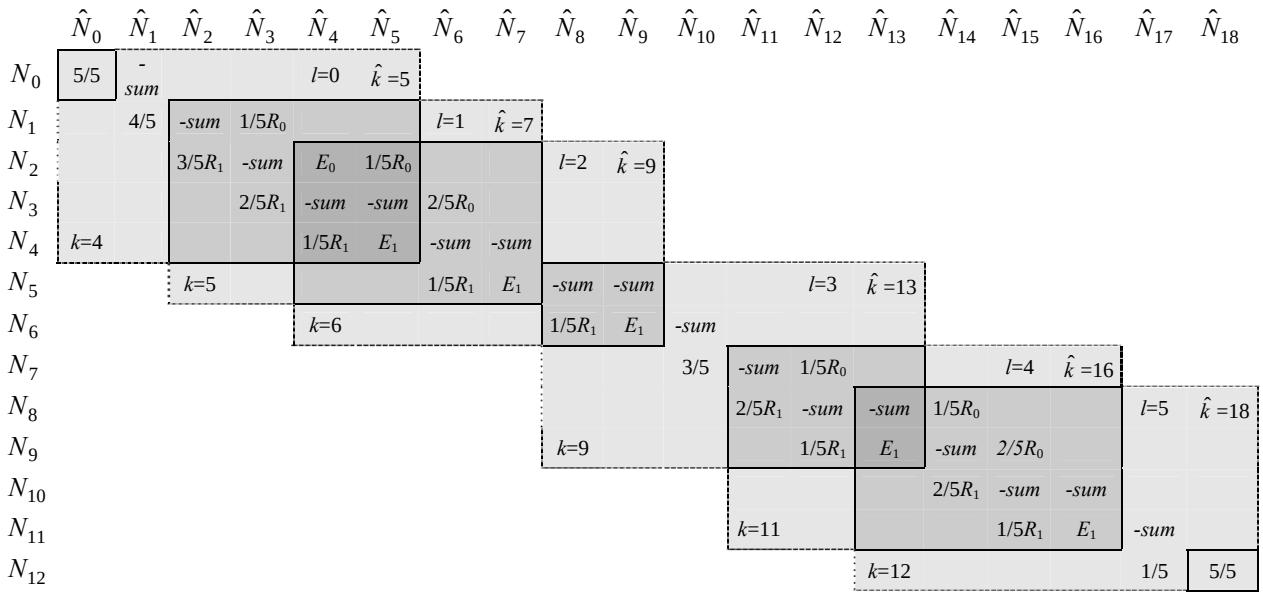
In this section we present the direct degree elevation scheme (see Subsection 4.1). We also provide several technical remarks regarding the implementation (see Subsection 4.2) and algorithm pseudo codes (see Subsections 4.3, 4.4, and 4.5).

4.1. Direct Degree Elevation Scheme

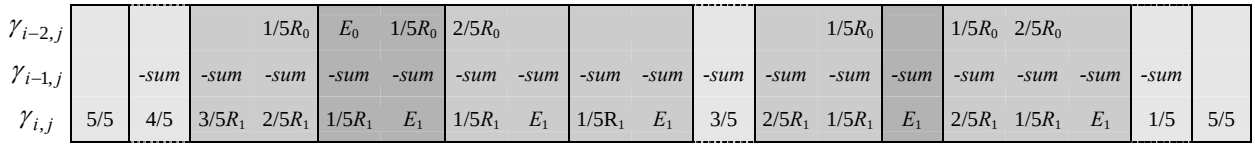
Say we are to elevate the degree of the quartic NURBS defined by the knot vector with various knot multiplicities:

$$\mathbf{U} = \{0,0,0,0,0,u_1^s,u_2^s,u_3^s,u_3^s,u_3^s,u_4^s,u_4^s,u_5^s,1,1,1,1,1\}. \quad (78)$$

Let us denote the multiplication of the second and the last ratio in (74) by R_1, and in (76) by R_0. Also, let E_1 denote the short equation (75) and let E_0 denote the short equation (77). Figure 2 explains how elevation coefficients are acquired from the relations between the basis functions of original NURBS and degree elevated NURBS, where $-sum$ symbolizes the subtraction of non-zero gammas from one (convex scheme restriction (17)).



