Mathematics 308—Fall 1996

Third homework—due Monday, October 21

1. Write a PostScript procedure **pixelcurve** with arguments 4 arrays P_0, P_1, P_2, P_3 of size 2, with the effect of drawing the corresponding Bezier curve, including also black pixels of width $0.05''$ at each of these points.

2. In the figure in the notes on how computers draw Bezier curves, the point $P_{1/6}$ is $(1/2)P_0 + (1/2)P_{1/3}$. Find similar expressions for all the constructed points in terms of the original four.

3. The purpose of this exercise is to prove the assertion about Bezier curves (as drawn by the bisection algorithm) and cubic polynomials. Let

$$P(s) = (1-s)^3 y_0 + 3s(1-s)^2 y_{1/3} + 3s^2(1-s) y_{2/3} + s^3 y_1$$

The point is to verify that this formula agrees with the geometrical process described earlier. Let $P_{1/6}$ etc. be the points defined in section 2. (1) In the previous exercise you (should have) found that

$$P_{1/2} = \frac{P_0 + 2P_1 + 2P_2 + P_3}{6} .$$

Verify that $P(1/2) = P_{1/2}$. (2) The rest of that construction relied on the claim that the curve from P_0 to $P_{1/2}$ was itself a Bezier curve with control points $P_{1/6}$ and $P_{2/6}$. Now the first half of the parametrized curve has a normalized parametrization $s \mapsto P(s/2)$ as s goes from 0 to 1. Verify that

$$P(s/2) = (1-s)^3 P_0 + 3s(1-s)^2 P_{1/6} + 3s^2(1-s) P_{2/6} + s^3 P_{1/2}$$

by using the expressions for $P_{1/6}$ and $P_{2/6}$ in terms of P_0, P_1, P_2, P_3.

4. Write a PostScript procedure **polynomial** that you can fit into **mkpath** or **mkgraph** (the version that accepts an array of parameters) that will graph a polynomial between x_0 and x_1 with N Bezier segments. You will use it like this:

```
[2 3 1] /polynomial -2 2 4 mkgraph
```

to draw the graph of $2x^2 + 3x + 1$ between $x = -2$ and $x = 2$, using 4 Bezier segments. As a beginning, you should start by drawing, say, quartic polynomials, then move on to the more difficult problem of variable degree.

There are a couple of of things you might think about: (1) For evaluating a polynomial in a program it is easiest to use an expression like $5x^3 + 2x + 3x + 4 = ((5x+2)x+3)x+4$. This is called Horner's method for evaluation of polynomials. (2) You will have to add an argument to this procedure to pass the polynomial coefficients as an array. Recall that **length** returns the size of an array.

5. Write a PostScript program to draw the Lissajous figure with parametrization $t \mapsto (\cos 2t, \cos 3t)$.

Below I exhibit all of **mkgraph.inc**.

```
% mkgraph.inc

% [...]  /f x0 x1 N

/mkgraph {
8 dict begin
/N exch def
/x1 exch def
/x0 exch def
/f exch cvx def
```

```
/pars exch deff

% h = (x1 - x0)/N
/h x1 x0 sub N div def

/x x0 def
/F pars x f def
/y F 0 get def
/s F 1 get def

x y moveto

N {
    x 0.33333 h mul add
    y h 0.33333 mul s mul add
    /x x h add def
    /F x pars f def
    /y F 0 get def
    /s F 1 get def
    x h 0.33333 mul sub
y h 0.33333 mul s mul sub
x y
curveto
} repeat

end
} def

% a sample:  cx^4

% [cx^4 4cx^3]

/quartic {
    4 dict begin
    /x exch def
/pars exch def
/c pars 0 get def
[
x x mul x mul x mul c mul
x x mul x mul 4 mul c mul
]
end
} def
```