Solutions to Assignment #6

1. The plotting part of this problem is very easy to do because plotting piecewise-linear functions for given points is MATLAB/OCTAVE's *default plotting mode*. The next program, which produces Figure 1, is mostly an excuse to show a few plotting commands which you may not have used yet: setting default values for line widths and such, `clf`, `subplot`, `legend`, `title`.

prob1a6.m

```
% PROB1A6  Produce one plot for both parts of problem 1 on A#6.

% useful commands so figures have bolder lines and markers
set(0,'defaultlinelinewidth',2.0)
set(0,'defaultlinemarkersize',12.0)
clf  % clear the current figure

% part (a)
xxa = 0:.01:3;  % fine grid
xa = [0, 1, 2, 3];
subplot(1,2,1)
plot(xxa,cos(xxa),'r',xa,cos(xa),'b-o')
legend('y = cos(x)','y = R(x)')
title('part (a)'),  xlabel x

% part (b)
xxb = 0:.01:5;  % fine grid
xb = [0, 0.5, 1.2, 3, 5];
f = @(x) exp(-x.^2);
subplot(1,2,2)
plot(xxb,f(xxb),'r',xb,f(xb),'b-o')
legend('y = e^{-x^2}','y = R(x)')
title('part (b)'),  xlabel x
```

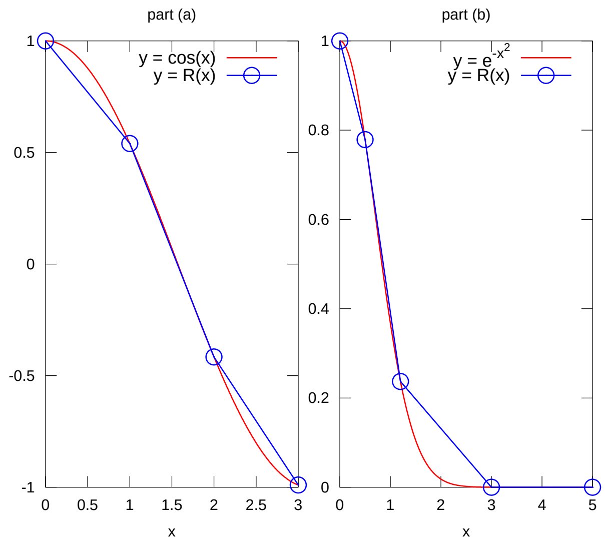

FIGURE 1. Plot of the function and its piecewise-linear interpolant $R(x)$, for each part of problem 1.

(a) Now we apply the theorem. Here $f''(x) = -\cos x$ so $M = \max_{0\le x\le 3} |f''(x)| = 1$. Also $\Delta x = \max_{j=0,1,2} |x_{j+1} - x_j| = 1$; the grid is equally-spaced. Thus we know in advance that

$$|R(x) - f(x)| \le \frac{M}{2}\Delta x^2 = \frac{1}{2}\cdot 1^2 = 0.5$$

for $0 \le x \le 3$. The plot confirms that this upper bound is correct, but a bit pessimistic; the actual maximum of $|R(x) - f(x)|$ is about 0.1.

(b) Here $f'(x) = \exp(-x^2)(-2x)$ and $f''(x) = \exp(-x^2)(4x^2 - 2)$. An apparently-reasonable upper bound for $|f''(x)|$ comes from maximizing each factor:

$$M = \max_{0\le x\le 5} |f''(x)| \le \left(\max_{0\le x\le 5} |\exp(-x^2)|\right)\left(\max_{0\le x\le 5} |4x^2-2|\right) = (1)(4\cdot 5^2 - 2) = 98.$$

This is pessimistic because the factors are large at different places, while we would really want the maximum of the product. Because $\Delta x = 2$, the length of the largest gap, it follows that

$$|R(x) - f(x)| \le \frac{M}{2}\Delta x^2 = \frac{98}{2}\cdot 2^4 = 196.$$

The plot confirms both the truth and the substantial pessimism of this statement.

(*The largest* actual *error* $|R(x) - f(x)|$, *of about 0.15, occurs on an subinterval* $[1.2, 3]$ *which is shorter than* $\Delta x = 2$. *This illustrates that the best interpolation strategy would come from putting interpolation points in the right places. It is less efficient to uniformly reduce* Δx.)

