Technical Report DIKU-TR-02/17
Department of Computer Science
University of Copenhagen
Universitetsparken 1
DK-2100 KBH Ø
DENMARK

August 2002

B-splines

Kenny Erleben and Knud Henriksen

Abstract: In this paper we will work our way through classical B-spline theory with focus on efficient implementation. Afterwards we will look into a few advanced details, such as curve decomposition and regular nonuniform B-splines.

Contents

1 Introduction **2**

2 The B-spline Basis Functions **2**
2.1 Definition and Properties of B-spline Basis Functions . . . 2
2.2 Implementing The Basis Functions . . . 3
2.3 The Derivatives of The Basis Functions . . . 5
2.4 Implementing The Derivatives of The Basis Functions . . . 9

3 The B-spline **13**
3.1 Definition of B-spline . . . 14
3.2 Implementing The B-spline . . . 14

4 Global Interpolation **15**

5 Cubic Curve Decomposition **17**
5.1 The Cubic Bezier Curve and the B-Spline Connection . . . 17
5.2 Global and Local Parameter Conversion of Derivatives . . . 19
5.3 Knot Insertion . . . 20
5.3.1 Divided Difference . . . 20
5.3.2 Leibniz' Formula . . . 20
5.3.3 The Funny Equation . . . 21
5.3.4 The Ordinary B-spline Basis Function . . . 21
5.3.5 The Knot Insertion Algorithm . . . 23

6 Accumulated Arc Length Table **26**

7 The Regular Cubic Nonuniform B-spline **26**
7.1 First Order Derivatives at Knot Values . . . 27
7.2 First Order Derivatives between Knot Values . . . 29
7.2.1 Two Linear Dependent Δ's . . . 30
7.2.2 Three Linear Dependent Δ's . . . 30
7.2.3 Conclusion on Linear Dependency . . . 31

8 Conclusion **31**

A The de Boor Algorithm **31**

B Repeated Knot Insertion **32**

1 Introduction

In this paper we will present the classical theory of nonuniform B-splines and while we do it we will derive a c-style pseudocode for efficient implementation of the theory.

The theory represented in this paper is spawn from another project we worked on about "Scripted Motion and Spline Driven Animation" [8]. Due to this the topics we present in this paper are orientated towards regular nonuniform open B-splines and efficient implementation of the spline theory.

The theory of B-splines is a much wider area than the treatment we give in this paper, we therefore encourage the reader to look at our references if a wider coverage is wanted.

We believe this paper is perfect for those, just wanting to implement and use open nonuniform B-splines or for those in need of a regular nonuniform open B-spline.

2 The B-spline Basis Functions

Before we actually look at the B-splines we will treat the B-spline basis functions.

2.1 Definition and Properties of B-spline Basis Functions

Let us start by defining the normalized basis function for the B-spline. The normalized basis functions are defined recursively by the Cox de Boor definition. For $k > 1$:

$$N_{i,k} = \frac{u - t_i}{t_{i+k-1} - t_i} N_{i,k-1} + \frac{t_{i+k} - u}{t_{i+k} - t_{i+1}} N_{i+1,k-1} \tag{1}$$

and for $k = 1$:

$$N_{i,1} = \begin{cases} 1 & \text{if } t_i \leq u < t_{i+1} \\ 0 & \text{otherwise} \end{cases}$$

The index k is called the order of the basis function. For a given value of k this definition results in a polynomium of degree $k - 1$. All the t_i's are called knot values, and are usually arranged in a so called knot vector, T.

$$T = [\ldots, t_{i-1}, t_i, t_{i+1}, t_{i+2}, \ldots]^T$$

Now let us assume we have a value u such that

$$t_i \leq u < t_{i+1}$$

That is the value of the index i is known. Looking at the definition of the B-spline basis functions we can esily derive the "dependency" table, shown in Table 1. The table should be read in a bottom-up fashion,

$k = 3$		$N_{i-2,3}$		$N_{i-1,3}$		$N_{i,3}$	
$k = 2$			$N_{i-1,2}$		$N_{i,2}$		
$k = 1$				$N_{i,1}$			
	t_{i-1}		t_i	u	t_{i+1}		t_{i+2}

Table 1: Depedency Table.

starting at the bottom with the knot vector, one can track all non-zero basis functions up to any wanted k-value. The patteren is obvious and we could easily have extended the table to any value of k. There are three properties, which can be seen immediately from the table and which we are going to put to practical use later on.

1. At the k'th order there are exactly k basis functions, which are different from zero.

2. From the table we can derive that if
$$t_i \le u < t_{i+1}$$
then only
$$N_{i-k+1,k}, \ldots, N_{i,k}$$
will be nonzero.

3. Computing the basis functions bottom-up, instead of top-down, will make it possible to reuse previously computed results.

We will use properties 1 and 2 to make a sort of fast rejection on which basis functions we need to compute. The third property is obvious from the table, but it requires some work before it can be applied efficiently in a practical implementation.

2.2 Implementing The Basis Functions

Let us look at an example. Let us assume that we have
$$t_i \le u < t_{i+1}$$
Then by the the definition of the basis functions and the properties 1 and 2 we can write up all the nonzero basis functions for $k = 1$
$$N_{i,1} = 1$$
and for $k = 2$
$$\begin{aligned} N_{i-1,2} &= \frac{u - t_{i-1}}{t_{i+0} - t_{i-1}} N_{i-1,1} + \frac{t_{i+1} - u}{t_{i+1} - t_i} N_{i,1} = \frac{t_{i+1} - u}{t_{i+1} - t_i} N_{i,1} \\ N_{i,2} &= \frac{u - t_i}{t_{i+1} - t_i} N_{i,1} + \frac{t_{i+2} - u}{t_{i+2} - t_{i+1}} N_{i+1,1} = \frac{u - t_i}{t_{i+1} - t_i} N_{i,1} \end{aligned}$$
and for $k = 3$
$$\begin{aligned} N_{i-2,3} &= \frac{u - t_{i-2}}{t_{i+0} - t_{i-2}} N_{i-2,2} + \frac{t_{i+1} - u}{t_{i+1} - t_{i-1}} N_{i-1,2} = \frac{t_{i+1} - u}{t_{i+1} - t_{i-1}} N_{i-1,2} \\ N_{i-1,3} &= \frac{u - t_{i-1}}{t_{i+1} - t_{i-1}} N_{i-1,2} + \frac{t_{i+2} - u}{t_{i+2} - t_i} N_{i,2} \\ N_{i,3} &= \frac{u - t_i}{t_{i+2} - t_i} N_{i,2} + \frac{t_{i+3} - u}{t_{i+3} - t_{i+1}} N_{i+1,2} = \frac{u - t_i}{t_{i+2} - t_i} N_{i,2} \end{aligned}$$
We can immediately make some observations if we for instance look at $N_{i-1,3}$ then we observe that the part
$$\frac{\ldots}{t_{i+2} - t_i} N_{i,2}$$
of the second term reapperas in the first term of $N_{i,3}$. Another observation we can make is that the first part of $N_{i-2,3}$ is allways zero and the same goes for the last part of $N_{i,3}$. This is seen directly from our dependency table in Table 1. These two observations can we use to speed up our computations, but we can do even better. Looking at the fractions of knot differences we see that they appear to be similar for increasing values of k. Let us try to see if we can put some system into these fractions. We will start by introducing two new auxilliary functions called $left$ and $right$. These are defined as follows
$$\begin{aligned} left(j) &= u - t_{i+1-j} \\ right(j) &= t_{i+j} - u \end{aligned}$$

With these new methods we can write up the equations in our example like we have done below (for the case where $k = 3$).

$$\begin{aligned}
N_{i-2,3} &= \frac{left(3)}{right(0)+left(3)}N_{i-2,2} + \frac{right(1)}{right(1)+left(2)}N_{i-1,2} \\
N_{i-1,3} &= \frac{left(2)}{right(1)-left(2)}N_{i-1,2} + \frac{right(2)}{right(2)+left(1)}N_{i,2} \\
N_{i,3} &= \frac{left(1)}{right(2)+left(1)}N_{i,2} + \frac{right(3)}{right(3)+left(0)}N_{i+1,2}
\end{aligned}$$

From this we can derive the following general relation

$$\begin{aligned}
M(r,k) &= N_{i-(k-r-1),k} \\
&= \frac{left(k-r)}{right(r)-left(k-r)}N_{i-1,k-1} \\
&+ \frac{right(r+1)}{right(r+1)+left(k-r-1)}N_{i,k-1}
\end{aligned}$$

Where

$$0 \leq r < k$$

If we closely examine our equations from earlier we notice that when we begin a computation on a new k'th level then the only left and right values we do not allready have computed are

$$left(k) \qquad \text{and} \qquad right(k)$$

We are now ready to write up the pseudo code, which computes the values of all nonzero $N_{i,k}$

```
Algorithm BasisFunc(i,u,K,T)
  left = array(K+1)
  right = array(K+1)
  M = array(K)
  left[0]  = u - T[i+1]
  right[0] = T[i] - u

  left[1]  = u - T[i]
  right[1] = T[i+1] - u
  M[0] = 1

  For k = 2 to K do
    left[k] = u - T[i+1-k]
    right[k] = T[i+k] - u

    saved = 0

    For r=0 to k-2
      tmp = M[r]/(right[r+1]+left[k-r-1])
      M[r] = saved + right[r+1]*tmp
      saved = left[k-r-1]*tmp;
    Next r

    M[k-1] = saved

  Next k
  return M
End algorithm
```

2.3 The Derivatives of The Basis Functions

We will start out by showing the result of computing the first order derivative of a basis function

$$\frac{dN_{i,k}}{du} = \frac{k-1}{t_{i+k-1}-t_i}N_{i,k-1} - \frac{k-1}{t_{i+k}-t_{i+1}}N_{i+1,k-1} \tag{2}$$

Now let us try to prove that this equation is infact true. We will prove the equation by induction on k. Setting $k = 2$ we see by direct differentiation of equation (1) that the derivative would be

$$\frac{1}{t_{i+k-1}-t_i} \quad \text{or} \quad \frac{1}{t_{i+k}-t_{i+1}} \quad \text{or} \quad 0$$

depending on what interval u lies within. By direct substitution into equation (2) we can easily verify that equation (2) is true in the case of $k = 2$. Now let us assume that equation (2) is true for $k-1$ and that we have $k > 2$. Now we are ready to try to prove that equation (2) is true for the k'th case. We start by applying the product rule to equation (1) and get the following equation

$$\begin{aligned}\frac{dN_{i,k}}{du} &= \frac{1}{t_{i+k-1}-t_i}N_{i,k-1} - \frac{1}{t_{i+k}-t_{i+1}}N_{i+1,k-1} \\ &+ \frac{u-t_i}{t_{i+k-1}-t_i}\frac{dN_{i,k-1}}{du} + \frac{t_{i+k}-u}{t_{i+k}-t_{i+1}}\frac{dN_{i+1,k-1}}{du}\end{aligned} \tag{3}$$

Now we will substitute equation (2) for the terms $N'_{i,k-1}$ and $N'_{i+1,k-1}$. This gives the following equation

$$\begin{aligned}
\frac{dN_{i,k}}{du} &= \frac{1}{t_{i+k-1}-t_i}N_{i,k-1} - \frac{1}{t_{i+k}-t_{i+1}}N_{i+1,k-1} \\
&+ \frac{u-t_i}{t_{i+k-1}-t_i}\left(\frac{k-2}{t_{i+k-2}-t_i}N_{i,k-2} - \frac{k-2}{t_{i+k-1}-t_{i+1}}N_{i+1,k-2}\right) \\
&+ \frac{t_{i+k}-u}{t_{i+k}-t_{i+1}}\left(\frac{k-2}{t_{i+k-1}-t_{i+1}}N_{i+1,k-2} - \frac{k-2}{t_{i+k}-t_{i+2}}N_{i+2,k-2}\right)
\end{aligned}$$

Now this looks rather complicated, so let us try to make it a little more readable by rearranging the equation.

$$\begin{aligned}
\frac{dN_{i,k}}{du} &= \frac{1}{t_{i+k-1}-t_i}N_{i,k-1} - \frac{1}{t_{i+k}-t_{i+1}}N_{i+1,k-1} \\
&+ \frac{k-2}{t_{i+k-1}-t_i}\frac{u-t_i}{t_{i+k-2}-t_i}N_{i,k-2} \\
&+ \frac{k-2}{t_{i+k-1}-t_{i+1}}\left(\frac{t_{i+k}-u}{t_{i+k}-t_{i+1}} - \frac{u-t_i}{t_{i+k-1}-t_i}\right)N_{i+1,k-2} \\
&- \frac{k-2}{t_{i+k}-t_{i+1}}\frac{t_{i+k}-u}{t_{i+k}-t_{i+2}}N_{i+2,k-2}
\end{aligned}$$

Now we do a little mathematical manipulation. We add zero to the equation, by adding 1 and subtracting 1.