(a) Matrix representation of gamma elevation coefficients



(b) Compact representation of gamma elevation coefficients

Figure 2. DDE scheme

Figure 2 (a) illustrates the relation between the basis functions and gamma elevation coefficients in the form of a single gamma matrix $\mathbf{\Gamma}$. Each set of non-zero basis functions at $u_k < u < u_{k+1}$ is represented by a light gray rectangle with a dashed outline. The overlapping area of two adjacent sets is depicted by a darker gray rectangle with a solid outline. Coincidentally, two adjacent sets of non-zero functions at $u_{k-s_l} < u < u_k$ and $u_k < u < u_{k+1}$ overlap at $u = u_k$. The darkest gray areas indicate the basis functions that have non-zero value in three adjacent sets.

The elements of $\mathbf{\Gamma}$ indicate the method used to calculate $\gamma_{i,j} = [\mathbf{\Gamma}_{i,j}]$. The non-zero elements of a row induce to the restriction:

$$N_i = \sum_{j=0}^{\hat{n}-1} \gamma_{i,j} \hat{N}_j \quad . \quad (79)$$

In words, one row holds a single equation. For example, the row N_1 describes the equation (80) at $u_4 < u < u_5$, which is reduced to equation (81) at $u = u_5$:

$$N_1 = \gamma_{1,1}\hat{N}_1 + \gamma_{1,2}\hat{N}_2 + \gamma_{1,3}\hat{N}_3,\ u_4 < u < u_5, \quad (80)$$

$$N_1 = \gamma_{1,2}\hat{N}_2 + \gamma_{1,3}\hat{N}_3,\ u = u_5. \quad (81)$$

Every column submits to the restriction (17). Therefore, we do not have to calculate pre-last gamma, because it can be obtained from (17).

It is quite obvious that matrix representation of gamma coefficients mostly consists of zeros, because the maximum number of gamma elevation coefficients for a single control point $\hat{\mathbf{P}}_j$ is:

$$n_{\gamma\max} = \left\lceil \frac{\hat{p}+1}{2} \right\rceil, \quad (82)$$

where brackets $\lceil\ \rceil$ denote the ceiling operation (rounding to a greater integer). Hence, the matrix form is inefficient. Therefore, we suggest a compact form given in Fig (b). In fact, the elevation coefficients for each $\hat{\mathbf{P}}_j$ can be obtained on-the-fly and used in (16) to acquire the position of $\hat{\mathbf{P}}_j$. The idea is to calculate the last non-zero gamma using either (74) or (75) while traversing through knot intervals. If there are three non-zero gammas, the first one is obtained either from (76) or from (77). The pre-last gamma is obtained from (17). The calculation of the third gamma is necessary when:

$$\hat{k} + \hat{s}_l - \hat{p} < j < \hat{k}. \quad (83)$$

A simplified DDE scheme includes the following steps:

1. Initialize $k := -1$, $\hat{k} := -1$
2. For all $l := 1...\eta - 1$ do

   2.1. Set $k := k + s_{l-1}$ and $\hat{k} := \hat{k} + s_{l-1} + 1$

   2.2. For $j := \hat{k} - s_{l-1}...\hat{k}$ do

      2.2.1. Set $i := j - l + 1$ and $\gamma_{i-1,j} := 1$

      2.2.2. If $j = \hat{k}$, then set $i := i - 1$

      2.2.3. Calculate $\gamma_{i,j}$ using (74) or (75) and set $\gamma_{i-1,j} := \gamma_{i-1,j} - \gamma_{i,j}$

      2.2.4. If (83) is satisfied, then calculate $\gamma_{i-2,j}$ using (76) or (77) and set $\gamma_{i-1,j} := \gamma_{i-1,j} - \gamma_{i-2,j}$

      2.2.5. Calculate $\hat{\mathbf{P}}_j$ from (16)

4.2. Technical Remarks

The performance of algorithms depends on the chosen platform, the amount of arithmetic calculations and data handling technique. We have chosen to implement our algorithms on the basis of .NET framework using C# programming language. This subsection consists of several observations regarding platform-specific implementation.

Experiments indicate that minimization of arithmetic operations is not the only crucial factor that affects the performance. Memory allocation overhead may reduce the performance several times. .NET has an option to allocate a portion of memory either statically or dynamically. One must avoid allocating additional portions of memory during the calculation cycles for it may take more CPU time than calculation itself. The number of local variables is also important, but it plays a minor role.