2. *I apologize for not making clear where my "summary" of the trapezoid rule started and ended. The requested calculation is simple. It shows that, for functions with bounded second derivatives, the trapezoid rule can be made as accurate as desired by choosing n appropriately large. Because of the square on "n" in the denominator of the final expression, doubling the number of points generally reduces the error by a factor of four.*

We use the integral triangle inequality to turn our bound on the error in piecewise-linear interpolation into a bound for the error in the trapezoid rule:

$$\begin{aligned}\left|\int_a^b R(x)\,dx - \int_a^b f(x)\,dx\right| &\le \int_a^b |R(x) - f(x)|\,dx \le \int_a^b \frac{M}{2}\Delta x^2\,dx \\ &= \frac{M}{2}\Delta x^2\,(b-a) = \frac{M}{2}\left(\frac{b-a}{n}\right)^2(b-a) \le \frac{M(b-a)^3}{2n^2}.\end{aligned}$$

Note that at one stage we actually integrate a constant, and in the next stage we use the expression $\Delta x = (b-a)/n$.

3. **(a)** Here $n = 1$. There are no interior points at which we "match up" the cubic parts. We seek $S_0(x) = a_0 + b_0(x-0) + c_0(x-0)^2 + d_0(x-0)^3$ satisfying only four conditions:

$$S_0(0) = 1.0000, \quad S_0(0.5) = 2.71828, \quad S_0''(0) = 0, \quad S_0''(0.5) = 0.$$

These conditions determine the cubic. Note that $S_0''(x) = 2c_0 + 6d_0(x-0)$. Thus, the first and third conditions imply $a_0 = 1$ and $c_0 = 0$, so that the second and fourth conditions say

$$1 + b_0(0.5) + d_0(0.5)^3 = 2.71828, \quad 6d_0(0.5) = 0.$$

respectively. The latter of these implies $d_0 = 0$, and thus the former implies $b_0 = (2.71828 - 1)/(0.5) = 3.43656$. Thus

$$S_0(x) = 1 + 3.43656x.$$

This is a linear function. (*Why? Linear functions are the only cubic polynomials which have zero second derivative at two distinct locations, because the second derivative of a cubic polynomial has either exactly one root or is identically zero.*)

(b) Here you should set up and solve a system using MATLAB/OCTAVE. When I do this I get:

$$S_0(x) = -0.29004996 - 2.7512863(x-0.1) + 4.38125(x-0.1)^3,$$
$$S_1(x) = -0.56079734 - 2.6198488(x-0.1) + 1.314375(x-0.1)^2 - 4.38125(x-0.1)^3.$$

4. **(a)** Using `ncspline.m` to "compute" the natural cubic spline means using the program to compute the coefficients. If you look at the program you will see lines 53 and 54 saying

```
% uncomment this to view table of coefficients:
%[b c d]
```

For this part of the problem, you get the same solution, i.e. coefficients, as problem **3(b)**. The following program produces the left plot in Figure 2.

`prob4aona6.m`

```
% PROB4AONA6  Solve problem 4(a) on Assignment #6: compute and plot the spline.
x = [0.1, 0.2, 0.3];
y = [-0.29004996, -0.56079734, -0.81401972];
xp = 0.1:.0001:0.3;
yp = ncspline(x,y,xp);  % with line 54 uncommented, this produces coeffs [b c d]
set(0,'defaultlinelinewidth',2.0,'defaultlinemarkersize',12.0)
plot(x,y,'o',xp,yp), legend('data','natural cubic spline')
xlabel x, axis tight
```