$$\begin{aligned}
\frac{dN_{i,k}}{du} &= \frac{1}{t_{i+k-1}-t_i}N_{i,k-1} - \frac{1}{t_{i+k}-t_{i+1}}N_{i+1,k-1} \\
&+ \frac{k-2}{t_{i+k-1}-t_i}\frac{u-t_i}{t_{i+k-2}-t_i}N_{i,k-2} \\
&+ \frac{k-2}{t_{i+k-1}-t_{i+1}}\left(\frac{t_{i+k}-u}{t_{i+k}-t_{i+1}} - 1 + 1 - \frac{u-t_i}{t_{i+k-1}-t_i}\right)N_{i+1,k-2} \\
&- \frac{k-2}{t_{i+k}-t_{i+1}}\frac{t_{i+k}-u}{t_{i+k}-t_{i+2}}N_{i+2,k-2}
\end{aligned}$$

Now we express the ones we have added and subtracted as fractions

$$\begin{aligned}
\frac{dN_{i,k}}{du} &= \frac{1}{t_{i+k-1}-t_i}N_{i,k-1} - \frac{1}{t_{i+k}-t_{i+1}}N_{i+1,k-1} \\
&+ \frac{k-2}{t_{i+k-1}-t_i}\frac{u-t_i}{t_{i+k-2}-t_i}N_{i,k-2} \\
&+ \frac{k-2}{t_{i+k-1}-t_{i+1}}\left(\frac{t_{i+k}-u}{t_{i+k}-t_{i+1}} - \frac{t_{i+k}-t_{i+1}}{t_{i+k}-t_{i+1}}\right. \\
&\left. + \frac{t_{i+k-1}-t_i}{t_{i+k-1}-t_i} - \frac{u-t_i}{t_{i+k-1}-t_i}\right)N_{i+1,k-2} \\
&- \frac{k-2}{t_{i+k}-t_{i+1}}\frac{t_{i+k}-u}{t_{i+k}-t_{i+2}}N_{i+2,k-2}
\end{aligned}$$

Setting on common denominators we get

$$\begin{aligned}
\frac{dN_{i,k}}{du} &= \frac{1}{t_{i+k-1}-t_i}N_{i,k-1} - \frac{1}{t_{i+k}-t_{i+1}}N_{i+1,k-1} \\
&+ \frac{k-2}{t_{i+k-1}-t_i}\frac{u-t_i}{t_{i+k-2}-t_i}N_{i,k-2} \\
&+ \frac{k-2}{t_{i+k-1}-t_{i+1}}\left(\frac{t_{i+k}-u-t_{i+k}+t_{i+1}}{t_{i+k}-t_{i+1}}\right. \\
&\quad \left.+\frac{t_{i+k-1}-t_i-u+t_i}{t_{i+k-1}-t_i}\right)N_{i+1,k-2} \\
&- \frac{k-2}{t_{i+k}-t_{i+1}}\frac{t_{i+k}-u}{t_{i+k}-t_{i+2}}N_{i+2,k-2}
\end{aligned}$$

Cleaning up a bit we get the following

$$\begin{aligned}
\frac{dN_{i,k}}{du} &= \frac{1}{t_{i+k-1}-t_i}N_{i,k-1} - \frac{1}{t_{i+k}-t_{i+1}}N_{i+1,k-1} \\
&+ \frac{k-2}{t_{i+k-1}-t_i}\frac{u-t_i}{t_{i+k-2}-t_i}N_{i,k-2} \\
&+ \frac{k-2}{t_{i+k-1}-t_{i+1}}\left(\frac{t_{i+1}-u}{t_{i+k}-t_{i+1}}+\frac{t_{i+k-1}-u}{t_{i+k-1}-t_i}\right)N_{i+1,k-2} \\
&- \frac{k-2}{t_{i+k}-t_{i+1}}\frac{t_{i+k}-u}{t_{i+k}-t_{i+2}}N_{i+2,k-2}
\end{aligned}$$

Now we are almost at our goal. All that we have to do is to rearrange the equation a little bit more.

$$\begin{aligned}
\frac{dN_{i,k}}{du} &= \frac{1}{t_{i+k-1}-t_i}N_{i,k-1} - \frac{1}{t_{i+k}-t_{i+1}}N_{i+1,k-1} \\
&+ \frac{k-2}{t_{i+k-1}-t_i}\left(\frac{u-t_i}{t_{i+k-2}-t_i}N_{i,k-2}+\frac{t_{i+k-1}-u}{t_{i+k-1}-t_{i+1}}N_{i+1,k-2}\right) \\
&- \frac{k-2}{t_{i+k}-t_{i+1}}\left(\frac{u-t_{i+1}}{t_{i+k-1}-t_{i+1}}N_{i+1,k-2}+\frac{t_{i+k}-u}{t_{i+k}-t_{i+2}}N_{i+2,k-2}\right)
\end{aligned}$$

Now if we look at the terms in the parenteses we immediately recognize equation (1), and if we apply it we get the following

$$\begin{aligned}
\frac{dN_{i,k}}{du} &= \frac{1}{t_{i+k-1}-t_i}N_{i,k-1} - \frac{1}{t_{i+k}-t_{i+1}}N_{i+1,k-1} \\
&+ \frac{k-2}{t_{i+k-1}-t_i}N_{i,k-1} - \frac{k-2}{t_{i+k}-t_{i+1}}N_{i+1,k-1}
\end{aligned}$$

Cleaning up a bit more we get

$$\begin{aligned}
\frac{dN_{i,k}}{du} &= \frac{(k-2)+1}{t_{i+k-1}-t_i}N_{i,k-1} - \frac{(k-2)+1}{t_{i+k}-t_{i+1}}N_{i+1,k-1} \\
&= \frac{k-1}{t_{i+k-1}-t_i}N_{i,k-1} - \frac{k-1}{t_{i+k}-t_{i+1}}N_{i+1,k-1}
\end{aligned}$$

Finally we can conclude that equation (2) is true. Now let us try to look at higher order derivatives. If we differentiate equation (2) then we get

$$\frac{d^2N_{i,k}}{du^2} = \frac{k-1}{t_{i+k-1}-t_i}\frac{dN_{i,k-1}}{du} - \frac{k-1}{t_{i+k}-t_{i+1}}\frac{dN_{i+1,k-1}}{du}$$

From this it is not hard to see that a general formula can be written for the j'th derivative.

$$\frac{d^j N_{i,k}}{du^j} = \frac{k-1}{t_{i+k-1}-t_i}\frac{d^{j-1}N_{i,k-1}}{du^{j-1}} - \frac{k-1}{t_{i+k}-t_{i+1}}\frac{d^{j-1}N_{i+1,k-1}}{du^{j-1}} \tag{4}$$

Our results can be generalized into the following equation.

$$\frac{d^j N_{i,k}}{du^j} = \frac{(k-1)!}{(k-j-1)!}\sum_{p=0}^{j} a_{j,p}N_{i+p,k-j} \tag{5}$$

With

$$j < k$$

and

$$\begin{aligned} a_{0,0} &= 1 && (6)\\ a_{j,0} &= \frac{a_{j-1,0}}{t_{i+k-j}-t_i} && (7)\\ a_{j,p} &= \frac{a_{j-1,p}-a_{j-1,p-1}}{t_{i+k+p-j}-t_{i+p}} && (8)\\ a_{j,j} &= -\frac{a_{j-1,j-1}}{t_{i+k}-t_{i+j}} && (9) \end{aligned}$$

We will now try to prove equation (5) by induction on j. Let us start out by examining if the equation holds for $j = 1$. From direct substitution into equation (5) we get the following result.

$$\begin{aligned} \frac{dN_{i,k}}{du} &= \frac{(k-1)!}{(k-2)!}\left(a_{1,0}N_{i,k-1} + a_{1,1}N_{i+1,k-1}\right)\\ &= (k-1)\left(\frac{a_{0,0}}{t_{i+k-1}-t_i}N_{i,k-1} + \frac{-a_{0,0}}{t_{i+k}-t_{i+1}}N_{i+1,k-1}\right)\\ &= (k-1)\left(\frac{1}{t_{i+k-1}-t_i}N_{i,k-1} - \frac{1}{t_{i+k}-t_{i+1}}N_{i+1,k-1}\right)\\ &= \frac{k-1}{t_{i+k-1}-t_i}N_{i,k-1} - \frac{k-1}{t_{i+k}-t_{i+1}}N_{i+1,k-1} \end{aligned}$$

We immediately recoginize equation (2) and conclude that equation (5) is true for $j = 1$. Now, let us assume that equation (5) is true for the case of $j = j-1$ and let us try to prove that it also is true in the j'th case. We have

$$\frac{d^{j-1}N_{i,k}}{dt^{j-1}} = \frac{(k-1)!}{(k-(j-1)-1)!}\sum_{p=0}^{j-1} a_{j-1,p}N_{i+p,k-j+1}$$

If we differentiate this once more we get the j'th derivative, that is

$$\frac{d^j N_{i,k}}{dt^j} = \frac{(k-1)!}{(k-(j-1)-1)!}\sum_{p=0}^{j-1} a_{j-1,p}\frac{dN_{i+p,k-j+1}}{du}$$

Now let us apply equation (2) to the derivative inside the summation

$$\begin{aligned} \frac{d^j N_{i,k}}{dt^j} &= \frac{(k-1)!}{(k-j)!}\sum_{p=0}^{j-1} a_{j-1,p}\left(\frac{k-j}{t_{i+p+k-j}-t_{i+p}}N_{i+p,k-j}\right.\\ &\qquad \left. - \frac{k-j}{t_{i+p+k-j+1}-t_{i+p+1}}N_{i+p+1,k-j}\right) \end{aligned}$$

Rearranging yields

$$\begin{aligned}\frac{d^j N_{i,k}}{dt^j} &= \frac{(k-1)!}{(k-j-1)!}\sum_{p=0}^{j-1}\left(\frac{a_{j-1,p}}{t_{i+p+k-j}-t_{i+p}}N_{i+p,k-j}\right.\\ &\left.-\frac{a_{j-1,p}}{t_{i+p+k-j+1}-t_{i+p+1}}N_{i+p+1,k-j}\right)\end{aligned}$$

Let us write up the summation into explicit terms. This results in the following rather lengthy equation

$$\begin{aligned}\frac{d^j N_{i,k}}{dt^j} &= \frac{(k-1)!}{(k-j-1)!}\left(\frac{a_{j-1,0}}{t_{i+k-j}-t_i}N_{i,k-j}-\frac{a_{j-1,0}}{t_{i+k-j+1}-t_{i+1}}N_{i+1,k-j}\right.\\ &+\frac{a_{j-1,1}}{t_{i+1+k-j}-t_{i+1}}N_{i+1,k-j}-\frac{a_{j-1,1}}{t_{i+k-j+2}-t_{i+2}}N_{i+2,k-j}\\ &+\frac{a_{j-1,2}}{t_{i+2+k-j}-t_{i+2}}N_{i+2,k-j}-\frac{a_{j-1,2}}{t_{i+k-j+3}-t_{i+3}}N_{i+3,k-j}\\ &\cdots\\ &\left.+\frac{a_{j-1,j-1}}{t_{i+k-1}-t_{i+j-1}}N_{i+j-1,k-j}-\frac{a_{j-1,j-1}}{t_{i+k}-t_{i+j}}N_{i+j,k-j}\right)\end{aligned}$$

This equation can be rewritten by collecting terms of equal N-factors

$$\begin{aligned}\frac{d^j N_{i,k}}{dt^j} &= \frac{(k-1)!}{(k-j-1)!}\left(\frac{a_{j-1,0}}{t_{i+k-j}-t_i}N_{i,k-j}\right.\\ &+\frac{a_{j-1,1}-a_{j-1,0}}{t_{i+1+k-j}-t_{i+1}}N_{i+1,k-j}\\ &+\frac{a_{j-1,2}-a_{j-1,1}}{t_{i+2+k-j}-t_{i+2}}N_{i+2,k-j}\\ &\cdots\\ &+\frac{a_{j-1,j-1}-a_{j-1,j-2}}{t_{i+k-1}-t_{i+j-1}}N_{i+j-1,k-j}\\ &\left.-\frac{a_{j-1,j-1}}{t_{i+k}-t_{i+j}}N_{i+j,k-j}\right)\end{aligned}$$

Looking at how the a-coefficients in equations (6)-(9) are defined we can esily rewrite the equation into the following form.

$$\begin{aligned}\frac{d^j N_{i,k}}{dt^j} &= \frac{(k-1)!}{(k-j-1)!}(a_{j,0}N_{i,k-j}+a_{j,1}N_{i+1,k-j}+a_{j,2}N_{i+2,k-j}+\cdots\\ &\cdots+a_{j,j}N_{i+j,k-j})\end{aligned}$$

Which conviently can be written as

$$\frac{d^j N_{i,k}}{dt^j} = \frac{(k-1)!}{(k-j-1)!}\sum_{p=0}^{j} a_{j,p}N_{i+p,k-j}$$

By comparison with equation (5) we can now conclude that it also holds in j'th case.

