Math128aHw6

Trustin Nguyen

October 18, 2024

Exercise Set 3.6

Exercise 2: Repeat Exercise 1 using cubic Bezier polynomials: Let $(x_0, y_0) = (0, 0)$ and $(x_1, y_1) = (5, 2)$ be the endpoints of a curve. Use the given guidepoints to construct parametric Bezier approximations $(x(t), y(t))$ to the curve and graph the approximations.

(c) $(1, 1)$ and $(6, 3)$.

Answer. We first calculate α_0, β_0, α_1, β_1. We have

$$(1, 1) - (0, 0) = (1, 1) = \alpha_0, \beta_0$$

and

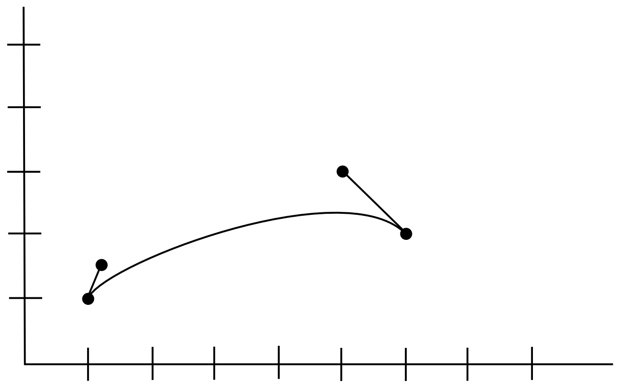
$$(6, 3) - (5, 2) = (1, 1) = -\alpha_1, -\beta_1$$

We also have $x_0, y_0 = (1, 1)$ and $x_1, y_1 = (5, 2)$. Then by the formula, the parametric Bezier is

$$\begin{aligned} x(t) &= 1 + t + (3 \cdot 4 - 3(-1 + 2))t^2 + (3(1 + -1) - 2(1 - 5))t^3 \\ &= 1 + t + 9t^2 + 8t^3 \\ y(t) &= 1 + 1(t) + (3(2 - 0) - 3(1 + 2(-1)))t^2 + (3(1 + (-1)) - 2(2 - 3))t^3 \\ &= 1 + t + 12t^2 + 2t^3 \end{aligned}$$

Exercise 3: Construct and graph the cubic Bezier polynomials given the following points and guidepoints.

(b) Point $(1, 1)$ with guidepoint $(1.25, 1.5)$ to point $(6, 2)$ with guidepoint $(5, 3)$

