Computational Algorithms

Let $U = \{ u_0, ..., u_m \}$ be a knot vector and suppose we are interested in the basis functions of degree p.

In addition, suppose u is fixed, and $u \in [u_i, u_{i+1})$. We will develop algorithms to compute the following:

1) the knot span index i

2) $N_{i-p,p}(u), ..., N_{i,p}(u)$

3) $N^{(k)}_{i-p,p}(u)$, ..., $N^{(k)}_{i,p}(u)$ for $k = 0, ..., p$.
For $k > p$, the derivatives are zero.

4) a single basis function, $N_{j,p}(u)$, where
$0 \le j \le m - p - 1$.

5) the derivatives of a single basis function,
$N^{(k)}_{j,p}(u)$, where, $0 \le j \le m - p - 1$, and
$k = 0, ..., p$.

From P2.2 and the assumption that $u \in [u_i, u_{i+1})$, it follows that we may focus our attention on $N_{i-p,p}(u)$, ..., $N_{i,p}(u)$ and their derivatives; all of the other functions are identically zero, and it would be wasteful to actually compute them.

The first step is to determine the knot space in which u lies. We will use a binary search.

Note, a subtle problem in the evaluation of the basis functions can result in the case where $u = u_m$. It is best to handle this case by setting the span index to n (= m - p - 1). Hence, in this case,

$u \in (u_{m-p-1}, u_{m-p}]$

See algorithm A2.1

Now Consider the second algorithm. You should note that direct implementation of the Cox-deBoor formulation leads to many redundant computations.

For example, consider the degree 2 basis functions in general terms:

Note that:

$$N_{i-2,2}(u) = \frac{u - u_{i-2}}{u_i - u_{i-2}} N_{i-2,1}(u) + \frac{u_{i+1} - u}{u_{i+1} - u_{i-1}} N_{i-1,1}(u) \quad \text{(a)}$$

$$N_{i-1,2}(u) = \frac{u - u_{i-1}}{u_{i+1} - u_{i-1}} N_{i-1,1}(u) + \frac{u_{i+2} - u}{u_{i+2} - u_i} N_{i,1}(u) \quad \text{(b)}$$

$$N_{i,2}(u) = \frac{u - u_i}{u_{i+2} - u_i} N_{i,1}(u) + \frac{u_{i+3} - u}{u_{i+3} - u_{i+1}} N_{i+1,1}(u) \qquad \text{(c)}$$

- the first term of equ. (a) and the last term of equ. (c) should not be computed, since $N_{i-2,1}(u) = N_{i+1,1}(u) = 0$

- the expression,

$$\frac{N_{i-1,1}(u)}{u_{i+1} - u_{i-1}}$$

which appears in the second term of equ. (a), also appears in the first term first term of equ. (b). A similar statement holds for the second term of equ. (b) and the first term of equ. (c).

Thus, we introduce the notation,

$$left[j] = u - u_{i+1-j}$$

$$right[j] = u_{i+j} - u$$

and rewrite equations (a)-(c) as follows:

$$N_{i-2,2}(u) = \frac{left[3]}{right[0] + left[3]} N_{i-2,1}(u) + \frac{right[1]}{right[1] + left[2]} N_{i-1,1}(u) \quad \text{(a)}$$

$$N_{i-1,2}(u) = \frac{left[2]}{right[1] + left[2]} N_{i-1,1}(u) + \frac{right[2]}{right[2] + left[1]} N_{i,1}(u) \quad \text{(b)}$$

$$N_{i,2}(u) = \frac{left[1]}{right[2] + left[1]} N_{i,1}(u) + \frac{right[3]}{right[3] + left[0]} N_{i+1,1}(u) \qquad \text{(c)}$$

Based on these observations, the following algorithm computes all the non-vanishing basis functions and stores them in the array N**[0], ..., **N**[**p**]**

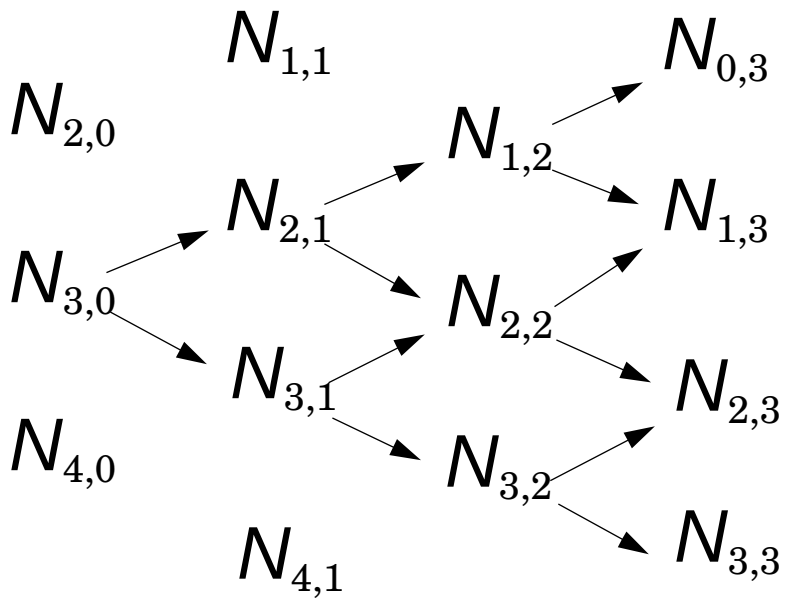
See algorithm A2.2

Note: algorithm A2.2 avoids zero-divides!