2.4 Implementing The Derivatives of The Basis Functions

Now let us turn our attention towards making an efficient implementation. By equation (5) we have

$$\frac{d^j N_{i,k}}{dt^j} = \frac{(k-1)!}{(k-j-1)!}\sum_{p=0}^{j} a_{j,p}N_{i+p,k-j} \qquad j<k$$

And from equation (5) we see that the numerators in the equations (7)-(9) are

$$\Delta T = t_{i+k+p-j} - t_{i+p} \qquad \text{for} \qquad 0 \le p \le j \tag{10}$$

By equation (1) we have

$$N_{i+p-1,k-j+1} = \frac{u - t_{i+p-1}}{t_{i+k+p-j-1} - t_{i+p-1}} N_{i+p-1,k-j} + \frac{t_{i+k+p-j} - u}{t_{i+k+p-j} - t_{i+p}} N_{i+p,k-j}$$

Now if we compare the numerator of the second term to equation (10) we immediately see that they are identical. In other words we have seen that the knot differences used to compute the N-values can be reused in the computation of the derivatives. Let us create a table, which we call M, and let us store values into M as follows, for $0 \le r < k$:

$$\begin{aligned} N_{i-k+1+r,k} &\rightarrow M_{r,k-1} \\ \Delta T &\rightarrow M_{k-1,r} \end{aligned}$$

The diagonal is special since we will store $N_{i,k}$ in it. In Table 2 we have given an example of an M-table to help familiarize you with it. Observe that M contains all the non-zero values of N from the dependency

$N_{i,1}$	$N_{i-1,2}$	$N_{i-2,3}$	$N_{i-3,4}$
$t_{i+1} - t_i$	$N_{i,2}$	$N_{i-1,3}$	$N_{i-2,4}$
$t_{i+1} - t_{i-1}$	$t_{i+2} - t_i$	$N_{i,3}$	$N_{i-1,4}$
$t_{i+1} - t_{i-2}$	$t_{i+2} - t_{i-1}$	$t_{i+3} - t_i$	$N_{i,4}$

Table 2: M-table example for $k = 4$.

table and it also stores the corresponding knot differeneces (from the right terms only) in the "transpose" position of the corresponding N value.

The knot difference we need in the a-coefficient of $N_{i+p,k-j}$ is computed by $N_{i+p-1,k-j+1}$. Where are these two located relatively to each other in the M-table? $N_{i+p-1,k-j+1}$ would lie just to the right of $N_{i+p,k-j}$. Knowing this we can easily come up with the following rule for looking up the knot difference we need when we want to compute the a-coeffiecients. The rule is as follows

1. Get the transpose position of the corresponding N-value.
2. Add one to the row index.

If we compute the derivatives in succeding order starting at 0, then $1, 2, .., j-1, j$ then we observe that when we compute the j'th derivative then we only need the a-coefficent values which we found in the computation of the $j-1$'th derivatives (and the knot differences of course). This means that we only have to remember the a-coeffiencent from the previous iteration. Using a cyclic memory buffer makes it possible to implement a very effiecient way of computing the a-coefficients.

Looking closely at equation (5) we obviously see that some of the terms involved in the summation could potentially have a zero N-value. In an implementation we could take advantage of this knowledge. This could be done as follows. First we split the summation into three parts, each part corresponding to the three different ways of computing the a-coefficients (equations (7)-(9)). That is, we are going to use the following index values for the N-values.

$$\begin{array}{rcl} i-k+1+r & \text{for} & a_{j,0} \\ i-k+1+r+1 \quad \text{to} \quad i-k+1+r+j-1 & \text{for} & a_{j,1} \quad \text{to} \quad a_{j,j-1} \\ i-k+1+r+j & \text{for} & a_{j,j} \end{array}$$

Now, since all these N-values are of order $k-j$ we know that there are at most $k-j$ non-zero N-values and we also know that their index range is

$$i-k+j+1 \quad \text{to} \quad i$$

In other words, if

$$\begin{aligned} i-k+1+r &\geq i-k+j+1 \\ r &\geq j \end{aligned}$$

then the $a_{j,0}$ term have a non-zero N-value and should therefore be included in the summation. Similar we see that if

$$\begin{aligned} i-k+1+r+j &\leq i \\ r &\leq k-1-j \end{aligned}$$

Then the $a_{j,j}$ term should be included as well. The $a_{j,p}$ terms (for $p = 1$ to $j-1$) is handled a little different. In this case we want to derive the range of p-values, $p_{min}, \ldots, p_{max}$, which results of non-zero values. First we look for a lower limit if

$$\begin{aligned} i-k+1+r+1 &\geq i-k+j+1 \\ r+1 &\geq j \end{aligned}$$

then p_{min} should be set to 1. If not we have to solve for p_{min} such that we find the lowest value of p_{min} where

$$\begin{aligned} i-k+1+r+p_{min} &\geq i-k+j+1 \\ p_{min} &\geq j-r \\ p_{min} &= j-r \end{aligned}$$

In a similar way we find p_{max}. If

$$\begin{aligned} i-k+1+r+j-1 &\leq i \\ r+j &\leq k \\ r &\leq k-j \end{aligned}$$

then $p_{max} = j-1$, and if not then we need the bigest value of p_{max} such that

$$i-k+1+r+p_{max} \leq i$$

which is $p_{max} = k-1-r$.

This concludes our walkthrough of implementation details, and we can now show the entire algorithm.

```
Algorithm initializeM(i,u,K,T)
  left = array(K+1)
  right = array(K+1)
  M = array(K,K)
  left[0]  = u - T[i+1]
  right[0] = T[i] - u

  left[1]  = u - T[i]
  right[1] = T[i+1] - u
  M[0][0] = 1

  For k = 2 to K do
    left[k] = u - T[i+1-k]
    right[k] = T[i+k] - u

    saved = 0

    For r=0 to k-2
      M[k-1][r] = (right[r+1]+left[k-r-1])
      tmp = M[r][k-2]/M[k-1][r]

      M[r][k-1] = saved + right[r+1]*tmp
      saved = left[k-r-1]*tmp;
    Next r
    M[k-1][k-1] = saved
  Next k
  return M
End Algorithm
```

```
Algorithm derivatives(i,u,K,T,J)
  D =  array(J+1,K)  // D[j][r] = j'th derivative of $N_{i-k+1+r,k}$
  a = array(2,K)
  M = initializeM(i,u,K,T)

  For r=0 to K-1
    D[0][r] = M[r][K-1]
  Next r

  For r=0 to K-1
    s1 = 0, s2 = 1, a[0][0] = 1
    For j=1 to J
      djN = 0
      If r >= j then
        a[s2][0] = a[s1][0]/M[K-j][r-j]
        djN +=  a[s2][0] * M[r-j][K-j-1]
      End if
      If r + 1 >= j then pMin = 1
      Else pMin = j-r End if
      If r <= K -j  then pMax = j - 1
      Else pMax = K - 1 -r End if
      For p=pMin to pMax
        a[s2][p]=(a[s1][p]-a[s1][p-1])/M[K-j][r+p-j]
        djN+=a[s2][p]*M[r+p-j][K-j-1]
      Next p
      If r <= (K-1-j) then
        a[s2][j] = -a[s1][K-1]/M[K-j][r]
        djN +=  a[s2][j] * M[r][K-j-1]
      End if
      D[j][r] = djN
      swap(s1,s2)
    Next j
  Next r

  factor = K-1;
  For j=1 to J
    For r=0 to K-1
      D[j][r] = factor * D[j][r]
    Next r
    factor = factor * (K-1-j)
  next J
  return D
End Algorithm
```

3 The B-spline

Having taken care of the basis functions we are now ready for the actual B-spline.

3.1 Definition of B-spline

A non-periodic non-uniform k-order B-spline is defined by

$$C(u) = \sum_{i=0}^{n} N_{i,k} P_i \tag{11}$$

Where

$$a \leq u \leq b$$

The $n+1$ points $\{P_i | i = 0, \ldots, n\}$ are called the control points and the functions $\{N_{i,k} | i = 0, \ldots, n\}$ are the k'th order normalized basis functions, which were treated in the previous sections. These are defined on a knot vector T, such that

$$T = [\underbrace{a, \ldots, a}_{k}, t_k, \ldots, t_n, \underbrace{b, \ldots, b}_{k}]^T$$

Notice that the knot vector have a total of $n+k+1$ elements and the first k-elements all have the same value and the last k elements also have the same value. Succeding values in the knot vector should be non-decreasing.

Now let us differentiate equation (11) with respect to u. By the product rule we get

$$C'(u) = \sum_{i=0}^{n} N'_{i,k} P_i + N_{i,k} P'_i$$

And since $\{P_i\}$ are all constants and independent of u this reduces to

$$C'(u) = \sum_{i=0}^{n} N'_{i,k} P_i$$

From this it is easily seen that

$$C^{(j)}(u) = \sum_{i=0}^{n} N^{(j)}(i,k) P_i \tag{12}$$

3.2 Implementing The B-spline

Recalling our very first algorithm for computing the basis functions (see section 2.2) we can easily derive an algorithm for computing a single point on a B-spline.

```
Algorithm curvePoint(u,K,T,P)
  i = knot index, such that $t_i \leq u < t_{i+1}$
  N = BasisFuncs(i,u,K,T)
  C = 0
  For r=0 to K-1
    C = C + N[r]*P[i-K+1+r]
  Next r
  return C
End Algorithm
```

From equation (12) it is not hard to see how we can put our previously algorithm for computing the derivatives of the basis functions (see section 2.4) to practical use.

```
Algorithm curveDerivatives(u,K,T,P,J)
  i = knot index, such that $t_i \leq u < t_{i+1}$
  J = Min(J,K-1)
  dC = array(J+1)
  dN = derivatives(i,u,K,T,J)
  For j=0 to J
    dC[j] = 0
    For r=0 to K-1
      dC[j] = dC[j] + N[j][r]*P[i-K+1+r]
    Next r
  Next j
  return dC
End Algorithm
```

4 Global Interpolation

In this section we will look at the "inverse" problem of computing points on a B-spline. Instead we will try to compute the B-spline given some points on it. The approach we will take is called global interpolation.

Now let us call the initial given points on the spline we want to compute for break points and introduce the notation $\{X_i\}$ for them. The indices have the meaning that if $j < i$ then we should encoutner X_j before X_i if we walk along the wanted spline as the parameter u increases.

With the introduced formalism we can state our task at hand quite simple: Given a set of break points and a knot vector, we need to find the control points $\{P_i\}$ of the nonuniform B-spline, which passes through all the break points. Assume that we have a knot vector consisting of the following knot values

$$t_0, t_1, t_2, \ldots, t_{k-1}, t_k, \ldots, t_n, t_{n+1}, t_{n+2}, \ldots, t_{n+k}$$

Where

$$t_0 = \cdots . = t_{k-1}$$

And

$$t_{n+1} = \cdots = t_{n+k}$$

Given such a knot vector we know

$$\begin{aligned} C(t_{k-1}) &= P_0 \\ C(t_{n+1}) &= P_n \end{aligned} \tag{13}$$

$$\tag{14}$$

Each (unique) knot value will correspond to a single break point. From this we see that we must have a total of $n - k + 3$ break points.

$$X_0, X_1, X_2, \ldots, X_{n-k+1}, X_{n-k+2}$$

For each of these break points we will require that

$$C(t_{i+k-1}) = X_i \qquad \text{for} \qquad 0 \leq i \leq n - k + 2 \tag{15}$$

That is, we have a total of $n - k + 3$ equations, but we have $n + 1$ unknowns, and a spline with $n + k + 1$ knot values must have $n + 1$ control points. In short we might need a few more equations in order to solve our problem. Typically $k = 4$ and two more equations are needed in this case. We could simply pick

$$P_0 = P_1 \qquad \text{and} \qquad P_n = P_{n-1}$$

as the two extra equations. Now we can solve the system of linear equations given by equation (15). This strategy could be applied for an arbitary value of k. However, we wish to work in the case where $k = 4$, so let us set the value of k to 4 and see if we can derive an efficient way of solving equation (10). We start out by writting up $C(t_{i+3})$.

$$C(t_{i+3}) = \sum_{i=0}^{n} P_i N_{i,4}$$

Only $k = 4$ of the basis functions are potentially nonzero, since they have a support that overlaps with t_{i+3}. That is

$$C(t_{i+3}) = \sum_{i=0}^{n} P_i N_{i,4} = P_i N_{i,4} + P_{i+1} N_{i+1,4} + P_{i+2} N_{i+2,4} + P_{i+3} N_{i+3,4}$$

Now let us examine each of the 4 N-terms. We start out by looking at $N_{i,4}$. By our definition of a basis function we can construct the following tree, which shows how $N_{i,4}$ is computed.