Conversions between static arrays and *List* collections should be avoided at any time. This procedure requires the framework to allocate additional memory and initialize additional cycles to transport data. The best bet is the static array when the number of elements is known. Also, *List* collection produces reasonable overhead and is useful when allocated memory is filled with data in a consistent manner. The last element can be introduced by simply using the method *Add()* and there is no need to track the index of the last element. Methods, like *Insert()* should be avoided for the rearrangement of the elements takes a lot of time.

Apart from the performance factor, we also consider the simplicity of algorithms. This property is defined by the number of code lines and the code structure. Essentially, there are two different strategies for the degree elevation problem: an arithmetic approach and a special case scenario. The arithmetic approach offers more elegant code structure because of fewer conditional sentences. However, the special case scenario performs faster because of the lesser number of arithmetic calculations.

4.3. Elevation of NURBS Knot Vector

In this paper we refer to the elevation of the knot vector as to the procedure that produces the knot vector of degree elevated NURBS curve. According to Section 3.3, we propose the following knot vector elevation algorithm. In addition, this algorithm produces the multiplicity vector in the form of (22).

```
Procedure ElevateKnots
// in: p – degree
// in: U[m] – knot vector
// out: S[ns] – knot multiplicity vector
// out: V[r] – elevated knot vector (worst case r = 2*(n+3))
begin
   n = m - p - 1;  k = 0;
   r = 0; l = 1; ns = 0;
   while (k < m-1) do begin
       V[r] = U[k]; r = r + 1;
       if (U[k] != U[k+1]) then
          V[r] = U[k]; r = r + 1; S[ns] = l;
          ns = ns + 1; l = 0;
       endif;
       l = l + 1;  k = k + 1;
   endwhile;
   S[ns] = l; ns = ns + 1; V[r] = U[m-1];
   r = r + 2; V[r-1] = U[m-1];
end.
```

4.4. Elevation of NURBS Control Polygon

As we have shown in Section 3.4, the linear case of clamped NURBS has a single knot vector form and the only variable is the number of knots. Therefore, we suggest purely arithmetic solution for this problem. We also compare our algorithm against the algorithm published by Cohen, Lyche and Schumaker in [1]. Both algorithms calculate the control point positions of degree elevated spline.

```
Procedure DDELinear
// in: P[n] – control points
// out: Q[2*n-1] – elevated control points
begin
   for i = 0 to n-2 do begin
       Q[2*i] = P[i];
       Q[2*i+1] = (P[i] + P[i+1])/2;
   endfor;
   Q[2*n-2] = P[n-1];
end.
```

```
Procedure CLSLinear
// in: U[m] – knot vector
// in: P[n] – control points
// out: Q[p+1] – elevated control points
begin
   p = 1;
   if (U[0] < U[1]) then
       Q[1] = P[0]/2; p = p + 1;
   endif;
   Q[p] = P[0];
   for i = 1 to n-1 do begin
       if (U[i] < U[i+1]) then
          p = p + 1;
          Q[p] = (P[i] + P[i-1])/2;
       endif;
       p = p + 1; Q[p] = P[i];
   endfor;
   if (U[n+1] > U[n]) then
       p = p + 1; Q[p] = P[n-1]/2;
   endif;
end.
```

```
Procedure DDEQuadratic
// in: U[m] – knot vector
// in: P[n] – control points
// in: S[ns] – knot multiplicity vector
// out: Q[n+ns-1] – elevated control points
begin
   k = 2; B = 1; b1 = 1/3; b2 = 2/3;
   Q[0] = P[0];
   for l = 1 to nl – 1 do begin
       if (S[l-1] > 1) then
         Q[k+l-2] = b2*P[k-1] + b1*P[k-2];
       endif;
       if (S[l] > 1) then
         Q[k+l-1] = b1*P[k] + b2*P[k-1];
         Q[k+l] = P[k];
       else
         B=(U[k+1]-U[k])/(3*(U[k+2]-U[k]));
         Q[k+l-1] = B*P[k] + (1-B)*P[k-1];
         Q[k+l] = (B+b2)*P[k]+(b1-B)*P[k-1];
       endif;
       k = k + S[l];
   endfor;
   Q[2*n-2] = P[n-1]);
end.
```

CLS algorithm has more steps, but is able to handle an unclamped knot vector. Let us compare the algorithm designed regarding the information provided in Section 0 against CLS algorithm when $p = 2$ and $\hat{p} = 3$. Consider that knot multiplicity vector $\mathbf{S}$ was composed during the knot elevation (see Section 4.3).