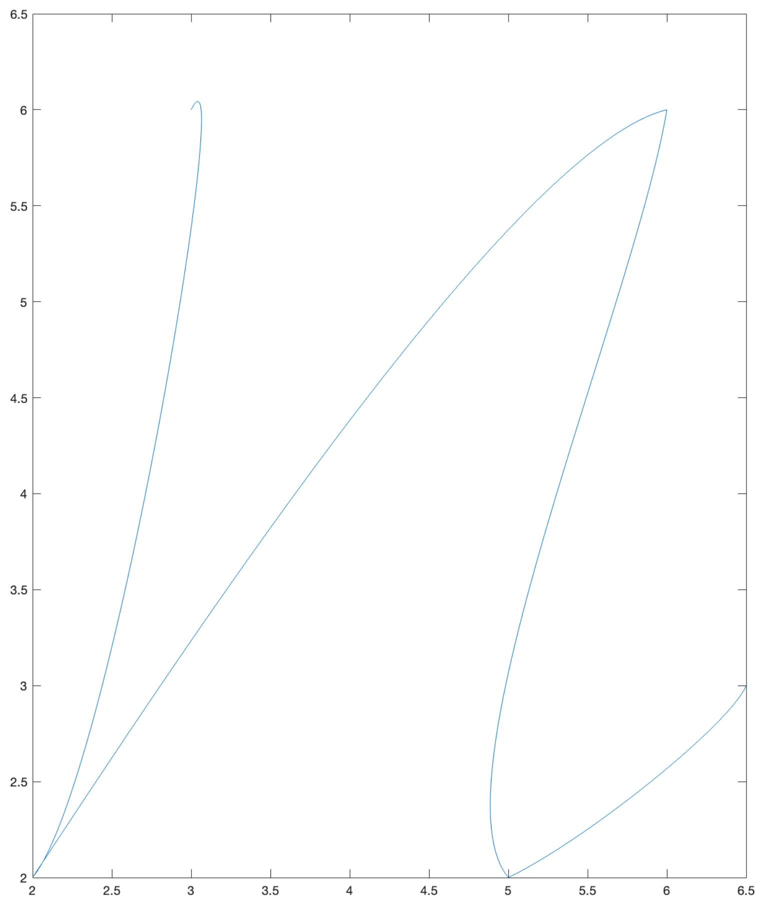


Exercise 4: Use the data in the following table and Algorithm 3.6 to approximate the shape of the letter N

i	x_i	y_i	α_i	β_i	α'_i	β'_i
0	3	6	3.3	6.5		
1	2	2	2.8	3.0	2.5	2.5
2	6	6	5.8	5.0	5.0	5.8
3	5	2	5.5	2.2	4.5	2.5
4	6.5	3			6.4	2.8

```
function out = Bezier(ep0, ep1, gp0, gp1)
x0 = ep0(1); y0 = ep0(2); x1 = ep1(1); y1 = ep1(2);
xp = gp0(1); yp = gp0(2); xn = gp1(1); yn = gp1(2);
out.x = [x0, 3 * (xp - x0), 3 * (x0 + xn - 2 * xp), x1 - x0 + 3 * (xp - xn)];
out.y = [y0, 3 * (yp - y0), 3 * (y0 + yn - 2 * yp), y1 - y0 + 3 * (yp - yn)];
end

function out = BezierCurves( points, lGP, rGP)
out.x = [];
out.y = [];
for i = 1:size(points, 1) - 1
    coeff = Bezier(points(i, :), points(i + 1, :), lGP(i, :), rGP(i, :));
    out.x = [out.x; coeff.x];
    out.y = [out.y; coeff.y];
end
end

function polys = BezierPolys( xCoeff, yCoeff )
syms s;
polys.x = [];
polys.y = [];
for i=1:size(xCoeff, 1)
    xc = xCoeff(i, :);
    polys.x = [polys.x; xc(1) + xc(2) * s + xc(3) * s^2 + xc(4) * s^3];
    yc = yCoeff(i, :);
    polys.y = [polys.y; yc(1) + yc(2) * s + yc(3) * s^2 + yc(4) * s^3];
```

```
end
end

points = [[3, 6]; [2, 2]; [6, 6]; [5, 2]; [6.5, 3]];
lGP = [[3.3, 6.5]; [2.8, 3.0]; [5.8, 5.0]; [5.5, 2.2]];
rGP = [[2.5, 2.5]; [5.0, 5.8]; [4.5, 2.5]; [6.4, 2.8]];

coeff = BezierCurves(points, lGP, rGP);
polys = BezierPolys(coeff.x, coeff.y);

x = [];
y = [];
t = 0:0.01:1;
for i=1:size(points, 1) - 1
    xPoly = matlabFunction(polys.x(i));
    x = [x, xPoly(t)];
    yPoly = matlabFunction(polys.y(i));
    y = [y, yPoly(t)];
end
plot(x, y)
```



Exercise 5: Suppose a cubic Bezier polynomial is placed through (u_0, v_0) and (u_3, v_3) with guidepoints (u_1, v_1) and (u_2, v_2), respectively.