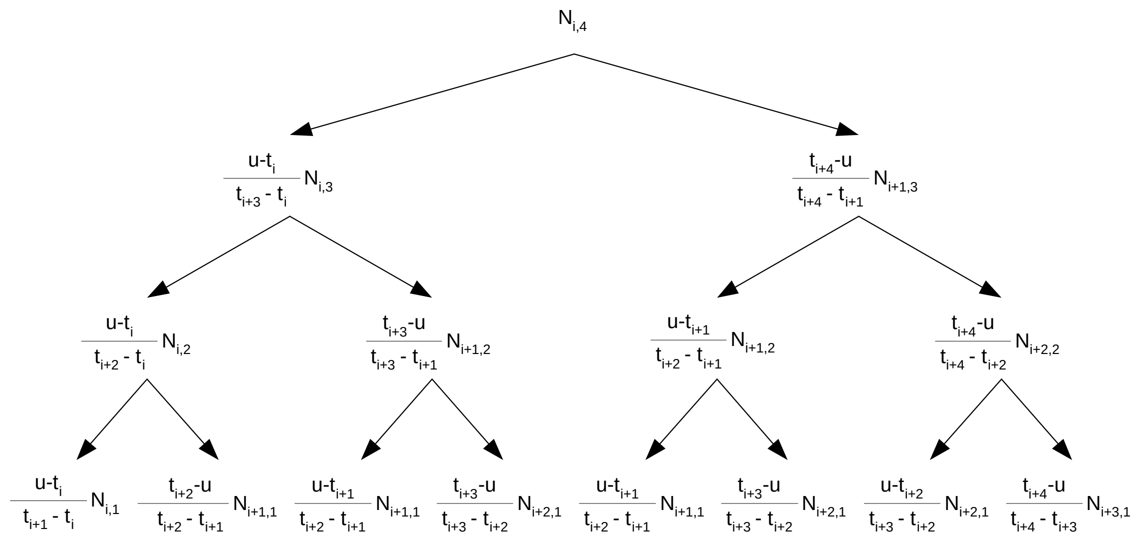
$N_{i,4}$
$\frac{u-t_i}{t_{i+3}-t_i} N_{i,3}$
$\frac{t_{i+4}-u}{t_{i+4}-t_{i+1}} N_{i+1,3}$
$\frac{u-t_i}{t_{i+2}-t_i} N_{i,2}$
$\frac{t_{i+3}-u}{t_{i+3}-t_{i+1}} N_{i+1,2}$
$\frac{u-t_{i+1}}{t_{i+2}-t_{i+1}} N_{i+1,2}$
$\frac{t_{i+4}-u}{t_{i+4}-t_{i+2}} N_{i+2,2}$
$\frac{u-t_i}{t_{i+1}-t_i} N_{i,1}$
$\frac{t_{i+2}-u}{t_{i+2}-t_{i+1}} N_{i+1,1}$
$\frac{u-t_{i+1}}{t_{i+2}-t_{i+1}} N_{i+1,1}$
$\frac{t_{i+3}-u}{t_{i+3}-t_{i+2}} N_{i+2,1}$
$\frac{u-t_{i+1}}{t_{i+2}-t_{i+1}} N_{i+1,1}$
$\frac{t_{i+3}-u}{t_{i+3}-t_{i+2}} N_{i+2,1}$
$\frac{u-t_{i+2}}{t_{i+3}-t_{i+2}} N_{i+2,1}$
$\frac{t_{i+4}-u}{t_{i+4}-t_{i+3}} N_{i+3,1}$

Figure 1: The computation of the basis function.

Since we know that $N_{i,4}$ should be evaluated at t_{i+3}, we can easily write up an explicit expression for $N_{i,4}$, from the tree in figure 1 we see that it would be the terms along the path from the rightmost leaf up to the root of the tree.

$$N_{i,4} = \frac{(t_{i+4} - t_{i+3})^2}{(t_{i+4} - t_{i+1})(t_{i+4} - t_{i+2})}$$

If we write up similar trees for $N_{i+1,4}, N_{i+2,4}$, and $N_{i+3,4}$ and evaluate them at t_{i+3} then we would derive the remaining expressions we need.

$$\begin{aligned} N_{i+1,4} &= \frac{(t_{i+5} - t_{i+3})(t_{i+3} - t_{i+2})}{(t_{i+5} - t_{i+2})(t_{i+4} - t_{i+2})} + \frac{(t_{i+3} - t_{i+1})(t_{i+4} - t_{i+3})}{(t_{i+4} - t_{i+1})(t_{i+4} - t_{i+2})} \\ N_{i+2,4} &= \frac{(t_{i+3} - t_{i+2})^2}{(t_{i+5} - t_{i+2})(t_{i+4} - t_{i+2})} \\ N_{i+3,4} &= 0 \end{aligned}$$

Now let us write up all the equations that determine the break points.

$$\begin{array}{lclclcl} i=0 & : & C(t_3) & = & P_1 & = & X_0 \\ i=1 & : & C(t_4) & = & \alpha_1 P_1 + \beta_1 P_2 + \gamma_1 P_3 & = & X_1 \\ \dots & & \dots & & & & \\ i=i & : & C(t_{i+3}) & = & \alpha_i P_i + \beta_i P_{i+1} + \gamma_i P_{i+2} & = & X_i \\ \dots & & \dots & & & & \\ i=n-2 & : & C(t_{n+1}) & = & P_{n-1} & = & X_{n-2} \end{array}$$

Where

$$\begin{aligned} \alpha_i &= \frac{(t_{i+4}-t_{i+3})^2}{(t_{i+4}-t_{i+1})(t_{i+4}-t_{i+2})} \\ \beta_i &= \frac{(t_{i+5}-t_{i+3})(t_{i+3}-t_{i+2})}{(t_{i+5}-t_{i+2})(t_{i+4}-t_{i+2})} + \frac{(t_{i+3}-t_{i+1})(t_{i+4}-t_{i+3})}{(t_{i+4}-t_{i+1})(t_{i+4}-t_{i+2})} \\ \gamma_i &= \frac{(t_{i+3}-t_{i+2})^2}{(t_{i+5}-t_{i+2})(t_{i+4}-t_{i+2})} \end{aligned}$$

All these $n-1$ equations can be expressed in matrix form, as shown below

$$\begin{bmatrix} 1 & & & & & & \\ \alpha_1 & \beta_1 & \gamma_1 & & & & \\ & & & \dots & & & \\ & & & \dots & \alpha_{n-3} & \beta_{n-3} & \gamma_{n-3} \\ & & & & & & 1 \end{bmatrix} \begin{bmatrix} P_1 \\ \\ \\ \\ P_{n-1} \end{bmatrix} = \begin{bmatrix} X_0 \\ \\ \\ \\ X_{n-2} \end{bmatrix}$$

Looking at this matrix equation, we observe that it has a particulary nice looking shape, such a shape is called tridiagonal. It is particular easy to solve a system of tridiagonal linear equations (see [7]) and by doing so we can compute all the control points.

Of course we have omitted a little detail: How should the knot vector be constructed if we are only given the break points? A good approach to this problem is to use a heuristic, which is known as the "chord length heuristic". Basically this means that we will require that the following condition is always fulfilled

$$\frac{t_{i+4}-t_{i+3}}{t_{i+5}-t_{i+4}} = \frac{|X_{i+1}-X_i|}{|X_{i+2}-X_{i+1}|} \tag{16}$$

5 Cubic Curve Decomposition

Nonuniform B-splines are easily converted into a composition of corresponding bezier curve segments by performing an operation on them which is called knot insertion. We have devoted this section to treat all the details on how one takes a nonuniform B-spline and ends up with a composition of cubic bezier curve segements.

B-splines are attractive due to two properties. They have local support and they guarantee continuity across adjacent curve segments. These properties are not mutually present when having a composition of bezier curve segements. Changing a control point on one curve segement means that one has to change the control points on all other curve segements to ensure continuity. However a bezier curve composition is far better to work with for rendering and evaluation. In our opinion it pays off in terms of performance to convert a B-spline to a composition of bezier curve segements when one is finished with modelling the final shape of the spline.

5.1 The Cubic Bezier Curve and the B-Spline Connection

The best way to see how a cubic nonuniform B-spline corresponds to a cubic bezier curve is by direct computation. Assume that we have the nonuniform B-spline

$$C(u) = \sum_{i=0}^{3} P_i N_{i,4}$$

Defined on the knot vector

$$T = [0,0,0,0,1,1,1,1]^T$$

We say that such a knot vector has no internal knots and that its multiplicity of all its knots are $k = 4$. We immediately see that only four basis functions are involved, these are

$$N_{0,4}, N_{1,4}, N_{2,4}, N_{3,4}$$

Looking at figure 1 it is fairly easy to evaluate these four basis functions. We immediately see that only those paths with leaves where the numerator $t_{j+1} - t_j$ are nonzero contributes to the value of the N-function. By direct substitution we derive

$$\begin{aligned} N_{0,4} &= (1-u)^3 \\ N_{1,4} &= 3u(1-u)^2 \\ N_{2,4} &= 3u^2(1-u) \\ N_{3,4} &= u^3 \end{aligned}$$

Cleaning up a bit we get

$$\begin{aligned} N_{0,4} &= -u^3 + 3u^2 - 3u + 1 \\ N_{1,4} &= 3u^3 - 6u^2 + 3u \\ N_{2,4} &= -3u^3 + 3u^2 \\ N_{3,4} &= u^3 \end{aligned}$$

Plugging our results into the original summation for the B-spline and using matrix notation we derive

$$C(u) = \left[u^3, u^2, u, 1\right] \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{bmatrix} \tag{17}$$

But this is actually the matrix definition of a cubic bezier curve. There is no doubt by now that there definitely is a connection between the spline and bezier curve representations. However we do have a little problem: Not all nonuniform B-splines have the particular nice looking knot vector as we used to show the connection with bezier curves. What should we do about a general looking B-spline? Let us for the moment being assume that every knot value in the general B-spline have multiplicity k (in the next subsection we will start on attacking the problem of twisting the spline into this form). The “general” cubic nonuniform B-spline can then be seen as a sequence of separate nonuniform B-splines because at each unique knot value the first $k-1$ derivatives vanishes (see section 7.1). Each “sequence” of this special looking spline looks almost like the one we used in our derivation above the only difference is that the knot values are not zeroes and ones. Instead the knot vector looks like this

$$T = \left[t_i, t_i, t_i, t_i, t_{i+1}, t_{i+1}, t_{i+1}, t_{i+1}\right]^T$$

The indices are the ones from the original spline with multiplicity 1. We can easily get the zeroes simply be subtracting the “left” knot value from all the knot values. This is perfectly legal because we are looking at a local segment of the spline. We end up with a knot vector like the following

$$T = [0,0,0,0,s,s,s,s]^T$$

Where

$$s = t_{i+1} - t_i$$

This knot vector allmost look like the one we want except for the s-value, which is not necessarily 1. If we repeat the above computation we will derive at a cubic bezier curve where its parameter has been scaled by s. We can conclude that we still have the connection between cubic bezier curves and nonuniform B-splines.

Let us try to get rid of this scaling problem. We introduce the terminology of global and local parameters. The global parameter is denoted by U and it is the parameter that was used to define the original nonuniform B-spline with (see (1)). We now use u for the local parameter of the i'th curve segment $B(u)$ of the original spline. We want

$$B(u) = C(U)$$

If we compute u as follows

$$u = \frac{U - t_i}{t_{i+1} - t_i}$$

Then our property is fulfilled. Another nice property is that u runs from 0 to 1.

5.2 Global and Local Parameter Conversion of Derivatives

If we have a local bezier segement $B_i(u)$ of $C(U)$ where we know

$$C(U) = B_i(u)$$

and recall that the conversion between the local and global spline parameter is given by

$$u = f(U)$$

where

$$f(U) = \frac{U - t_i}{t_{i+1} - t_i}$$

By straigthforward differentiation we get

$$\frac{d}{dU}C(U) = \frac{d}{dU}B_i(f(U)) = B_i'(f(U))f'(U)$$

where

$$f'(U) = \frac{1}{t_{i+1} - t_i}$$

Putting it together we get the conversion rule for the first order derivative we were looking for.

$$C'(U) = \frac{1}{t_{i+1} - t_i}B_i'(f(U)) = \frac{1}{t_{i+1} - t_i}B_i'(u) \tag{18}$$

If we continue our differentiation we get

$$\frac{d^2}{dU^2}C(U) = \frac{d}{dU}B_i'(f(U))f'(U) = B_i''(f(U))f'(U)^2 + B_i'(f(U))f''(U)$$

And we can easily compute

$$f''(U) = \frac{d}{dU}\left(\frac{1}{t_{i+1} - t_i}\right) = 0$$

Substituting back into our computation we get the conversion rule we were looking for.

$$C''(U) = \left(\frac{1}{t_{i+1} - t_i}\right)^2 B_i''(f(U)) = \left(\frac{1}{t_{i+1} - t_i}\right)^2 B_i''(u) \tag{19}$$

By now the patteren should be obvious and we can conclude

$$C^{(j)}(U) = \left(\frac{1}{t_{i+1} - t_i}\right)^j B_i^{(j)}(f(U)) = \left(\frac{1}{t_{i+1} - t_i}\right)^j B_i^{(j)}(u) \tag{20}$$

5.3 Knot Insertion

We are now ready to take on the task of knot insertion. Even though the resulting algorithm is quite simple and bare some resemblence to the de Boor algorithm (see Appendix A) we need to take a detour to "divided differences" in order to prove the algorithm.