Both algorithms have a similar number of steps and follow the special case scenario. In addition, CLS algorithm handles unclamped knot vectors. Let us examine the degree elevation of quartic and lower degree NURBS.

```
Procedure CLSQuadratic
// varable description coresponds to DDEQuadratic
begin
   k = -1; p = -1;
   for l = 0 to ns-2 do begin
      k = k + S[l];
      if (S[l] == 3) then
         p = p + 1; Q[p] = P[k-2];
      endif;
      if (S[l] >= 2) then
         p = p + 1;
         Q[p] = (P[k-2] + 2*P[k-1])/3;
      endif;
      h1 = U[k+1] - U[k];
      h2 = U[k+2] - U[k+1];
      comb = (h1*P[k] + h2*P[k-1])/(h1+h2);
      p = p + 1; Q[p] = (2*P[k-1] + comb)/3;
      p = p + 1;
      if (S[l+1] == 1) then
         Q[p] = (2*P[k] + comb)/3;
      else
         Q[p] = P[k];
       endif;
   endfor;
end.
```

```
Procedure NRatio
// in: U[m] – knot vector
// in: V[r] – elevated knot vector
// in: p – degree
// in: sl – knot multiplicity
// in: k1 – current knot
// in: k2 – current elevated knot
// in: type – ratio type
// out: NR[p+1] – basis function ratios
begin
   rat = 1;
   for i = 0 to p do begin NR[i] = rat;
   endfor;
   if (p >= 3) then
      for i = sl+1 to p-1
         if (type == 0) then
            rat = (V[k2+1] – V[k2-i-1]) /
            (U[k1+1] - U[k1-i]);
         else
            rat = (V[k2+i+2] - V[k2]) /
            (U[k1+i+1] - U[k1]);
         endif;
         NR[i+1] = NR[i]*rat;
   endif;
end.
```

As one might notice, in order to calculate (74) and (76) we must obtain the ratio of a certain basis functions first. Luckily, those functions are either the first or the last in non-zero basis function set at $u = u_k$. This means that the ratio is easy to calculate. In $p \le 4$ case, this ratio can be reduced to a single knot interval in numerator and another one in denominator. However, it is necessary to obtain all ratios

with intermediate values of the degree. Therefore, we suggest the method *NRatio* to calculate them. We provide the method *SolveEq* for the calculation of (75) and (76) as well. Setting variable *type* to 1 tells procedures to calculate (74) and (75). Any other value of *type* is used to calculate (76) and (77).

```
Procedure SolveEq
// in: V[r] – elevated knot vector
// in: k2 – current elevated knot
// in: p – degree
// in: sl – knot multiplicity
// in: NR – basis function ratio
// in: type – equation type
// out: Eq – short equation solution
begin
   sum = 1; q = p + 1;
   if (type == 1) then
      if (sl < q) then
         for i = k2+q-sl to k2+q do
         begin
            sum = sum - (V[i] – V[k2+sl]) /
            (q*(V[i] - V[k2]));
         endfor;
      else
         for i = k2-q+1 to k2-q+sl+1 do
         begin
            sum = sum - (V[k2] – V[i]) /
            (q*(V[k2+1] – V[i]));
         endfor;
   endif;
   Eq = NR*sum;
end.
```