(a) Derive the parametric equations for $u(t)$ and $v(t)$ assuming that

$$u(0) = u_0,\ u(1) = u_3,\ u'(0) = u_1 - u_0,\ u'(1) = u_3 - u_2$$

and

$$v(0) = v_0,\ v(1) = v_3,\ v'(0) = v_1 - v_0,\ v'(1) = v_3 - v_2$$

Answer. We can use the formula for the parametric Bezier polynomial:

$$u(t) = [2(u_0 - u_3) + 3(u'(0) - u'(1))]t^3 + [3(u_3 - u_0) - 3(-u'(1) + 2u'(0))]t^2 + u'(0)t + u_0$$
$$v(t) = [2(v_0 - v_3) + 3(v'(0) - v'(1))]t^3 + [3(v_3 - v_0) - 3(-v'(1) + 2v'(0))]t^2 + v'(0)t + v_0$$

Then substitute in the rest, and that is our answer.

Exercise Set 4.1

Exercise 2: Use the forward-difference formulas and backward-difference formulas to determine each missing entry in the following tables.

(b)

x	$f(x)$	$f'(x)$
1.0	1.0000	
1.2	1.2625	
1.4	1.6595	

Answer. The formula is given by:

$$f'(x) = \frac{f(x+h) - f(x)}{h} - \frac{h}{2}f''(\mathcal{E})$$

Then:

$$f'(1.0) \approx \frac{f(1.0 + .2) - f(1.0)}{.2} = \frac{1.2625 - 1.0000}{.2} = 0.2625/0.2 = 1.3125$$

and

$$f'(1.2) \approx \frac{f(1.2 + .2) - f(1.2)}{.2} = \frac{1.6595 - 1.2625}{.2} = 1.985$$

For the last one, we need to use the backward difference:

$$f'(1.4) \approx \frac{f(1.4 - .2) - f(1.4)}{-.2} = \frac{1.2625 - 1.6595}{-.2} = 1.985$$

So these are the table entries.

Exercise 4: The data in Exercise 2 were taken from the following functions. Compute the actual errors in Exercise 2 and find error bounds using the error formulas.

(b) $f(x) = x^2 \ln x + 1$

Answer. I got

x	$f'(x)$
1.0	1
1.2	1.63757173631
1.4	1.98214708762

So for the errors:

$$\left|\frac{1 - 1.3125}{1.3125}\right| = 0.23809523809524$$

$$\left|\frac{1.985 - 1.63757173631}{1.985}\right| = 0.1750268330932$$

$$\left|\frac{1.985 - 1.98214708762}{1.985}\right| = 0.0014372354559194$$

The error is bounded by

$$\left|\frac{f''(\mathcal{E})}{2}\right|$$

where $\mathcal{E} \in [x_0, x_0 + h]$. The second derivative is given as

$$2\ln x + 3$$

Since $\ln x$ is increasing, we just need to calculate it at the left endpoint for our approximation interval. Here are the error bounds:

$$\left|\frac{2\ln 1.2 + 3}{2}\right| = 1.68232155679$$
$$\left|\frac{2\ln 1.4 + 3}{2}\right| = 1.83647223662$$
$$\left|\frac{2\ln 1.4 + 3}{2}\right| = 1.83647223662$$

Exercise 6: Use the most accurate three-point formula to determine each missing entry in the following tables.

(d)

x	$f(x)$	$f'(x)$
-2.7	0.054797	
-2.5	0.11342	
-2.3	0.65536	
-2.1	0.98472	

Answer. We will use the three-point endpoint formula for $-2.7, -2.1$ and the three-point midpoint formula for $-2.5, -2.3$. We have the formulas as:

$$\begin{aligned} f'(x_0) &= \frac{1}{2h}[-3f(x_0) + 4f(x_0+h) - f(x_0+2h)] + \frac{h^2}{3}f^{(3)}(\mathcal{E}_0) \\ f'(x_0) &= \frac{1}{2h}[f(x_0+h) - f(x_0-h)] - \frac{h^2}{6}f^{(3)}(\mathcal{E}_0) \end{aligned}$$