5.3.1 Divided Difference

We have some function $g(t)$ and a knot vector

$$T = [\dots, t_i, t_{i+1}, \dots, t_{i+k}, \dots]^T$$

Where

$$t_i \le t_{i+1} < t_{i+k}$$

For all values of i. We define the divided difference on g recursively[1] in the following manner. The zero'th divided difference is given by

$$\Delta_i g = g(t_i)$$

and for $k = 1$

$$\Delta_{i,i+1} g = \frac{\Delta_{i+1} g(t_i) - \Delta_i g(t_i)}{t_{i+1} - t_i}$$

The general k'th divided difference[2] is defined by

$$\Delta_{i,\dots,i+k} g = \frac{\Delta_{i,\dots,t_{r-1},t_{r+1},\dots,i+k} g - \Delta_{i,\dots,t_{s-1},t_{s+1},\dots,i+k} g}{t_s - t_r}$$

An important property of the divided difference, which we are going to use, is that it is symmetric, This is easily shown. It follows trivially in the case of $k = 0$. Using operator notation and assuming the $k-1$'th divided difference is symmetric for all $k > 2$, we notice that

$$\begin{aligned}
\Delta_{i,\dots,v,\dots,w,\dots,i+k} &= \frac{\Delta_{i+1,\dots,v,\dots,w,\dots,i+k} - \Delta_{i,\dots,v,\dots,w,\dots,i+k-1}}{t_{i+k} - t_i} \\
&= \frac{\Delta_{i+1,\dots,w,\dots,v,\dots,i+k} - \Delta_{i,\dots,w,\dots,v,\dots,i+k-1}}{t_{i+k} - t_i} \\
&= \Delta_{i,\dots,w,\dots,v,\dots,i+k}
\end{aligned}$$

This clearly shows that the divided difference is symmetric.

5.3.2 Leibniz' Formula

Later on we will use Leiniz' formula, so we will state it here, without proof due to space considerations and because the proof requires another definition of the divided difference than the one we are using. The interested reader should refer to [4] for a proof. If we have $f(t) = g(t)h(t)$ then Leibniz' formula states

$$\Delta_{i,\dots,i+k} f = \sum_{r=i}^{i+k} \Delta_{i,\dots,r} g \Delta_{r,\dots,i+k} h \tag{21}$$

[2]There are other ways to define the divided difference, see [4].
[2]We have ignored the case $t_i = \dots = t_{i+k}$, see [4] for details.

5.3.3 The Funny Equation

Later on we will use a rather funny equation

$$0 = (t_i - t_{i+k})\Delta_{i,\dots,i+k} + (t_j - t_i)\Delta_{i,\dots,i+k-1,j} + (t_{i+k} - t_j)\Delta_{j,i+1,\dots,i+k} \tag{22}$$

For now we will try to show that this funny equation is infact true. Let us start out by using the definition of the divided differences on the funny equation (22):

$$0 = (t_i - t_{i+k})\Delta_{i,\dots,i+k} + (t_j - t_i)\frac{\Delta_{i+1,\dots,i+k-1,j} - \Delta_{i,\dots,i+k-1}}{t_j - t_i} + (t_{i+k} - t_j)\frac{\Delta_{i+1,\dots,i+k} - \Delta_{j,i+1,\dots,i+k-1}}{t_{i+k} - t_j}$$

Cleaning up gives us

$$0 = \Delta_{i,\dots,i+k} + \frac{\Delta_{i+1,\dots,i+k-1,j} - \Delta_{j,i+1,\dots,i+k-1}}{t_i - t_{i+k}} - \frac{\Delta_{i+1,\dots,i+k} - \Delta_{i,\dots,i+k-1}}{t_{i+k} - t_i}$$

By symmetry of the divided difference we have

$$\Delta_{i+1,\dots,i+k-1,j} = \Delta_{j,i+1,\dots,i+k-1}$$

so our second term vanishes from our equation. Now looking at the third term we recognize the definition of the divided difference, that is:

$$\Delta_{i,\dots,i+k} = \frac{\Delta_{i+1,\dots,i+k} - \Delta_{i,\dots,i+k-1}}{t_{i+k} - t_i}$$

So the first and third terms cancel each other out, and we immediately see that the funny equation (22) is true.

5.3.4 The Ordinary B-spline Basis Function

The unnormalized k'th order B-spline basis function $M_{i,k}$, also called the ordinary B-spline basis function, is defined recursively. This is very similar to equation (1). We have for $k > 1$:

$$M_{i,k} = \frac{u - t_i}{t_{i+k} - t_i} M_{i,k-1} + \frac{t_{i+k} - u}{t_{i+k} - t_i} M_{i+1,k-1} \tag{23}$$

and for $k = 1$:

$$M_{i,1} = \begin{cases} \frac{1}{t_{i+1} - t_i} & \text{if } t_i \leq u < t_{i+1} \\ 0 & \text{otherwise} \end{cases}$$

The oridinary B-spline basis function can be turned into the normalized B-spline basis function

$$N_{i,k} = (t_{i+k} - t_i) M_{i,k} \tag{24}$$

This is all easily veryfied by straightforward computation and comparision with the definition of the normalized B-spline basis function (see equation (1)).

What has all this to do with divided differences? If we pick a special known function[3] for g

$$g(t) = \begin{cases} 0 & \text{if } t \leq u \\ (t - u)^{k-1} & \text{if } t > u \end{cases}$$

With our choice of g the k'th divided difference corresponds to the ordinary k'th order B-spline basis function, that is

$$M_{i,k} = \Delta_{i,\dots,i+k}\left((t - u)^{k-1}\right)_{>u} \tag{25}$$

[3]Notice we have defined g differently than in [4, 5].

Where we have adopted the notation $\left((t-u)^{k-1}\right)_{>u}$ to mean $(t-u)^{k-1}$ if $t > u$ and 0 otherwise. We will now try to show this by induction[4] on k . First we observe that for $k = 1$ we have

$$M_{i,1} = \Delta_{i,i+1}\left((t-u)^0\right)_{>u} = \frac{\Delta_{i+1}\left((t-u)^0\right)_{>u} - \Delta_i\left((t-u)^0\right)_{>u}}{t_{i+1}-t_i} = \frac{\left((t_{i+1}-u)^0\right)_{>u} - \left((t_i-u)^0\right)_{>u}}{t_{i+1}-t_i}$$

Now if $u < t_i$ then

$$M_{i,1} = \frac{(t_{i+1}-u)^0 - (t_i-u)^0}{t_{i+1}-t_i} = \frac{1-1}{t_{i+1}-t_i} = 0$$

If $t_i \le u < t_{i+1}$ then

$$M_{i,1} = \frac{(t_{i+1}-u)^0 - 0}{t_{i+1}-t_i} = \frac{1-0}{t_{i+1}-t_i} = \frac{1}{t_{i+1}-t_i}$$

And finally if $u \ge t_{i+1}$ then

$$M_{i,1} = \frac{0-0}{t_{i+1}-t_i} = 0$$

So we can conclude that equation (25) is true when $k = 1$. Now let us assume that equation (25) is true for $k = k-1$, and take a close look on the equation:

$$\left((t-u)^{k-1}\right)_{>u} = (t-u)\left((t-u)^{k-2}\right)_{>u}$$

We can think of this as the product of two functions, so let us try to use Leibniz' formula (21) on it.

$$\begin{aligned}\Delta_{i,\dots,i+k}\left((t-u)^{k-1}\right)_{>u} &= \Delta_i(t-u)\Delta_{i,\dots,i+k}\left((t-u)^{k-2}\right)_{>u} \\ &\quad + \Delta_{i,i+1}(t-u)\Delta_{i+1,\dots,i+k}\left((t-u)^{k-2}\right)_{>u} \\ &\quad + \Delta_{i,\dots,i+2}(t-u)\Delta_{i+2,\dots,i+k}\left((t-u)^{k-2}\right)_{>u} \\ &\quad \dots \\ &\quad + \Delta_{i,\dots,i+k}(t-u)\Delta_{i+k}\left((t-u)^{k-2}\right)_{>u}\end{aligned} \tag{26}$$

Now by straightforward computation we have

$$\begin{aligned}\Delta_i(t-u) &= (t_i-u) \\ \Delta_{i,i+1}(t-u) &= \frac{(t_{i+1}-u)-(t_i-u)}{t_{i+1}-t_i} = 1 \\ \Delta_{i,\dots,i+2}(t-u) &= \frac{\Delta_{i+1,i+2}(t-u)-\Delta_{i,i+1}(t-u)}{t_{i+2}-t_i} = \frac{1-1}{t_{i+2}-t_i} = 0 \\ \dots &\quad \dots \\ \Delta_{i,\dots,i+k}(t-u) &= 0\end{aligned}$$

Using these results equation (26) reduces to

$$\Delta_{i,\dots,i+k}\left((t-u)^{k-1}\right)_{>u} = (t_i-u)\Delta_{i,\dots,i+k}\left((t-u)^{k-2}\right)_{>u} + \Delta_{i+1,\dots,i+k}\left((t-u)^{k-2}\right)_{>u} \tag{27}$$

By the definition of the k'th divided difference we have

$$\Delta_{i,\dots,i+k} = \frac{\Delta_{i+1,\dots,i+k} - \Delta_{i,\dots,i+k-1}}{t_{i+k}-t_i}$$

Multiplying by $(t_i - u)$ gives us

$$(t_i-u)\Delta_{i,\dots,i+k} = \frac{(t_i-u)}{(t_{i+k}-t_i)}\left(\Delta_{i+1,\dots,i+k} - \Delta_{i,\dots,i+k-1}\right)$$

[4]Note that if we use the definition of g in [4, 5] then the base case will always fail.

Substituting this into equation (27) yields

$$\Delta_{i,\ldots,i+k}\left((t-u)^{k-1}\right)_{>u} = \frac{(t_i-u)}{(t_{i+k}-t_i)}\left(\Delta_{i+1,\ldots,i+k}\left((t-u)^{k-2}\right)_{>u} - \Delta_{i,\ldots,i+k-1}\left((t-u)^{k-2}\right)_{>u}\right) + \Delta_{i+1,\ldots,i+k}\left((t-u)^{k-2}\right)_{>u}$$

Now by induction this means that we have

$$M_{i,k} = \frac{(t_i-u)}{(t_{i+k}-t_i)}\left(M_{i+1,k-1} - M_{i,k-1}\right) + M_{i+1,k-1}$$

Rearranging the terms this equation can be rewritten into

$$M_{i,k} = \frac{u-t_i}{t_{i+k}-t_i}M_{1,k-1} + \frac{t_{i+k}-u}{t_{i+k}-t_i}M_{i+1,k-1}$$

But this is the definition of $M_{i,k}$, which means that we have proven the correspondence of $M_{i,k}$ and the k'th divided difference.

5.3.5 The Knot Insertion Algorithm

Imagine that we insert a new knot value, t^*, into the knot vector, T, where

$$t_r \leq t^* < t_{r+1}$$

Then we get a new knot vector

$$\hat{T} = [\ldots, t_r, t^*, t_{r+1}, \ldots, t_{r+k}, \ldots]^T$$

Observe that

$$\hat{t}_i = \begin{cases} t_i & \text{if } i \leq r \\ t^* & \text{if } i = r+1 \\ t_{i-1} & \text{if } i > r+1 \end{cases}$$

If we have a nonuniform B-spline, $C(u)$, defined on T then we want to find a new nonuniform B-spline, $\hat{C}(u)$, defined on $\hat{T}$, such that

$$C(u) = \hat{C}(u)$$

for all values of u. If we look at the definition of the nonuniform B-spline this means that we have to find all $\hat{P}_i$'s such that

$$\sum_{i=0}^{n} N_{i,k}P_i = \sum_{i=0}^{n+1} \hat{N}_{i,k}\hat{P}_i$$

Where the hat-notation, means that the quantity is defined with respect to $\hat{T}$. Recall that $N_{i,k}$ has support on the knot values $t_i, \ldots, t_{i+k}$. This means that t^* only affects the basis functions

$$N_{r-k+1,k}, \ldots, N_{r,k}$$

Since $C(u)$ and $\hat{C}(u)$ should have the same shape we can conclude that

$$\hat{N}_i = \begin{cases} N_i & \text{if } i < r-k+1 \\ N_{i-1} & \text{if } i > r+1 \end{cases}$$

From which it follows that we must have

$$\hat{P}_i = \begin{cases} P_i & \text{if } i < r-k+1 \\ P_{i-1} & \text{if } i > r+1 \end{cases}$$

Now this leaves us with the problem of determining the $\hat{P}_i$'s such that

$$\sum_{i=r-k+1}^{r} N_{i,k} P_i = \sum_{i=r-k+1}^{r+1} \hat{N}_{i,k} \hat{P}_i \tag{28}$$

To solve this problem we will try to write N in terms of $\hat{N}$. To accomplish this we will turn back to the funny equation (22). Let us set $t_j = t^*$ and rearrange the equation. Then we have

$$(t_{i+k} - t_i)\Delta_{i,\ldots,i+k} = (t^* - t_i)\Delta_{i,\ldots,i+k-1,*} + (t_{i+k} - t^*)\Delta_{*,i+1,\ldots,i+k}$$

Using equation (25) we get for all values of $i = r - k + 1$ to r.

$$(t_{i+k} - t_i)M_{i,k} = (t^* - t_i)\hat{M}_{i,k} + (t_{i+k} - t^*)\hat{M}_{i+1,k}$$