As the special case scenario offers better performance, we have composed the direct degree elevation algorithm for the quartic and lower degree NURBS (*DDE*) using conditional segmentation.

```
Procedure DDE
// in: p – degree, p <= 4
// in: U[m] – knot vector
// in: V[r] – elevated knot vector
// in: P[n] – control points
// in: S[ns] – knot multiplicity vector
// local: N0[p+1] – basis function ratios from (74)
// local: N1[p+1] – basis function ratios from (76)
// out: Q[n2] – elevated control points
begin
   q = p+1; n2 = n+ns–1; k1 = -1; k2 = -1; Bnum = 1; Bden = 1;
   for l = 1 to ns-1 do begin
       k1 = k1+S[l-1]; k2 = k2+S[l-1] + 1; lb = k2+S[l]-p;
       N0S = 0; N1 = NRatio(U, V, p, S[l], k1, k2, 1);          // N1 denotes 3rd ratio in (74)
       if (ns-l > 1) then jE = lb+S[l+1]+1;
       else jE = n2; endif;
       if (jE < k2) then
          N0S = (V[k2+5]-V[k2+2])/(U[k1+3]-U[k1-1]);            // p=4 case, 3rd ratio in (76)
       endif;
       if (p-S[l] > 1) then
          N0 = NRatio(U, V, p, S[l], k1+S[l], k2+S[l]+1, 0);
          Bnum = V[k2+S[l]+2] - V[k2+S[l]+1];
       endif;
       for j = k2-S[l-1] to k2 do begin                         // for every non-zero knot interval
          G = {0, 1, 0}; i = j-l+1;                             // gammas and the last i
          if (j >= lb) then Bden = V[j+q+1] - V[j]; endif;      // 2nd ratio den. in (74) and (76)
          if (j == k2) then
          G[2] = SolveEq(V, k2, p, S[l]+1, N1[p], 1); i = i-1;  //Eq. (75)
          if (N0S > 0) then
                G[0] = (V[k2+5]-V[k2+4])*N0S/(5*Bden);          //Eq. (76)
          else
             G[2] = (k2-j)/q;                                   // 1st ratio from (74)
          if (j >= lb) then
             G[2] = G[2] * (V[k2+1]-V[k2])/Bden;                // 2nd ratio from (74)
                if (j > lb && j < k2) then                      // condition (83)
                   G[2] = G[2] * N1[q+j-k2];                    // 3rd ratio from (74)
                   if (j == jE) then
                      G[0] = SolveEq(V, k2+4, p, 2, N0S, 0);    //Eq. (75)
                   else
                      G[0] = (j-lb)/q*Bnum/Bden*N0[k2+S[l]-j+1]; //Eq. (76)
                   endif;
                endif;
             endif;
          G[1] = G[1] - G[2] + G[0];                            //restriction (17)
          if (G[1] == 0) Q[j] = P[i];
          else if (G[0] == 0) then Q[j] = G[2]*P[i] + G[1]*P[i-1]; endif;
          else Q[j] = G[2]*P[i] + G[1]*P[i-1] + G[0]*P[i-2]); endif;
       endfor;
   endfor;
end.
```

The next subsection discusses the algorithm designed for the uniform case of NURBS.

4.5. Degree Elevation of Uniform NURBS

The property of the uniformity is embedded into a knot vector [4]. NURBS is considered uniform whenever all knot intervals are equal, with the exception of the clamped ends. As the modification of knot intervals produces non-intuitive result, the great majority of CAD applications do not permit such adjustment. So, the knot vector stays uniform. Equal knot intervals suggest that all ratios in (74) and (76) become fractions that depend on two things: the degree p and the number of control points n. Hence, all elevation coefficient values can be pre-calculated and stored in the arrays. There are several facts to be considered before the inspection of the code.

- There is a transitional range between the clamped and unclamped knot intervals. Gammas that are not affected by the clamping can be stored in matrix $\mathbf{\Gamma}^{un}$ of size $n_{\gamma\max} \times 2$. The first dimension is obtained from (82). The second dimension equals the multiplicity of elevated knot, so in the uniform case it is $s_l + 1 = 2$, $l = 1..\eta - 1$.
- Transitional gammas $\mathbf{\Gamma}^{tr}$ are applicable for n_{tr} of control points at the ends of a control polygon: $n_{tr} = 2(p-1)$. (84)
- Calculations may be performed in a mirror-like fashion (with respect to a middle control point).
- The degree elevation of uniform spline produces a non-uniform spline ($s_l \neq 1$).