Then for -2.7:

$$\begin{aligned} f'(-2.7) &= \frac{1}{2\cdot .2}[-3(0.054797) + 4(0.11342) - 0.65536] \\ &= \frac{1}{.4}(-0.164391 + 0.45368 - 0.65536) \\ &= 2.5 * (-0.164391 + 0.45368 - 0.65536) \\ &= -0.9151775 \end{aligned}$$

for -2.5:

$$\begin{aligned} f'(-2.5) &= \frac{1}{2\cdot .2}[0.65536 - 0.054797] \\ &= \frac{1}{.4}(0.600563) \\ &= 2.5 * 0.600563 \\ &= 0.9151775 \end{aligned}$$

for -2.3:

$$\begin{aligned} f'(-2.3) &= \frac{1}{2\cdot .2}[0.98472 - 0.11342] \\ &= \frac{1}{.4}(0.8713) \\ &= 2.5 * 0.8713 \\ &= 2.17825 \end{aligned}$$

and finally for −2.1:

$$\begin{aligned} f'(-2.1) &= \frac{1}{2\cdot -.2}[-3(0.98472) + 4(0.65536) - 0.11342] \\ &= \frac{-1}{.4}[-2.95416 + 2.62144 - 0.11342] \\ &= -2.5(-0.44614) \\ &= 1.11535 \end{aligned}$$

Exercise 8: The data in Exercise 6 were taken from the following functions. Compute the actual errors in Exercise 6 and find error bounds using the error formulas.

(d) $f(x) = (\cos 3x)^2 - e^{2x}$.

Answer. The derivatives that I got are:

x	$f'(x)$
−2.7	−1.42629912108
−2.5	1.93738762647
−2.3	2.81098333684
−2.1	0.0708779880225

So calculating the errors:

$$\left|\frac{-0.9151775 + 1.42629912108}{-0.9151775}\right| = 0.55849452273466$$

$$\left|\frac{0.9151775 - 1.93738762647}{0.9151775}\right| = 1.116952860467$$

$$\left|\frac{2.17825 - 2.81098333684}{2.17825}\right| = 0.29047783167221$$

$$\left|\frac{1.11535 - 0.0708779880225}{1.11535}\right| = 0.94096577655631$$

The error is bounded by

$$\frac{h^2}{3}f^{(3)}(\mathcal{E})$$

for $\mathcal{E}$ in the interval $[x_0, x_0+2h]$ or $[x_0-h, x_0+h]$ depending on the approximation method.

Exercise 24: Derive an $\mathcal{O}(h^4)$ five-point formula to approximate $f'(x_0)$ that uses $f(x_0-h), f(x_0), f(x_0+h), f(x_0+2h)$, and $f(x_0+3h)$.

Answer.

Exercise 28: Derive a method for approximating $f'''(x_0)$ whose error term is of order h^2 by expanding the function f in a fourth Taylor polynomial about x_0 and evaluating at $x_0 \pm h$ and $x_0 \pm 2h$.

Exercise Set 4.2

Exercise 1: Apply the extrapolation process described in Example 1 to determine $N_3(h)$, an approximation to $f'(x_0)$, for the following functions and step sizes.

(b) $f(x) = x + e^x, x_0 = 0.0, h = 0.4$

Answer. I got $N_3(h) = 2.0554$. Here is my code:

```
function y = ThreePointEndpoint(f, x, h)
y = (-3 * f(x) + 4 * f(x + h) - f(x + 2 * h)) / (2 * h);
end

function y = ThreePointMidpoint(f, x, h)
y = (f(x + h) - f(x - h)) / (2 * h);
end

function out = approxDerive(f, x, h, n)
out = zeros(n, 1);
out(1) = ThreePointEndpoint(f, x, h);
x = x + h;
for i=2:n-1
    out(i) = ThreePointMidpoint(f, x, h);
    x = x + h;
end
out(end) = ThreePointEndpoint(f, x, -h);
end

function res = Extrapolate(initial)
sz = size(initial, 1);
res = zeros(sz, sz);
res(:, 1) = initial;
for i = 2:sz
    for j = 2:i
        res(i, j) = res(i, j - 1) + (res(i, j - 1) - res(i - 1, j - 1)) / (4^(j - 1) -
    end
end
end

syms s;

f(s) = s + exp(s);
cols = 3;
h = 0.4;
numIter = 2^(cols - 1);
approx = approxDerive(f, h, -h / numIter, numIter);
indices = zeros(cols, 1);
indices(end) = numIter;
for i=2:cols
    numIter = numIter - 2^(i - 2);
    indices(end-i+1) = numIter;
end
approx = approx(indices);

ex = Extrapolate(approx)
```

Exercise 2: Add another line to the extrapolation table in Exercise 1 to obtain the approximation $N_4(h)$.