Let us convert the M-terms into N-terms by using equation (24)

$$N_{i,k} = \frac{t^* - t_i}{\hat{t}_{i+k} - \hat{t}_i}\hat{N}_{i,k} + \frac{t_{i+k} - t^*}{\hat{t}_{i+k+1} - \hat{t}_{i+1}}\hat{N}_{i+1,k}$$

Now we would like to shift notation from old knot values into new knot values, we can do this by observing that for any value of $i = r - k + 1$ to r we have

$$t_i = \hat{t}_i \quad \text{and} \quad t_{i+k} = \hat{t}_{i+k+1}$$

So we end up with

$$N_{i,k} = \frac{t^* - \hat{t}_i}{\hat{t}_{i+k} - \hat{t}_i}\hat{N}_{i,k} + \frac{\hat{t}_{i+k+1} - t^*}{\hat{t}_{i+k+1} - \hat{t}_{i+1}}\hat{N}_{i+1,k}$$

Now let us try to use this to rewrite the terms in the summation of $C(u)$, that is

$$\sum_{i=r-k+1}^{r} N_{i,k} P_i = \sum_{i=r-k+1}^{r} \left(\frac{t^* - \hat{t}_i}{\hat{t}_{i+k} - \hat{t}_i}\hat{N}_{i,k} + \frac{\hat{t}_{i+k+1} - t^*}{\hat{t}_{i+k+1} - \hat{t}_{i+1}}\hat{N}_{i+1,k} \right) P_i$$

Now let us write out the summation

$$\begin{aligned}
\sum_{i=r-k+1}^{r} N_{i,k} P_i \quad = \quad & \left(\frac{t^* - \hat{t}_{r-k+1}}{\hat{t}_{r+1} - \hat{t}_{r-k+1}}\hat{N}_{r-k+1,k} + \frac{\hat{t}_{r+2} - t^*}{\hat{t}_{r+2} - \hat{t}_{r-k+2}}\hat{N}_{r-k+2,k} \right) P_{r-k+1} \\
& + \left(\frac{t^* - \hat{t}_{r-k+2}}{\hat{t}_{r+2} - \hat{t}_{r-k+2}}\hat{N}_{r-k+2,k} + \frac{\hat{t}_{r+3} - t^*}{\hat{t}_{r+3} - \hat{t}_{r-k+3}}\hat{N}_{r-k+3,k} \right) P_{r-k+2} \\
& \ldots \\
& + \left(\frac{t^* - \hat{t}_r}{\hat{t}_{r+k} - \hat{t}_r}\hat{N}_{r,k} + \frac{\hat{t}_{r+k+1} - t^*}{\hat{t}_{r+k+1} - \hat{t}_{r+1}}\hat{N}_{r+1,k} \right) P_r
\end{aligned}$$

Recall that $t^* = \hat{t}_{r+1}$, with this knowledge we immediately see that some of the fractions in the summation become 1.

$$\begin{aligned}
\sum_{i=r-k+1}^{r} N_{i,k} P_i \quad = \quad & \left(\hat{N}_{r-k+1,k} + \frac{\hat{t}_{r+2} - t^*}{\hat{t}_{r+2} - \hat{t}_{r-k+2}}\hat{N}_{r-k+2,k} \right) P_{r-k+1} \\
& + \left(\frac{t^* - \hat{t}_{r-k+2}}{\hat{t}_{r+2} - \hat{t}_{r-k+2}}\hat{N}_{r-k+2,k} + \frac{\hat{t}_{r+3} - t^*}{\hat{t}_{r+3} - \hat{t}_{r-k+3}}\hat{N}_{r-k+3,k} \right) P_{r-k+2} \\
& \ldots \\
& + \left(\frac{t^* - \hat{t}_r}{\hat{t}_{r+k} - \hat{t}_r}\hat{N}_{r,k} + \hat{N}_{r+1,k} \right) P_r
\end{aligned}$$

Now let us rewrite once more to collect terms of equal $\hat{N}$

$$\begin{aligned}\sum_{i=r-k+1}^{r} N_{i,k}P_i &= \hat{N}_{r-k+1,k}P_{r-k+1} \\ &+ \hat{N}_{r-k+2,k}\left(\frac{\hat{t}_{r+2}-t^*}{\hat{t}_{r+2}-\hat{t}_{r-k+2}}P_{r-k+1} + \frac{t^*-\hat{t}_{r-k+2}}{\hat{t}_{r+2}-\hat{t}_{r-k+2}}P_{r-k+2}\right) \\ &\dots \\ &+ \hat{N}_{r,k}\left(\frac{\hat{t}_{r+k}-t^*}{\hat{t}_{r+k}-\hat{t}_{r}}P_{r-1} + \frac{t^*-\hat{t}_{r}}{\hat{t}_{r+k}-\hat{t}_{r}}P_{r}\right) \\ &+ \hat{N}_{r+1,k}P_r\end{aligned}$$

Observe that we now have a summation over $\hat{N}_i$ just like we wanted in equation (28) All we need now is to make things a little more nice looking. We do this by first expressing the fractions in terms of T and not $\hat{T}$.

$$\begin{aligned}\sum_{i=r-k+1}^{r} N_{i,k}P_i &= \hat{N}_{r-k+1,k}P_{r-k+1} \\ &+ \hat{N}_{r-k+2,k}\left(\frac{t_{r+1}-t^*}{t_{r+1}-t_{r-k+2}}P_{r-k+1} + \frac{t^*-t_{r-k+2}}{t_{r+1}-t_{r-k+2}}P_{r-k+2}\right) \\ &\dots \\ &+ \hat{N}_{r,k}\left(\frac{t_{r+k-1}-t^*}{t_{r+k-1}-t_{r}}P_{r-1} + \frac{t^*-t_{r}}{t_{r+k-1}-t_{r}}P_{r}\right) \\ &+ \hat{N}_{r+1,k}P_r\end{aligned}$$

Finally we will introduce the notation

$$\alpha_i = \frac{t^*-t_i}{t_{i+k-1}-t_i}$$

From which we observe that

$$(1-\alpha_i) = \frac{t_{i+k-1}-t^*}{t_{i+k-1}-t_i}$$

With the new notation at hand we have

$$\begin{aligned}\sum_{i=r-k+1}^{r} N_{i,k}P_i &= \hat{N}_{r-k+1,k}P_{r-k+1} \\ &+ \hat{N}_{r-k+2,k}\left((1-\alpha_{r-k+2})P_{r-k+1} + \alpha_{r-k+2}P_{r-k+2}\right) \\ &\dots \\ &+ \hat{N}_{r,k}\left((1-\alpha_r)P_{r-1} + \alpha_r P_r\right) \\ &+ \hat{N}_{r+1,k}P_r\end{aligned}$$

From which we conclude that the solution to equation (28) is

$$\hat{P}_i = \begin{cases} P_i & \text{if } i = r-k+1 \\ (1-\alpha_i)P_{i-1} + \alpha_i P_i & \text{if } r-k+2 \le i \le r \\ P_{i-1} & \text{if } i = r+1 \end{cases}$$

Putting it all together we can now state the knot insertion algorithm. We have

$$\hat{C}(u) = \sum_{i=0}^{n+1} \hat{N}_{i,k}\hat{P}_i$$

with

$$\hat{P}_i = (1 - \alpha_i) P_{i-1} + \alpha_i P_i$$

Where

$$\alpha_i = \begin{cases} 1 & \text{if } i \leq r - k + 1 \\ \frac{t^* - t_i}{t_{i+k-1} - t_i} & \text{if } r - k + 2 \leq i \leq r \\ 0 & \text{if } i \geq r + 1 \end{cases}$$

If we compare the algorithm with the de Boor algorithm (see Appendix A) then we will discover that the newly computed control points correspond to the the de Boor points $P^1_r, P^1_{r-1}, \ldots, P^1_{r-k+2}$.

6 Accumulated Arc Length Table

In the previous section we explained how the spline could be decomposed into a sequence of bezier curves. Sometimes we are interrested in quickly finding the bezier curve segment which corresponds to a given arc length value along the spline $C(u)$. For instance if we do spline driven animation.

Stated mathematically we are given an arc length value s and we are now looking for the bezier curve segment B_i, such that

$$S(t_i) \leq s < S(t_{i+1})$$

One way to quickly determine the curve segment is by using an accumulated arc length table. After having found the decomposition of the spline into bezier curve segments, $B_0, \ldots, B_{n-1}$, this table is easily constructed by computing the total arc length of each bezier curve segment S_{B_i} and letting entry i store the value $\sum_{j=0}^{i-1} S_{B_j}$. We can now perform a linear search through the table to find the segment we are looking for, or even better we can do a binary search. The following pseudo code shows how a linear search can be done.

```
Algorithm lookupSegment(s)
  int i=0;
  do{
    i++;
  }while(ArcLengthTable[i]<s);
  i--;
  return i;
End Algorithm
```

This sort of strategy is also known under the name accumulated chord length and it comes in different flavors, but the basic idea still remains the same. For more details about the other “flavors” see [2].

7 The Regular Cubic Nonuniform B-spline

In this section we will look at a subclass of the B-splines, which is known as regular B-splines. A regular B-spline is defined by having nonzero first order derivatives everywhere. Regular B-splines are usefull for describing a physical meaningfull motion. That is the center of mass trajectory of a real world object (see for instance [8]).

In this section we will formulate different conditions on how one should define a B-spline such that it would be a regular B-spline. We will only consider cubic B-splines.

7.1 First Order Derivatives at Knot Values

Let us derive an equation for the derivative at the knot value t_i

$$\begin{aligned}
C'(t_i) &= N'_{i-3,4}P_{i-3} + N'_{i-2,4}P_{i-2} + N'_{i-1,4}P_{i-1} + N'_{i,4}P_i \\
&= \left(\frac{3}{t_i - t_{i-3}}N_{i-3,3} - \frac{3}{t_{i+1} - t_{i-2}}N_{i-2,3}\right)P_{i-3} + \\
&\quad \left(\frac{3}{t_{i+1} - t_{i-2}}N_{i-2,3} - \frac{3}{t_{i+2} - t_{i-1}}N_{i-1,3}\right)P_{i-2} + \\
&\quad \left(\frac{3}{t_{i+2} - t_{i-1}}N_{i-1,3} - \frac{3}{t_{i+3} - t_i}N_{i,3}\right)P_{i-1} + \\
&\quad \left(\frac{3}{t_{i+3} - t_i}N_{i,3} - \frac{3}{t_{i+4} - t_{i+1}}N_{i+1,3}\right)P_i
\end{aligned}$$

Recall that $N_{i-3,3} = 0$ and $N_{i+1,3} = 0$ since we are looking at the i'th knot value of a third order B-spline. Using this knowledge we can rewrite our equation into the following sum of differences

$$\begin{aligned}
C'(t_i) &= \frac{3N_{i-2,3}}{t_{i+1} - t_{i-2}}\left(P_{i-2} - P_{i-3}\right) + \\
&\quad \frac{3N_{i-1,3}}{t_{i+2} - t_{i-1}}\left(P_{i-1} - P_{i-2}\right) + \\
&\quad \frac{3N_{i,3}}{t_{i+3} - t_i}\left(P_i - P_{i-1}\right)
\end{aligned}$$

Let us evaluate the basis functions by using de Cox's definition (see equation (1)), we start by looking at $N_{i,3}$.

$$\begin{tikzcd}
 & & \frac{u-t_i}{t_{i+1}-t_i}N_{i,1} \arrow[dl] \\
 & \frac{u-t_i}{t_{i+2}-t_i}N_{i,2} \arrow[ddl] & \\
 & & \frac{t_{i+2}-u}{t_{i+2}-t_{i+1}}N_{i+1,1} \arrow[ul] \\
N_{i,3} & & \\
 & & \frac{u-t_{i+1}}{t_{i+2}-t_{i+1}}N_{i+1,1} \arrow[dl] \\
 & \frac{t_{i+3}-u}{t_{i+3}-t_{i+1}}N_{i+1,2} \arrow[uul] & \\
 & & \frac{t_{i+3}-u}{t_{i+3}-t_{i+2}}N_{i+2,1} \arrow[ul]
\end{tikzcd}$$

Recall that only $N_{i,1}$ is nonzero for $u = t_i$. From this we see that $N_{i,3}$ is always zero. Now let us look at $N_{i-1,3}$

$$\begin{tikzcd}
 & & \frac{u-t_{i-1}}{t_i-t_{i-1}}N_{i-1,1} \arrow[dl] \\
 & \frac{u-t_{i-1}}{t_{i+1}-t_{i-1}}N_{i-1,2} \arrow[ddl] & \\
 & & \frac{t_{i+1}-u}{t_{i+1}-t_i}N_{i,1} \arrow[ul] \\
N_{i-1,3} & & \\
 & & \frac{u-t_i}{t_{i+1}-t_i}N_{i,1} \arrow[dl] \\
 & \frac{t_{i+2}-u}{t_{i+2}-t_i}N_{i,2} \arrow[uul] & \\
 & & \frac{t_{i+2}-u}{t_{i+2}-t_{i+1}}N_{i+1,1} \arrow[ul]
\end{tikzcd}$$