Algorithm *DDEU* is based on the arithmetic approach and works for arbitrary degree if appropriate gamma values are known. Gamma values are given in Table 7 for all $p \le 4$.

```
Preocedure DDEU
// in: p – degree, p <= 4
// in: P[n] – control points
// local: Gtr[ng, ntr] – transitional gammas
// local: Gun[ng, 2] – unclamped gammas
// out: Q[n2] – elevated control points
begin
   ng = ceil(p+2)/2; ntr = 2*(p-1);                          // Eq. (82) and (84)
   switch (p) ... endswitch;                                 // Load gamma values
   n2 = 2*n-p; n2h = n2/2; odd = Mod(p, 2);                  // Mod – remainder of div. by 2
   i = 0; k = p+1; iii = 0; n = n - 1; n2 = n2 – 1; ng = ng - 1;   // To avoid n-1 operation
   Q[0] = P[0]; Q[n2] = P[n];
   for j = 1 to n2h+odd-1 do begin
       Q[j] = {0, 0, 0, 0}; Q[n2-j] = {0, 0, 0, 0};
       if (j < k) then i = i + 1; endif;
       if (j < ntr) then                                     // Transitional range
          for ii = 0 to ng do begin
             if (Gtr[ii, j] > 0) then
                iii = i+ii-ng; Q[j] = Q[j] + Gtr[ii, j] * P[iii];
                if (j != n2h) then Q[n2-j] += Gtr[ii, j] * P[n-iii]; endif;
             endif;
          endfor;
       else                                                  // Unclamped range
          for ii = 0 to ng do begin
             if (Gun[ii, 1-k+j] > 0) then
                iii = i+ii-ng; Q[j] = Q[j] + Gun[ii, 1-k+j] * P[iii];
                if (j != n2h) then Q[n2-j] = Q[n2-j] + Gun[ii, 1-k+j] * P[n-iii]; endif;
             endif;
          endfor;
       endif;
       if (j == k) then k = k + 2; endif;
   endfor;
end.
```

Table 7. Pre-calculated gamma values for the uniform NURBS

p	i	*unclamped*		*transitional*					
		$\left[\mathbf{\Gamma}^{un}_{i,0}\right]$	$\left[\mathbf{\Gamma}^{un}_{i,1}\right]$	$\left[\mathbf{\Gamma}^{tr}_{i,0}\right]$	$\left[\mathbf{\Gamma}^{tr}_{i,1}\right]$	$\left[\mathbf{\Gamma}^{tr}_{i,2}\right]$	$\left[\mathbf{\Gamma}^{tr}_{i,3}\right]$	$\left[\mathbf{\Gamma}^{tr}_{i,4}\right]$	$\left[\mathbf{\Gamma}^{tr}_{i,5}\right]$
1	0	1/2	0						
	1	1/2	1						
2	0	5/6	1/6	0	1/3				
	1	1/6	5/6	1	2/3				
3	0	1/12	0	0	0	0	1/8		
	1	5/6	1/2	0	1/4	3/4	19/24		
	2	1/12	1/2	1	3/4	1/4	1/12		
4	0	1/3	1/30	0	0	0	1/10	4/9	2/45
	1	19/30	19/30	0	1/5	7/10	23/30	47/90	28/45
	2	1/30	1/3	1	4/5	3/10	2/15	1/30	1/3

The results of performance tests are illustrated and discussed in Section 5.

5. Results and Discussion

The algorithms presented in Section 4 were tested on Intel Core2 Duo 1.86 GHz x2 CPU, 3 GB RAM machine. Each algorithm was performed 10^4 times and then calculation time was divided by 10^4. This procedure was repeated several times to obtain an average calculation time value. The visual example of the degree elevated quartic NURBS is given in Figure 3. The original control polygon is depicted as faint blue line and the degree elevated polygon is marked as opaque blue line.

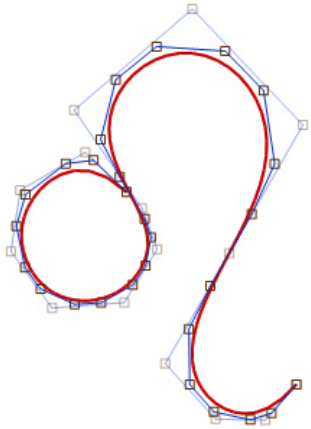


Figure 3. Degree elevated quartic NURBS

To emphasize the inefficiency of the schemes that involve redundant operations with control points, all control points were converted to homogeneous 4D space by using (6). In addition, we implemented CLS algorithms given in Subsection 4.4 and Piegl & Tiller's algorithm provided in [7]. CLS degree elevation version, optimized for linear and the quadratic cases, were tested against ours by measuring the control polygon elevation times. The results are presented in Figure 4.