(b) $f(x) = x + e^x, x_0 = 0.0, h = 0.4$

Answer. I got $N_4(h) = 2.0291$. Here is my code:

```
function y = ThreePointEndpoint(f, x, h)
y = (-3 * f(x) + 4 * f(x + h) - f(x + 2 * h)) / (2 * h);
end

function y = ThreePointMidpoint(f, x, h)
y = (f(x + h) - f(x - h)) / (2 * h);
end

function out = approxDerive(f, x, h, n)
out = zeros(n, 1);
out(1) = ThreePointEndpoint(f, x, h);
x = x + h;
for i=2:n-1
    out(i) = ThreePointMidpoint(f, x, h);
    x = x + h;
end
out(end) = ThreePointEndpoint(f, x, -h);
end

function res = Extrapolate(initial)
sz = size(initial, 1);
res = zeros(sz, sz);
res(:, 1) = initial;
for i = 2:sz
    for j = 2:i
        res(i, j) = res(i, j - 1) + (res(i, j - 1) - res(i - 1, j - 1)) / (4^(j - 1) -
    end
end
end

syms s;

f(s) = s + exp(s);
cols = 4;
h = 0.4;
numIter = 2^(cols - 1);
approx = approxDerive(f, h, -h / numIter, numIter);
indices = zeros(cols, 1);
indices(end) = numIter;
for i=2:cols
    numIter = numIter - 2^(i - 2);
    indices(end-i+1) = numIter;
end
approx = approx(indices);

ex = Extrapolate(approx)
```

Exercise 5: The following data give approximations to the integral

$$M = \int_0^{\pi} \sin x \, dx$$

$$N_1(h) = 1.570796,\ N_1\left(\frac{h}{2}\right) = 1.896119,\ N_1\left(\frac{h}{4}\right) = 1.974232,\ n_1\left(\frac{h}{8}\right) = 1.993570$$

Assuming $M = N_1(h) + K_1h^2 + K_2h^4 + K_3h^6 + K_4h^8 + \mathcal{O}(h^{10})$, construct an extrapolation table to determine $N_4(h)$.

Answer. I got $N_4(h) = 2.0000$. Here is my code:

```
function y = ThreePointEndpoint(f, x, h)
y = (-3 * f(x) + 4 * f(x + h) - f(x + 2 * h)) / (2 * h);
end

function y = ThreePointMidpoint(f, x, h)
y = (f(x + h) - f(x - h)) / (2 * h);
end

function out = approxDerive(f, x, h, n)
out = zeros(n, 1);
out(1) = ThreePointEndpoint(f, x, h);
x = x + h;
for i=2:n-1
    out(i) = ThreePointMidpoint(f, x, h);
    x = x + h;
end
out(end) = ThreePointEndpoint(f, x, -h);
end

function res = Extrapolate(initial)
sz = size(initial, 1);
res = zeros(sz, sz);
res(:, 1) = initial;
for i = 2:sz
    for j = 2:i
        res(i, j) = res(i, j - 1) + (res(i, j - 1) - res(i - 1, j - 1)) / (4^(j - 1) -
    end
end
end

initial = [1.570796; 1.896119; 1.974232; 1.993570];
res = Extrapolate(initial)
```

Exercise 8: The forward-difference formula can be expressed as

$$f'(x_0) = \frac{1}{h}[f(x_0 + h) - f(x_0)] - \frac{h}{2}f''(x_0) - \frac{h^2}{6}f'''(x_0) + \mathcal{O}(h^3)$$

Use extrapolation to derive an $\mathcal{O}(h^3)$ formula for $f'(x_0)$.