Setting $u = t_i$ we see that

$$N_{i-1,3} = \frac{t_i - t_{i-1}}{t_{i+1} - t_{i-1}} \geq 0$$

Finally we only have to evaluate $N_{i-2,3}$

$$\begin{tikzcd}
 & & \frac{u-t_{i-2}}{t_{i-1}-t_{i-2}}N_{i-2,1} \arrow[dl] \\
 & \frac{u-t_{i-2}}{t_i-t_{i-2}}N_{i-2,2} \arrow[ddl] & \\
 & & \frac{t_i-u}{t_i-t_{i-1}}N_{i-1,1} \arrow[ul] \\
N_{i-2,3} & & \\
 & & \frac{u-t_{i-1}}{t_i-t_{i-1}}N_{i-1,1} \arrow[dl] \\
 & \frac{t_{i+1}-u}{t_{i+1}-t_{i-1}}N_{i-1,2} \arrow[uul] & \\
 & & \frac{t_{i+1}-u}{t_{i+1}-t_i}N_{i,1} \arrow[ul]
\end{tikzcd}$$

Again we have $u = t_i$, and from this we conclude that

$$N_{i-2,3} = \frac{t_{i+1} - t_i}{t_{i+1} - t_{i-1}} \geq 0$$

Substituting the basis functions we just found into our equation for the first order derivative we obtain

$$C'(t_i) = \frac{3}{t_{i+1} - t_{i-1}} \left(\frac{t_{i+1} - t_i}{t_{i+1} - t_{i-2}} (P_{i-2} - P_{i-3}) + \frac{t_i - t_{i-1}}{t_{i+2} - t_{i-1}} (P_{i-1} - P_{i-2}) \right)$$

If we wanted to make $C'(t_i)$ equal to zero how could we then go about this? From our equation we can by straightforward inspection derive the following complete list of conditions, which will make $C'(t_i)$ equal to zero

- $P_{i-1} = P_{i-2} = P_{i-3}$
- $P_{i-1} = P_{i-2}$ and $t_{i+1} = t_i$
- $P_{i-2} = P_{i-3}$ and $t_i = t_{i-1}$
- $t_{i+1} = t_i = t_{i-1}$
- The final condition is a bit more difficult to derive

  $$0 = \frac{t_{i+1} - t_i}{t_{i+1} - t_{i-2}} (P_{i-2} - P_{i-3}) + \frac{t_i - t_{i-1}}{t_{i+2} - t_{i-1}} (P_{i-1} - P_{i-2})$$

  From which we get

  $$-\frac{(t_{i+1} - t_i)(t_{i+2} - t_{i-1})}{(t_{i+1} - t_{i-2})(t_i - t_{i-1})} (P_{i-2} - P_{i-3}) = (P_{i-1} - P_{i-2})$$

  This tells us that the three succeding control points P_{i-3},P_{i-2} and P_{i-1} all have to lie on the same line and the magnitude of $(P_{i-2} - P_{i-3})$ must be a scalar multiple of the magnitude of $(P_{i-1} - P_{i-2})$.

With all this theory to our disposal we surely know how to avoid that $C'(t_i)$ ever becomes zero for any knotvalue. $C'(t_i)$ is always nonzero for any knotvalue t_i if the following three criteria is fulfilled.

1. No knotvalue has multiplicity greather than one.
2. All control points are uniqe.
3. We never have three (or more) succeding control points lying on the same line.

Note that if we put this theory to practical use then it is slightly more restrictive than it need be, that is it does exclude some splines having nonzero first order derivatives.

7.2 First Order Derivatives between Knot Values

By now we have complete knowledge about how we can control $C'(u)$ at any knotvalue, t_i, but what happens between the knotvalues? That is how do we ensure that

$$C'(u) \neq 0$$

For an u-value inbeteween two knotvalues? That is

$$t_i < u < t_{i+1}$$

Let us start writting up the equation for $C'(u)$ again

$$\begin{aligned} C'(t_i) &= \frac{3N_{i-2,3}}{t_{i+1}-t_{i-2}}\left(P_{i-2}-P_{i-3}\right)+ \\ &\quad \frac{3N_{i-1,3}}{t_{i+2}-t_{i-1}}\left(P_{i-1}-P_{i-2}\right)+ \\ &\quad \frac{3N_{i,3}}{t_{i+3}-t_{i}}\left(P_{i}-P_{i-1}\right) \end{aligned}$$

And like previously we will now evaluate the basis functions.

$$\begin{aligned} N_{i-2,3} &= \frac{\left(t_{i+1}-u\right)^2}{\left(t_{i+1}-t_i\right)\left(t_{i+1}-t_{i-1}\right)} \\ N_{i-1,3} &= \frac{\left(t_{i+1}-u\right)}{\left(t_{i+1}-t_i\right)}\frac{\left(u-t_{i-1}\right)}{\left(t_{i+1}-t_{i-1}\right)}+\frac{\left(u-t_i\right)}{\left(t_{i+1}-t_i\right)}\frac{\left(t_{i+2}-u\right)}{\left(t_{i+2}-t_i\right)} \\ N_{i,3} &= \frac{\left(u-t_i\right)^2}{\left(t_{i+1}-t_i\right)\left(t_{i+2}-t_i\right)} \end{aligned}$$

From our knowledge of the knot vector

$$t_{i-1} < t_i < u < t_{i+1} < t_{i+2}$$

We can conclude that $N_{i,3}$,$N_{i-1,3}$ and $N_{i-2,3}$ are always positive and never zero. So our equation for the first order derivative reduces to the equation.

$$\begin{aligned} C'(t_i) &= a\left(P_{i-2}-P_{i-3}\right)+ \\ &\quad b\left(P_{i-1}-P_{i-2}\right)+ \\ &\quad c\left(P_{i}-P_{i-1}\right) \end{aligned}$$

Where

$$\begin{aligned} a &= \frac{3\left(t_{i+1}-u\right)^2}{\left(t_{i+1}-t_{i-2}\right)\left(t_{i+1}-t_i\right)\left(t_{i+1}-t_{i-1}\right)} > 0 \\ b &= \frac{3}{\left(t_{i+2}-t_{i-1}\right)}\left(\frac{\left(t_{i+1}-u\right)\left(u-t_{i-1}\right)}{\left(t_{i+1}-t_i\right)\left(t_{i+1}-t_{i-1}\right)}+\frac{\left(u-t_i\right)\left(t_{i+2}-u\right)}{\left(t_{i+1}-t_i\right)\left(t_{i+2}-t_i\right)}\right) > 0 \\ c &= \frac{3\left(u-t_i\right)^2}{\left(t_{i+3}-t_i\right)\left(t_{i+1}-t_i\right)\left(t_{i+2}-t_i\right)} > 0 \end{aligned}$$

All we have left to do is to solve the linear system

$$a\Delta_{i-2}+b\Delta_{i-1}+c\Delta_i = 0$$

Where $\Delta_i = P_i - P_{i-1}$ and since we allready have decided that none of the control points coincide we know that all Δ_i are nonzero. In other words we can only find a solution if the Δ_i's are lineary dependent. In three dimensions and higher it is quite easy to come up with a condition, which always ensure that $C'(u)$ is nonzero, simply make sure that no four succeding control points lie in the same plane. However in two or one dimensions it is unavoidable to have linear dependent Δ's, and the problem still persist in three dimensional space (or higher) if one wants the spline (or a subpart of it) to lie in a plane. So let us try to handle the linear dependence.

7.2.1 Two Linear Dependent Δ's

Recall that each a,b and c is a function of u, actually they are all second order polynomials. So assuming we have two lineary independent Δ's to our disposal then we have three posibilities.

$$\begin{aligned} s\Delta_i + t\Delta_{i-1} &= \Delta_{i-2} \\ v\Delta_{i-2} + w\Delta_i &= \Delta_{i-1} \\ x\Delta_{i-1} + y\Delta_{i-2} &= \Delta_i \end{aligned}$$

for some nonzero scalar values s,t,v,w,x and y. Now let us look closely at the first posibility.

$$\begin{aligned} a(u)\underbrace{(s\Delta_i + t\Delta_{i-1})}_{\Delta_{i-2}} + b(u)\Delta_{i-1} + c(u)\Delta_i &= 0 \\ (a(u)s + c(u))\Delta_i + (a(u)t + b(u))\Delta_{i-1} &= 0 \end{aligned}$$

Since Δ_i and Δ_{i-1} are lineary independent and different from zero, this can only occur if

$$\begin{aligned} a(u)s + c(u) &= 0 \\ a(u)t + b(u) &= 0 \end{aligned}$$

Similary the other two possibilites give rise to

$$\begin{aligned} b(u)v + a(u) &= 0 \\ b(u)w + c(u) &= 0 \end{aligned}$$

and

$$\begin{aligned} c(u)x + b(u) &= 0 \\ c(u)y + a(u) &= 0 \end{aligned}$$

In other words our problem has been reduced to determine if two parabola intersect and if they do then to determine the u-values where they intersect.

7.2.2 Three Linear Dependent Δ's

Following our derivation from previous section we have

$$a(u)x\Delta_i + b(u)y\Delta_i + c(u)z\Delta_i = 0$$

For some nonzero scalar values x,y and z (all nonzero). All this boils down to a single equation.

$$a(u)x + b(u)y + c(u)z = 0 \tag{29}$$

By now we are tempted to repeat our earlier idea and look at the linear dependency of the three polynomials $a(u)$,$b(u)$ and $c(u)$. If they all are linear independent then no solution exist and we are guaranteed that $C'(u)$ is always nonzero. If on the other hand we have some polynomials that are linear dependent then we are once again left with determining if and where two parabola intersect. To see this imagine we have

$$a(u) = vb(u) + wc(u)$$

Then equation (29) becomes

$$\begin{aligned} a(u)x + b(u)y + c(u)z &= 0 \\ (vb(u) + wc(u))x + b(u)y + c(u)z &= 0 \\ (vx + y)b(u) + (wx + z)c(u) &= 0 \\ \frac{(vx + y)}{(wx + z)}b(u) + c(u) &= 0 \end{aligned}$$

Just like we have seen previously.

7.2.3 Conclusion on Linear Dependency

We can conclude that we can only have $C'(u) = 0$ between t_i and t_{i+1} when two second order polynomials in u have an intersection in the interval $[t_i..t_{i+1}]$. In other words the first derivative can only vanish at isolated values of u, i.e. $C'(u) = 0$ between knot values can only occur at cusps.

8 Conclusion

In this paper we have presented the theory of open nonuniform B-splines. All the theory we have presented are selfcontained except for the Leibniz's formula. We have derived easily understandable c-style pseudocode for efficient implementation of open nonuniform B-splines.

Besides having presented the basic theory we have looked into some more advanced topics, which is frequently used when working in 3D, global interpolation, and curve decomposition.

Finally, we have treated regular B-splines in great detail and presented a set of "verification" rules for determining whetever a given open nonunifrom B-spline is regular or not. We currently speculate that these rules also could be used to construct an algorithm for interpolating a set of break points with a regular open nonuniform B-spline, but we leave this as a future research topic.

References

[1] L. Piegl and W. Tiller: *The NURBS Book*, Springer-Verlag Berlin Heidelberg New York. 1995.

[2] A. Watt and M. Watt: *Advanced Animation and Rendering Techniques*, Theory and Practice, Addison-Wesley 1992.

[3] J. Hoschek and D. Lasser: *Fundamentals of Computer Aided Geometric Design*, English translation 1993 by A K Peters, Ltd.

[4] Carl de Boor: *A Practical Guide to Splines*, Springer-Verlag, Applied Mathematical Sciences, vol. 27, 1978.

[5] Wolfgang Boehm: *Inserting new knots into B-spline curves*, Computer Aided Design 12 (1980) 199-201.

[6] Gerald Farin: *Curves and Surfaces for Computer Aided Geometric Design. A Practical Guide*, Third Edition, Academic Press, INC., Computer Science and Scientific Computing, 1993.

[7] William H. Press, Saul A. Teukolsky, William T. Vetterling and Brian P.Flannery: *Numerical Recipes in C: The Art of Scientific Computing*, Second Edition, Cambridge University Press, 1999.

[8] Knud Henriksen og Kenny Erleben: *Scripted Bodies and Spline Driven Motion*, Technical Report 02/18, Department of Computer Science University of Copenhagen, 2002.

