Derivatives of B-spline curves

Let $C^{(k)}(u)$ denote the $k$-th derivative of $C(u)$. If $u$ is fixed, we can obtain $C^{(k)}(u)$ by computing the $k$-th derivative of the basis functions,

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

\[\text{ME525x NURBS Curve and Surface Modeling Page 195}\]
An algorithm to compute the point on a B-spline curve and all derivatives up to and including the $d$-th, at a fixed $u$-value follows.

We allow $d > p$, although the derivatives are zero in this case (for nonrational curves); these derivatives are necessary for rational curves.
Input to the algorithm is $u$, $d$, and the B-spline curve, defined (throughout the course) by:

- $n$: the number of control points is $n + 1$
- $p$: the degree of the curve
- $U$: the knots
- $P$: the control points
Output is the array, \texttt{ck[]}, where \texttt{ck[]} is the \( k \)-th derivative, \( 0 \leq k \leq d \). A local array, \texttt{nders[][]}, is used to store the derivatives of the basis functions.

See algorithm A3.2
Now, instead of fixing $u$, we want to formally differentiate the $p$-th degree B-spline curve defined on the knot vector,

$$U = \{0, \ldots, 0, u_{p+1}, \ldots u_{m-p-1}, 1, \ldots, 1\}$$

Thus,
\[ C'(u) = \sum_{i=0}^{n} N'_{i,p}(u) P_i \]

\[
= \sum_{i=0}^{n} \left( \frac{p}{u_i + p - u_i} N_{i,p-1}(u) \right. \\
- \left. \frac{p}{u_i + p + 1 - u_{i+1}} N_{i+1,p-1}(u) \right) P_i
\]
\[ C'(u) = \left( p \sum_{i=-1}^{n-1} N_{i+1,p-1}(u) \frac{P_{i+1}}{u_{i+p+1} - u_{i+1}} \right) \]

or,

\[ -\left( p \sum_{i=0}^{n} N_{i+1,p-1}(u) \frac{P_i}{u_{i+p+1} - u_{i+1}} \right) \]
\[
C'(u) = p \frac{N_{0,p-1}(u)P_0}{u_p - u_0} \\
+ p \sum_{i=0}^{n-1} N_{i+1,p-1}(u) \frac{(P_{i+1} - P_i)}{u_{i+p+1} - u_{i+1}} \\
- p \frac{N_{n+1,p-1}(u)P_n}{u_{n+p+1} - u_{n+1}}
\]

Note that the first and last terms evaluate to 0/0, which is 0 by definition. Thus:
\[
C'(u) = p \sum_{i=0}^{n-1} N_{i+1,p-1}(u) \frac{(P_{i+1} - P_i)}{u_i + p + 1 - u_{i+1}}
\]

where,

\[
Q_i = p \frac{(P_{i+1} - P_i)}{u_i + p + 1 - u_{i+1}}
\]
Now, let $U'$ be the knot vector obtained by dropping the first and last knots from $U$, i.e.,

$$U' = \{0, \ldots, 0, u_{p+1}, \ldots u_{m-p-1}, 1, \ldots, 1\}$$

($U'$ has $m-1$ knots). Then it is easy to check that the function $N_{i+1,p-1}(u)$, computed on $U$, is equal to $N_{i,p-1}(u)$, computed on $U'$. Thus,
where the $N_{i,p-1}(u)$ are computed on $U'$. Hence, $C'(u)$ is a $(p - 1)$-th degree B-spline curve.
Example:

Let \( C(u) = \sum_{i=0}^{4} N_{i,2}(u)P_i \), defined on 
\( U = \{0, 0, 0, 2/5, 3/5, 1, 1, 1\} \).
Then, \( U' = \{0, 0, 2/5, 3/5, 1, 1, 1\} \), and

\[ C'(u) = \sum_{i=0}^{3} N_{i,1}(u)Q_i \]
where,

\[ Q_0 = \frac{2(\mathbf{P}_1 - \mathbf{P}_0)}{\frac{1}{3} - 0} = 6(\mathbf{P}_1 - \mathbf{P}_0) \]

\[ Q_1 = \frac{2(\mathbf{P}_2 - \mathbf{P}_1)}{\frac{3}{4} - 0} = \frac{8}{3}(\mathbf{P}_2 - \mathbf{P}_1) \]

and,
\[ Q_2 = \frac{2 (P_3 - P_2)}{1 - \frac{1}{3}} = 3 (P_3 - P_2) \]

\[ Q_3 = \frac{2 (P_4 - P_3)}{1 - \frac{3}{4}} = 8 (P_4 - P_3) \]
Since $C'(u)$ is a B-spline curve, we can apply this formulation recursively to obtain higher order derivative. Letting $P_i^{(0)} = P_i$, we write:

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

\[ C^{(k)}(u) = \sum_{i=0}^{n-k} N_{i,p-k}(u) P_{i}^{(k)} \]
with,

\[ P_i^{(k)} = \begin{cases} 
    P_i, & \text{if } k = 0 \\
    \frac{p - k + 1}{u_{i+p+1} - u_{i+k}} \left( P_{i+1}^{k-1} + P_i^{k-1} \right), & \text{if } k > 0
\end{cases} \]
and,

\[ U^{(k)} = \{0, \ldots, 0, u_{p+1}, \ldots u_{m-p-1}, 1, \ldots, 1\} \]

\[ \quad \begin{array}{c}
p - k + 1 \\
p - k + 1 \\
\end{array} \]
An algorithm based on this formulation computes the control points of all derivative curves up to and including the $d$-th derivative.

On output, $PK[k][j]$ is the $j$-th control point of the $k$-th derivative curve, where $0 \leq k \leq d$ and $r_1 \leq j \leq r_2 - k$. If $r_1 = 0$ and $r_2 = n$, all control points are computed.

See algorithm A3.3
This formulation can be used to develop another algorithm to compute the point on a B-spline curve and all of its derivatives up to and including the $d$-th at a fixed $u$-value.

This algorithm is based on Algorithm A3.3, and assumes a simple modification of Algorithm A2.2 (BasisFuns) to return all nonzero basis functions of all degrees (from 0 up to $p$).
In particular, $N[j][i]$ is the value of the $i$-th degree basis function $N_{span-i+j,i}(u)$, where $0 \leq i \leq p$ and $0 \leq j \leq i$

See algorithm A3.4