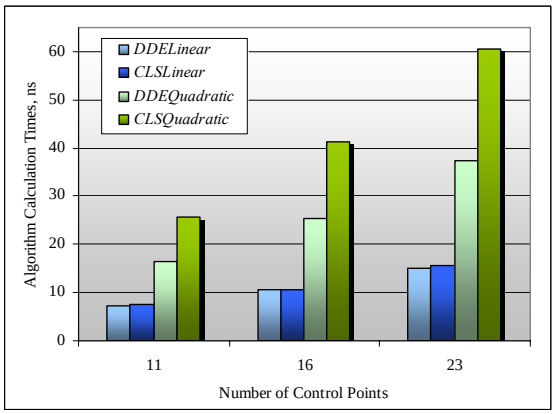


Figure 4. Control polygon elevation times

CLS algorithm designed for the linear case, clearly, has an upper hand. Its ability to handle the unclamped knot vectors comes at very small cost of calculation time. *DDELinear* managed to save only 1% - 3% of calculation time. The situation is very different in the case of the quadratic spline. *DDEQuadratic* finished the task from 36% to 39% faster than *CLSQuadratic*. This happens because of redundant operations with control points. E.g. one should use $(a/c)\mathbf{P}_i + (b/c)\mathbf{P}_{i+1}$ instead of $(a\mathbf{P}_i + b\mathbf{P}_{i+1})/c$ to avoid the third operation. In 4D case such a rearrangement saves up to 40% of calculation time.

Let us compare the entire degree elevation process of *DDE*, *DDEU* and Piegl & Tiller algorithms. The process includes the elevation of the knot vector (15), the conversion of the control polygon to homogeneous coordinates (6), the elevation of the control polygon, and its conversion back to the model space (7). The following tests were performed with a different number of control points and all the degree-specific cases, discussed in this paper. To reduce the amount of data, we divided the calculation time by the number of control points for every degree-specific case. Average calculation times per control point are given in Figure 5.

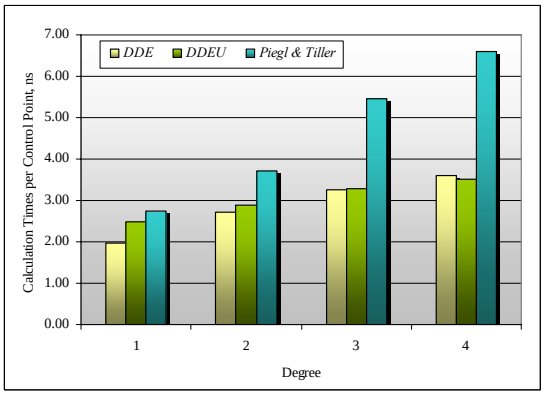


Figure 5. NURBS degree elevation times per control point

The direct degree elevation overtakes Piegl & Tiller's algorithm in all the cases of $p = 1...4$. From the first glance, the fact that the algorithm designed for the uniform case (*DDEU*) is slower than the generic case algorithm (*DDE*), may look odd. Actually, this example illustrates the advantage of the special case scenario used in *DDE* against the arithmetic approach used in *DDEU*.

Let us talk in the terms of calculation time economy. Figure 6 shows how much time *DDE* and *DDEU* algorithms were able to save in comparison to Piegl & Tiller's algorithm. Clearly, the degree of a spline has much greater effect to economy values than the number of control points. Another thing we can learn from this chart is that *DDEU* becomes competitive to *DDE* when $p \geq 3$. In general, the direct degree elevation algorithms we presented in this paper have demonstrated considerably better performance than Piegl & Tiller's algorithm. Calculation time economy varies from 8% to 49%.

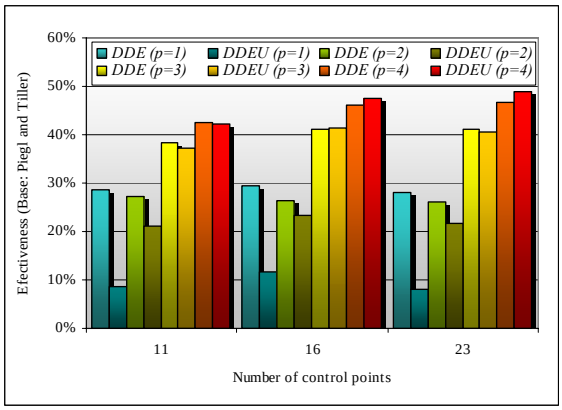


Figure 6. NURBS degree elevation time economy

Although, we did not test Huang, Hu & Martin's algorithm provided in [3], we can compare the efficiency against *DDE* by the means of calculation time economy. According to the data presented in [3], degree elevation of the cubic spline is performed 43% faster than Piegl & Tiller's elevation. Our algorithm reaches 41% economy under the same circumstances. Huang, Hu & Martin's approach is one of the most time-efficient algorithms up-to-date. So, we can claim that information uncovered in this paper enables the composition of highly competitive degree-specific elevation algorithms.