A The de Boor Algorithm

In this appendix we will state the the de Boor algorithm and prove it. The de Boor algorithm can be stated as follows:

$$C(u) = \sum_{i=0}^{n+j} N_{i,k-j} P_i^j \qquad \text{for} \quad 0 \leq j \leq k-1 \tag{30}$$

Where

$$P_i^j = (1 - a_{i,j}) P_{i-1}^{j-1} + a_{i,j} P_i^{j-1} \qquad \text{for} \quad j > 0$$

and

$$a_{i,j} = \frac{t - t_i}{t_{i+k-j} - t_i} \qquad \text{and} \quad P_i^0 = P_i$$

The points P_i^j are called the de Boor points. Now we will prove de Boors algorithm. First let us write up the definition of a nonunform B-spline.

$$C(u) = \sum_{i=0}^{n} N_{i,k} P_i$$

If we now apply de Cox's definition for $N_{i,k}$ then we get

$$C(u) = \sum_{i=0}^{n} \frac{u - t_i}{t_{i+k-1} - t_i} N_{i,k-1} P_i + \sum_{i=0}^{n} \frac{t_{i+k} - u}{t_{i+k} - t_{i+1}} N_{i+1,k-1} P_i$$

This looks a bit nasty, but by applying a little mathematical trick of shifting the index of the second term by $i = i - 1$ we get

$$C(u) = \sum_{i=0}^{n} \frac{u - t_i}{t_{i+k-1} - t_i} N_{i,k-1} P_i + \sum_{i=1}^{n+1} \frac{t_{i+k-1} - u}{t_{i+k-1} - t_i} N_{i,k-1} P_{i-1}$$

Finally if we define $P_{-1} = 0$ and $P_{n+1} = 0$ then we are allowed to rewrite the two summations into a single summation

$$C(u) = \sum_{i=0}^{n+1} \frac{P_i (u - t_i) + P_{i-1} (t_{i+k-1} - u)}{t_{i+k-1} - t_i} N_{i,k-1}$$

Again we apply a little mathematical trick of adding and subtracting t_i from the second term in the denominator. That is

$$\begin{aligned} C(u) &= \sum_{i=0}^{n+1} \frac{P_i (u - t_i) + P_{i-1} (t_{i+k-1} - t_i + t_i - u)}{t_{i+k-1} - t_i} N_{i,k-1} \\ &= \sum_{i=0}^{n+1} \frac{P_i (u - t_i) + P_{i-1} (t_{i+k-1} - t_i - (u - t_i))}{t_{i+k-1} - t_i} N_{i,k-1} \\ &= \sum_{i=0}^{n+1} \left(P_i \alpha_i^1 + P_{i-1} \left(1 - \alpha_i^1\right)\right) N_{i,k-1} \end{aligned}$$

Where α_i^1 is defined as in the definition of the de Boor Algorithm. The final equation is derived by introducing the notation

$$P_i^1 = \left(1 - \alpha_i^1\right) P_{i-1} + \alpha_i^1 P_i$$

So we get

$$C(u) = \sum_{i=0}^{n+1} P_i^1 N_{i,k-1}$$

From all this we can conclude that we have proven de Boors algorithm in the case where $j = 1$. It is not hard to see that if we apply the same index shifting and definitions recursively we will end up with the de Boors algorithm in the general j'th case.

B Repeated Knot Insertion

If our spline, $C(U)$, do not have any sequences of identical control points or knot values then the local bezier segement $B_i(u)$ corresponding to the segment t_i to t_{i+1} will have geometry vectors, G_0,G_1,G_2 and G_3, which are uniqe and distinct. This postulate follows rather easily from direct computation.

Let us assume that we have a multiplicity of $k = 4$ for all knot values to the left of t_r and that all knot values to the right of t_r and t_r itself have a multiplicity of 1. Now imagine we have

$$t_r \leq t^* < t_{r+1}$$

This means that for a cubic spline (i.e. $k = 4$) we have $N_{r-3,4}, \ldots, N_{r,4} \neq 0$. From this we also know that it is the control points P_{r-3},P_{r-2},P_{r-1} and P_r which have to be "recomputed" in the new spline. The new de Boor points are generated as in the tableu below

$$\begin{tikzcd}
P^0_{r-3} \arrow[dr] & & \\
& (1-a_{r-2}) \arrow[dr] & \\
P^0_{r-2} \arrow[r] \arrow[dr] & a_{r-2} \arrow[r] & P^1_{r-2} \\
& (1-a_{r-1}) \arrow[dr] & \\
P^0_{r-1} \arrow[r] \arrow[dr] & a_{r-1} \arrow[r] & P^1_{r-1} \\
& (1-a_r) \arrow[dr] & \\
P^0_r \arrow[r] & a_r \arrow[r] & P^1_r
\end{tikzcd}$$

Where

$$\begin{aligned}
a_{r-2} &= \frac{(t_r - t_{r-2})}{(t_{r+1} - t_{r-2})} > 0 \\
a_{r-1} &= \frac{(t_r - t_{r-1})}{(t_{r+2} - t_{r-1})} > 0 \\
a_r &= \frac{(t_r - t_r)}{(t_{r+3} - t_r)} = 0
\end{aligned}$$

Now we can construct the new control points $\hat{P}$.

P_0	P_1	P_2	$\ldots$	P_{r-3}	P^1_{r-2}	P^1_{r-1}	P^1_r	P_r	P_{r+1}	$\ldots$	P_n
P_0	P_1	P_2	$\ldots$	P_{r-3}	P^1_{r-2}	P^1_{r-1}	P_{r-1}	P_r	P_{r+1}	$\ldots$	P_n
$\hat{P}_0$	$\hat{P}_1$	$\hat{P}_2$	$\ldots$	$\hat{P}_{r-3}$	$\hat{P}_{r-2}$	$\hat{P}_{r-1}$	$\hat{P}_r$	$\hat{P}_{r+1}$	$\hat{P}_{r+2}$	$\ldots$	$\hat{P}_{n+1}$

And the new knot vector $\hat{T}$.

t_0	t_1	$\ldots$	t_r	t^*	t_{r+1}	$\ldots$	t_{n+k}
t_0	t_1	$\ldots$	t_r	t_r	t_{r+1}	$\ldots$	t_{n+k}
$\hat{t}_0$	$\hat{t}_1$	$\ldots$	$\hat{t}_r$	$\hat{t}_{r+1}$	$\hat{t}_{r+2}$	$\ldots$	$\hat{t}_{n+k+1}$

We now have multiplicity of 2 for the knot value t_r, let us try to insert $t^* = t_r$ once more. This time we will discover that

$$\hat{t}_{r+1} \leq t^* < \hat{t}_{r+2}$$

From which we conclude that it is the control points $\hat{P}_{r-2}$, $\hat{P}_{r-1}$, $\hat{P}_r$ and $\hat{P}_{r+1}$, which have to be "recomputed" in the new spline.

$$\begin{tikzcd}
\hat{P}^0_{r-2} \arrow[dr] & & \\
& (1-a_{r-1}) \arrow[dr] & \\
\hat{P}^0_{r-1} \arrow[r] \arrow[dr] & a_{r-1} \arrow[r] & \hat{P}^1_{r-1} \\
& (1-a_r) \arrow[dr] & \\
\hat{P}^0_r \arrow[r] \arrow[dr] & a_r \arrow[r] & \hat{P}^1_r \\
& (1-a_{r+1}) \arrow[dr] & \\
\hat{P}^0_{r+1} \arrow[r] & a_{r+1} \arrow[r] & \hat{P}^1_{r+1}
\end{tikzcd}$$

Where

$$\begin{aligned}
a_{r-1} &= \frac{(t_r - \hat{t}_{r-1})}{(\hat{t}_{r+2} - \hat{t}_{r-1})} = \frac{t_r - t_{r-1}}{t_{r+1} - t_{r-1}} > 0 \\
a_r &= \frac{(t_r - \hat{t}_r)}{(\hat{t}_{r+3} - \hat{t}_r)} = \frac{t_r - t_r}{t_{r+2} - t_r} = 0 \\
a_{r+1} &= \frac{(t_r - \hat{t}_{r+1})}{(\hat{t}_{r+4} - \hat{t}_{r+1})} = \frac{t_r - t_r}{t_{r+2} - t_r} = 0
\end{aligned}$$

With this information we can write up the new control points $\tilde{P}$

$\hat{P}_0$	$\dots$	$\hat{P}_{r-3}$	$\hat{P}_{r-2}$	$\hat{P}^1_{r-1}$	$\hat{P}^1_r$	$\hat{P}^1_{r+1}$	$\hat{P}_{r+1}$	$\dots$	$\hat{P}_{n+1}$
P_0	$\dots$	P_{r-3}	P^1_{r-2}	$\hat{P}^1_{r-1}$	$\hat{P}^0_{r-1}$	$\hat{P}^0_r$	P_r	$\dots$	P_n
P_0	$\dots$	P_{r-3}	P^1_{r-2}	$\hat{P}^1_{r-1}$	P^1_{r-1}	P_{r-1}	P_r	$\dots$	P_n
$\tilde{P}_0$	$\dots$	$\tilde{P}_{r-3}$	$\tilde{P}_{r-2}$	$\tilde{P}_{r-1}$	$\tilde{P}_r$	$\tilde{P}_{r+1}$	$\tilde{P}_{r+2}$	$\dots$	$\tilde{P}_{n+2}$

And the new knot vector $\tilde{T}$

$\hat{t}_0$	$\hat{t}_1$	$\dots$	$\hat{t}_r$	$\hat{t}_{r+1}$	t^*	$\hat{t}_{r+2}$	$\dots$	$\hat{t}_{n+k+1}$
t_0	t_1	$\dots$	t_r	t_r	t_r	t_{r+1}	$\dots$	t_{n+k}
$\tilde{t}_0$	$\tilde{t}_1$	$\dots$	$\tilde{t}_r$	$\tilde{t}_{r+1}$	$\tilde{t}_{r+2}$	$\tilde{t}_{r+3}$	$\dots$	$\tilde{t}_{n+k+2}$

We know have a multiplicity of 3 for the knot value t_r, let us see what happens upon the last knot insertion. First of all we see that we have.

$$\tilde{t}_{r+2} \leq t^* < \tilde{t}_{r+3}$$

From which we know that we have to recompute the following control points.

$$\begin{tikzcd}
\tilde{P}^0_{r-1} \arrow[dr] & & \\
& (1-a_r) \arrow[dr] & \\
\tilde{P}^0_r \arrow[r] \arrow[dr] & a_r \arrow[r] & \tilde{P}^1_r \\
& (1-a_{r+1}) \arrow[dr] & \\
\tilde{P}^0_{r+1} \arrow[r] \arrow[dr] & a_{r+1} \arrow[r] & \tilde{P}^1_{r+1} \\
& (1-a_{r+2}) \arrow[dr] & \\
\tilde{P}^0_{r+2} \arrow[r] & a_{r+2} \arrow[r] & \tilde{P}^1_{r+2}
\end{tikzcd}$$

Where

$$\begin{aligned}
a_r &= \frac{(t_r - \tilde{t}_r)}{(\tilde{t}_{r+3} - \tilde{t}_r)} = \frac{t_r - t_r}{t_{r+1} - t_r} = 0 \\
a_{r+1} &= \frac{(t_r - \tilde{t}_{r+1})}{(\tilde{t}_{r+4} - \tilde{t}_{r+1})} = \frac{t_r - t_r}{t_{r+2} - t_r} = 0 \\
a_{r+2} &= \frac{(t_r - \tilde{t}_{r+2})}{(\tilde{t}_{r+5} - \tilde{t}_{r+2})} = \frac{t_r - t_r}{t_{r+3} - t_r} = 0
\end{aligned}$$

Finally we write up the new control points ρ.

$\tilde{P}_0$	$\dots$	$\tilde{P}_{r-3}$	$\tilde{P}_{r-2}$	$\tilde{P}_{r-1}$	$\tilde{P}^1_r$	$\tilde{P}^1_{r+1}$	$\tilde{P}^1_{r+2}$	$\tilde{P}_{r+2}$	$\dots$	$\tilde{P}_{n+2}$
P_0	$\dots$	P_{r-3}	P^1_{r-2}	$\hat{P}^1_{r-1}$	$\tilde{P}_{r-1}$	$\tilde{P}_r$	$\tilde{P}_{r+1}$	P_r	$\dots$	P_n
P_0	$\dots$	P_{r-3}	P^1_{r-2}	$\hat{P}^1_{r-1}$	$\hat{P}^1_{r-1}$	P^1_{r-1}	P_{r-1}	P_r	$\dots$	P_n
ρ_0	$\dots$	ρ_{r-3}	ρ_{r-2}	ρ_{r-1}	ρ_r	ρ_{r+1}	ρ_{r+2}	ρ_{r+3}	$\dots$	ρ_{n+3}

And the new knot vector τ.

$\tilde{t}_0$	$\dots$	$\tilde{t}_r$	$\tilde{t}_{r+1}$	$\tilde{t}_{r+2}$	t^*	$\tilde{t}_{r+3}$	$\dots$	$\tilde{t}_{n+k+2}$
t_0	$\dots$	t_r	t_r	t_r	t_r	t_{r+1}	$\dots$	t_{n+k}
τ_0	$\dots$	τ_r	τ_{r+1}	τ_{r+2}	τ_{r+3}	τ_{r+4}	$\dots$	τ_{n+k+3}

Now let us examine our results a little closer. We have now discovered that only three new control points is actually computed: P^1_{r-2}, $\hat{P}^1_{r-1}$ and P^1_{r-1}, which are all distinct.

Notice that $\hat{P}^1_{r-1} = P^2_{r-1}$. This means the new control points actually can be computed directly by using the de Boor algorithm (see appendix A) on P_{r-3},P_{r-2} and P_{r-1}.

$$\begin{tikzcd}
P^0_{r-3} \arrow[dr] & & \\
P^0_{r-2} \arrow[r] \arrow[dr] & P^1_{r-2} \arrow[dr] & \\
P^0_{r-1} \arrow[r] & P^1_{r-1} \arrow[r] & \hat{P}^2_{r-1}
\end{tikzcd}$$