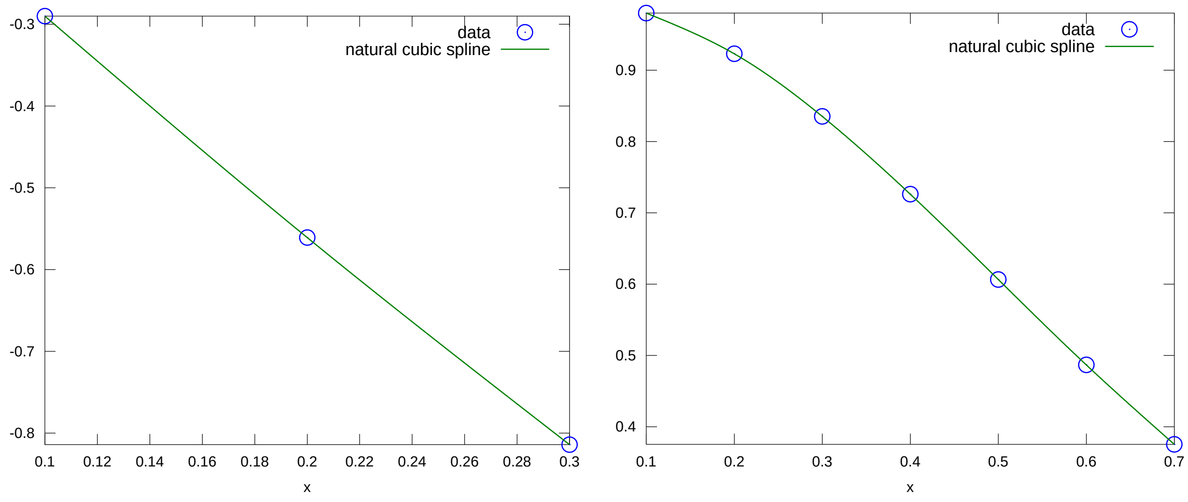

FIGURE 2. (left) Plot of the data points and the natural cubic spline interpolant $S(x) = \{S_0(x), S_1(x)\}$ through the points, for problem **4(a)**. (right) Plot of the data points and the spline interpolant for problem **4(b)**.

(b) A very similar program produces the right plot in Figure 2. The program, and a slightly-modified version of `ncspline.m` to output the coefficients more directly, are posted online at

`http://www.dms.uaf.edu/~bueler/prob4bona6.m`
`http://www.dms.uaf.edu/~bueler/bcd_ncspline.m`

The six polynomials are:

$$S_0(x) = 0.98020 + -0.50172(x-0.1) + 0.00000(x-0.1)^2 + -6.90812(x-0.1)^3$$
$$S_1(x) = 0.92312 + -0.70896(x-0.2) + -2.07243(x-0.2)^2 + 3.77058(x-0.2)^3$$
$$S_2(x) = 0.83527 + -1.01033(x-0.3) + -0.94126(x-0.3)^2 + 1.32581(x-0.3)^3$$
$$S_3(x) = 0.72615 + -1.15881(x-0.4) + -0.54352(x-0.4)^2 + 1.69619(x-0.4)^3$$
$$S_4(x) = 0.60653 + -1.21663(x-0.5) + -0.03466(x-0.5)^2 + 2.22942(x-0.5)^3$$
$$S_5(x) = 0.48675 + -1.15668(x-0.6) + 0.63417(x-0.6)^2 + -2.11388(x-0.6)^3$$

5.

prob5ona6.m

```
% PROB5ONA6  Solve problem 5 on Assignment #6.

x = [0.1, 0.2, 0.3];
y = [-0.29004996, -0.56079734, -0.81401972];

format long
y018 = ncspline(x,y,0.18)
f = @(x) x.^2 .* cos(x) - 3 * x;
f(0.18)
abs(y018 - f(0.18))  % actual error  |S0(0.18) - f(0.18)|

% I filled in the coefficients of these polynomials after running the above the
%   first time to get b,c,d
S0 = @(x) -0.29004996 - 2.7512863 * (x-0.1) + 4.38125 * (x-0.1)^3;
dS0dx = @(x) -2.7512863 + 3 * 4.38125 * (x-0.1)^2;
dfdx = @(x) 2*x.*cos(x) - x.^2 * sin(x) - 3;

dS0dx(0.18)
dfdx(0.18)
abs(dS0dx(0.18) - dfdx(0.18))  % actual error  |S0'(0.18) - f'(0.18)|
```

```
>> prob5ona6
y018 = -0.507909664000000
ans = -0.508123464353665
ans =  2.13800353664917e-04
ans = -2.66716630000000
ans = -2.65161682877527
ans =  0.0155494712247268
```

Thus $S(0.18) = -0.507909664$ while $f(0.18) = -0.508123464$ for an actual error of about 2.1×10^{-4}. Also, $S_0'(0.18) = -2.6671663$ while $f'(0.18) = -2.6516168$, for an actual error of about 1.6×10^{-2}.