Answer. Let

$$N_1(h) = \frac{1}{h}[f(x_0 + h) - f(x_0)]$$

as our initial guess. Then we see that it has an error of

$$-\frac{h}{2}f''(x_0) - \frac{h^2}{6}f'''(x_0) + \mathcal{O}(h^3)$$

To get rid of the $\frac{h}{2}$ error, we use

$$N_2(h) = N_1\left(\frac{h}{2}\right) + \left[N_1\left(\frac{h}{2}\right) - N_1(h)\right]$$

So

$$N_2(h) = \frac{2}{h}\left[f\left(x_0 + \frac{h}{2}\right) - f(x_0)\right] + \left[\frac{2}{h}\left[f\left(x_0 + \frac{h}{2}\right) - f(x_0)\right] - \frac{1}{h}[f(x_0 + h) - f(x_0)]\right]$$

And so on. We do the same process but then with:

$$N_3(h) = N_2\left(\frac{h}{2}\right) + \frac{1}{15}\left[N_2\left(\frac{h}{2}\right) - N_2(h)\right]$$

to get our $O(h^3)$ approximation.

Exercise 9: Suppose that $N(h)$ is an approximation to M for every $h > 0$ and that

$$M = N(h) + K_1 h + K_2 h^2 + K_3 h^3 + \cdots,$$

for some constant $K_1, K_2, K_3, \ldots$. Use the values $N(h), N(\frac{h}{3})$, and $N(\frac{h}{9})$ to produce an $O(h^3)$ approximation to M.

Answer. We have that

$$M = N_1\left(\frac{h}{3}\right) + K_1\left(\frac{h}{3}\right) + K_2\left(\frac{h}{3}\right)^2 + \cdots$$

Then

$$3M = 3N_1\left(\frac{h}{3}\right) + 3K_1\left(\frac{h}{3}\right) + 3K_2\left(\frac{h}{3}\right)^2 + \cdots$$

so

$$2M = 3N_1\left(\frac{h}{3}\right) - N_1(h) + K_2\left(\frac{h^2}{3}\right) - K_2h^2 + \cdots$$

and

$$M = \frac{1}{2}\left(3N_1\left(\frac{h}{3}\right) - N_1(h)\right) - \frac{1}{3}K_2h^2 + \cdots$$

Now let

$$N_2(h) = \frac{1}{2}\left(3N_1\left(\frac{h}{3}\right) - N_1(h)\right)$$

We have:

$$N_2\left(\frac{h}{3}\right) = \frac{1}{2}\left(3N_1\left(\frac{h}{9}\right) - N_1\left(\frac{h}{3}\right)\right)$$

Notice that now we have:

$$M = N_2(h) - \frac{1}{3}K_2h^2$$
$$M = N_2\left(\frac{h}{3}\right) - \frac{1}{27}K_2h^2 + \cdots$$

Then

$$9M = 9N_2\left(\frac{h}{3}\right) - \frac{1}{3}K_2h^2 + \cdots$$

and

$$8M = 9N_3\left(\frac{h}{2}\right) - N_2(h)$$

So the final $O(h^3)$ approximation is

$$\frac{9}{8}\left(N_2\left(\frac{h}{3}\right) - N_2(h)\right)$$

We can write this in terms of $N_1(h), N_1(h/3), N_1(h/9)$:

$$\frac{9}{8}\left(\frac{1}{2}\left(3N_1\left(\frac{h}{9}\right) - N_1\left(\frac{h}{3}\right)\right) - \frac{1}{2}\left(3N_1\left(\frac{h}{3}\right) - N_1(h)\right)\right)$$

Exercise 14: Suppose that $N_1(h)$ is a formula that produces $O(h)$ approximations to a number M and that

$$M = N_1(h) + K_1(h) + K_2h^2 + \cdots,$$

for a collection of positive constants $K_1, K_2, \ldots$. Then $N_1(h), N_1(h/2), N_1(h/4), \ldots$ are all lower bounds for M. What can be said about the extrapolated approximations $N_2(h), N_3(h), \ldots$?