

EECS 598-1  
Programming Assignment 2: B-splines  
due 2/22/2002 at 11:59pm

February 19, 2002

In this assignment you will be implementing two methods of evaluating cubic B-splines.

For every evaluation you will have  $n + 3$  control points  $p_0, \dots, p_{n+2}$ , and a sequence of knot values  $U = \{u_k\}_{k=0}^{n+6}$ . We set  $u_0 = u_1 = u_2 = u_3 = a$  and  $u_{n+3} = u_{n+4} = u_{n+5} = u_{n+6} = b$ , so that the curve's parameter belongs to the segment  $[a, b]$ . Note that the control point  $p_k$  is associated with the bag  $(u_{k+1}, u_{k+2}, u_{k+3})$ .

First of all, you will need to convert each polynomial piece into the Bezier form. There will be  $n$  Bezier segments after this operation,

$$F^k(u) = \sum_{s=0}^3 \beta_s^k B_s^3 \left( \frac{u - u_{k+2}}{u_{k+3} - u_{k+2}} \right), \text{ for } u \in [u_{k+2}, u_{k+3}] \text{ and } k = 1, \dots, n.$$

As you may recall  $B_s^3(t) = \binom{3}{s} u^s (1 - u)^{3-s}$  are Bernstein polynomials.

In order to do the conversion you will need to figure out the relation between the B-spline control points  $p_k$  and Bezier control points  $\beta_s^k$ . In order to do that consider four consecutive control points  $p_{k-1}, p_k, p_{k+1}, p_{k+2}$ : these are the values of the blossom  $f^k$  of  $F^k$ . To be precise,

$$\begin{aligned} p_{k-1} &= f^k(u_k, u_{k+1}, u_{k+2}), \\ p_k &= f^k(u_{k+1}, u_{k+2}, u_{k+3}), \\ p_{k+1} &= f^k(u_{k+2}, u_{k+3}, u_{k+4}), \\ p_{k+2} &= f^k(u_{k+3}, u_{k+4}, u_{k+5}), \end{aligned}$$

while the corresponding Bezier control points are given via

$$\begin{aligned}\beta_k^0 &= f^k(u_{k+2}, u_{k+2}, u_{k+2}), \\ \beta_k^1 &= f^k(u_{k+2}, u_{k+2}, u_{k+3}), \\ \beta_k^2 &= f^k(u_{k+2}, u_{k+3}, u_{k+3}), \\ \beta_k^3 &= f^k(u_{k+3}, u_{k+3}, u_{k+3}).\end{aligned}$$

You should be able to figure out how to obtain  $\beta_k^s$  from the corresponding four points  $p_{k-1}, p_k, p_{k+1}, p_{k+2}$ .

Once you have the Bezier control points for each segment  $k$  your evaluation of a B-spline curve will be done by drawing  $n$  Bezier segments. You will need to implement two ways of evaluating each Bezier segment:

1. Horner's rule;
2. adaptive forward differences.

For both of these evaluation methods you will first convert the Bezier representation into a polynomial form, that is you will need to express each  $F^k(u) = A_0^k + A_1^k(t_k(u)) + A_2^k(t_k(u))^2 + A_3^k(t_k(u))^3$  with  $t_k(u) = (u - u_{k+2}) / (u_{k+3} - u_{k+2})$ .

After that is done, continue to the next step:

- **Horner's rule:** Substitute  $t = 0, \tau, 2\tau, \dots, 1$  ( $\tau = 1/T$  for some positive integer  $T$ , for instance  $T = 20$ ) into and evaluate the expression  $A_0^k + t(A_1^k + t(A_2^k + tA_3^k))$ . Fill the output vector with the resulting values.
- **Adaptive Forward Differences:** You will need to implement the ADF method described in Lien, Shantz and Pratt's paper from Siggraph 1987. In short, you will first convert each Bezier segment to the particular polynomial basis they are using. Then, you will start the adaptive algorithm as described in the paper, filling the output vector with the resulting values.

You should make sure that your code works with different curve parameterizations: switch between the parameterization modes by pressing keys "u" – uniform, "n" – non-uniform, "c" – centripetal. Press "k" to see the knot sequence.