B-spline Basis Function Derivative Algorithm

The algorithm will be based on the second general formulation for the k-th derivative of $N_{r,p}(u)$, for $i - p \leq r \leq i$ and $0 \leq k \leq p$. The basic ingredients are:

- the inverted triangle of non-zero basis functions computed as in Algorithm A2.2, e.g.,



- differences of knots, also computed in A2.2 (the sums: **`right[r+1]+left[j-r]`**).

- differences of the $a_{k,j}$. Note that the $a_{k,j}$ depend on the $a_{k-1,j}$, but not the $a_{s,j}$, for $s < k - 1$.

Viewed in a 2D array, of dimension (p + 1) x (p + 1), the basis functions fit into the upper triangle (including the diagonal), and the knot differences fit into the lower triangle.
For example,

$N_{i,0}(u)$	$N_{i-1,1}(u)$	$N_{i-2,2}(u)$
$u_{i+1} - u_i$	$N_{i,1}(u)$	$N_{i-1,2}(u)$
$u_{i+1} - u_{i-1}$	$u_{i+2} - u_i$	$N_{i,2}(u)$

Example:

Let $p = 2$, $U = \{0, 0, 0, 1, 2, 3, 4, 4, 5, 5, 5\}$, and $u = 2.5$. Then $u \in [u_4, u_5)$, and the table becomes:

$N_{4,0}(2.5) = 1$	$N_{3,1}(2.5) = 1/2$	$N_{2,2}(2.5) = 1/8$
$u_5 - u_4 = 1$	$N_{4,1}(2.5) = 1/2$	$N_{3,2}(2.5) = 6/8$
$u_5 - u_3 = 2$	$u_6 - u_4 = 2$	$N_{4,2}(2.5) = 1/8$

Next, compute $N^{(1)}_{4,2}(2.5)$ and $N^{(2)}_{4,2}(2.5)$.
With i = 4 the formula yields:

$$a_{1,0} = \frac{1}{u_6 - u_4} = \frac{1}{2}$$

$$a_{1,1} = \frac{1}{u_7 - u_5} = -1$$

$$a_{2,0} = \frac{a_{1,0}}{u_5 - u_4} = \frac{1}{2}$$

$$a_{2,1} = \frac{a_{1,1} - a_{1,0}}{u_6 - u_5} = \frac{-1 - \frac{1}{2}}{4 - 3} = \frac{-3}{2}$$

$$a_{2,2} = \frac{a_{1,1}}{u_7 - u_6} = \frac{1}{4 - 4} = \frac{1}{0}$$

and,

$$N^{(1)}_{4,2}(u) = 2\,(a_{1,0}N_{4,1}(2.5) + a_{1,1}N_{5,1}(2.5))$$

$$N^{(2)}_{4,2}(u) = 2\,(a_{2,0}N_{4,0}(2.5) + a_{2,1}N_{5,0}(2.5) + a_{2,2}N_{6,0}(2.5))$$

Now, $a_{1,1}$, $a_{2,1}$ and $a_{2,2}$ all use knot differences which are not in the table, but they are multiplied by $N_{5,1}(2.5)$, $N_{5,0}(2.5)$ and $N_{6,0}(2.5)$ respectively, which are also not in the table (they are equal to zero). Thus the equations reduce to:

$$N^{(1)}_{4,2}(u) = 2a_{1,0}N_{4,1}(2.5) = \frac{1}{2}$$

$$N^{(2)}_{4,2}(u) = 2a_{2,0}N_{4,0}(2.5) = 1$$

These values can be checked by verifying that $N_{4,2}(u) = 0.5(u - 2)^2$ on $u \in [2, 3)$

Thus, Algorithm A2.3 is formulated based on observations drawn from this example.

It computes the non-zero basis functions and their derivatives, up to and including the n-th derivative ($n \leq p$). Output is in the 2D array, **`ders`**. **`ders[k][j]`** is the k-th derivative of the function $N_{i-p+j,p}$, where $0 \leq j \leq p$ and $0 \leq k \leq n$. Two local arrays are used:

- **`ndu[p+1][p+1]`** : as in the above table
- **`a[2][p+1]`** : to store (in an alternating fashion) the two most recently computed rows $a_{k,j}$ and $a_{k\text{-}1,j}$.

Note that the algorithm avoids divides by zero and/or the use of terms not in the table, **`ndu[][]`**