Theoretically, *DDE* algorithm is very limited because of $p \le 4$ restriction, but practically, there are two good reasons to limit any CAD system to lower degree splines: 1) lower degree splines are processed much faster and 2) human eye cannot detect discontinuities of third and higher derivatives [7].

The future work of this research would include the composition of algorithms applicable for an arbitrary degree, handling of the unclamped knot vector, and the introduction of the ability to elevate the degree of NURBS by arbitrary value at once.

6. Conclusions

In this paper we have discussed the degree elevation of a NURBS curve. By solving the linear equation system obtained from (13), we have discovered how original and elevated knot vectors relate to the control polygon of the degree elevated spline (through the elevation coefficients in equations (74) – (77)). Due to this discovery, we have composed the direct degree elevation scheme and several fairly simple and very fast algorithms.

By comparing the performance of our algorithms against CLS methods optimized for linear and quadratic cases we show that minimization of arithmetic operations with control points is very important. In the quadratic case our algorithm demonstrated 36% - 39% calculation time economy.

The performance tests of our direct degree elevation algorithms against Piegl & Tiller's degree elevation method showed that authors made a wrong assumption about the inefficiency of the linear equation solving approach (see [7]). Two of our algorithms have been tested: *DDE* that is based on the special case scenario and designed for quartic or lower degree splines, and *DDEU* that is based on the arithmetic approach and restricted to the uniform case of NURBS. The first one demonstrated 26% - 47% better performance than Piegl & Tiller's method, while the second one reached 8% - 49% of calculation time economy.

With all the facts in mind, we claim that in order to compose time-efficient and simple NURBS curve degree elevation algorithm, one must follow these steps: 1) define a direct relation between knot vectors and elevation coefficients, 2) apply a degree limitation if possible, and 3) use a special case scenario.

References

[1] **E. Cohen, T. Lyche, L.L. Schumaker.** Algorithms for Degree-Raising of Splines. *ACM Transactions on Graphics*, *Vol*.4 (3), 1985, 171–181.

[2] **G. Farin.** Curves and Surfaces for CAGD: A Practical Guide. *Morgan Kaufmann Publishers*, 2002

[3] **Q.X. Huang, S.M. Hu, R.R. Martin.** Fast degree elevation and knot insertion for B-spline curves. *Computer Aided Geometric Design*, *Vol*.22, 2005, 183–197.

[4] **K. Jankauskas.** Time-Efficient NURBS Curve Evaluation Algorithms. *Proceedings of the 16th International Conference on Information and Software Technologies*, 2010, 60–69.

[5] **B.G. Lee, Y. Park.** Degree Elevation of NURBS Curves by Weighted Blossom. *Journal of Applied Mathematics and Computing*, *Vol*.9 (1), 2002, 151–165.

[6] **L. Piegl, W. Tiller.** Software-Engineering Approach to Degree Elevation of B-Spline Curves. *Computer-Aided Design*, *Vol*.26 (1), 1994, 17–28.

[7] **L. Piegl, W. Tiller.** The NURBS Book. *Springer*, 1999.

[8] **H. Prautzsch, B. Piper.** A fast algorithm to raise the degree of spline curves. *Computer Aided Geometric Design*, *Vol*.8 (4), 1991, 253–265.

[9] **K. Qin.** A Matrix Method for Degree-Raising of B-Spline Curves. *Science in China (E)*, *Vol*.40 (1), 1997, 71–81

[10] **G. Wang, C. Deng.** On the Degree Elevation of B-Spline Curves and Corner Cutting. *Computer Aided Geometric Design*, *Vol*.24 (2), 2007, 90–98.

Received August 2010.

DOI: 10.5755/j01.itc.39.4.12382