Solutions to Assignment #8

Problems 6.4, exercise 14: The answer turns out to be yes, which means that I have to be organized in writing it up. There are a lot of facts to check; I count 8. Let's start with the first and second derivatives:

$$f'(x) = \begin{cases} 2 + 3(x+1)^2 & [-1,0] \\ 5 + 6x & [0,1] \\ 11 + 6(x-1) - 3(x-1)^2 & [1,2] \end{cases} \qquad f''(x) = \begin{cases} 6(x+1) & [-1,0] \\ 6 & [0,1] \\ 6 - 6(x-1) & [1,2] \end{cases}$$

Next, note $f''(-1) = 0$ and $f''(2) = 0$ so we it is "natural". Furthermore the second derivative is continuous at $x = 0$ ($f''(0) = 6$) and at $x = 1$ ($f''(1) = 6$). The first derivative is continuous at $x = 0$ ($f'(0) = 5$) and at $x = 1$ ($f'(1) = 11$). Finally the function itself is continuous at $x = 0$ ($f(0) = 3$) and at $x = 1$ ($f(1) = 11$). The function is, therefore, the unique natural cubic spline through the points $(-1, 0), (0, 3), (1, 11), (2, 24)$. (*Checking continuity at an x-value means plugging that value into the polynomial on each side and checking that results agree.*)

Problems 6.4, exercise 17: *This is a good one to do by hand, but I will only show here how to check your answer with* `ncspline.m`. The program `ncspline.m` only computes the coefficients internally, without displaying them, so I copied it to a new file `ncspline_dump.m`, and uncommented the line "`%[b c d]`". I modified just a bit more to display a clean table of coefficients. Then I ran it, with result

```
>> ncspline_dump([-1 0 1],[13 7 9],0);
table [a | b | c | d] of coefficients:
ans =
   13   -8    0    2
    7   -2    6   -2
```

Thus the natural cubic spline with these values and knots is

$$S(x) = \begin{cases} 13 - 8(x+1) + 2(x+1)^3, & x \in [-1,0] \\ 7 - 2x + 6x^2 - 2x^3. & x \in [0,1] \end{cases}$$

On the other hand, producing a clean plot is exactly the job for which `ncspline.m` was built, so:

```
>> x=[-1 0 1]; y=[13 7 9]; xf=-1:0.01:1; yf=ncspline(x,y,xf); plot(x,y,'o',xf,yf)
```

(The result is not shown.)

Computer Problems 6.4, exercise 1: (*Sorry, I overlooked the* "several test cases" *phrase. I should have said that one test case is fine.*) My curve on graph paper looked like the script letter "v". It generated two lists of $x(t)$ and $y(t)$ values, and each list generated a cubic spline (with independent variable t in each case). The function `ncspline.m` generated finely-interpolated x and y values, and plotting these, without any reference to t, gave back the figure. The rest is details:

letterv.m

```
% LETTERV  Use natural cubic splines (ncspline.m) to plot a script letter
% "v", using points taken from a written letter on graph paper.

t = 0:1:10;         % 11 points
xy = [3.0 7.0;      % I found it easiest to enter coordinates of points
      4.0 7.9;      %     this way, as a list of (x y) pairs
      4.8 6.0;
      5.1 4.0;
```

```
      5.3 2.6;
      6.0 2.0;
      7.1 3.0;
      7.9 5.0;
      8.2 6.0;
      9.0 7.4;
      10.0 7.6];
x = xy(:,1)';  y = xy(:,2)';      % extract just x and y separately

tf = 0:0.01:10;     % fine grid for plotting (on t-vs-x and t-vs-y axes)
xf = ncspline(t,x,tf);   yf = ncspline(t,y,tf);

% t-versus-x and t-versus-y plots, using subplot:
figure(1)
subplot(2,1,1), plot(t,x,'o',tf,xf), grid on, xlabel t, ylabel x
title("parameterized curve in pieces")
subplot(2,1,2), plot(t,y,'o',tf,yf), grid on, xlabel t, ylabel y

% plot in x,y plane; parameter "t" is never mentioned!
figure(2), plot(x,y,'o',xf,yf)
axis equal, grid on, xlabel x, ylabel y, title("the letter v")
```

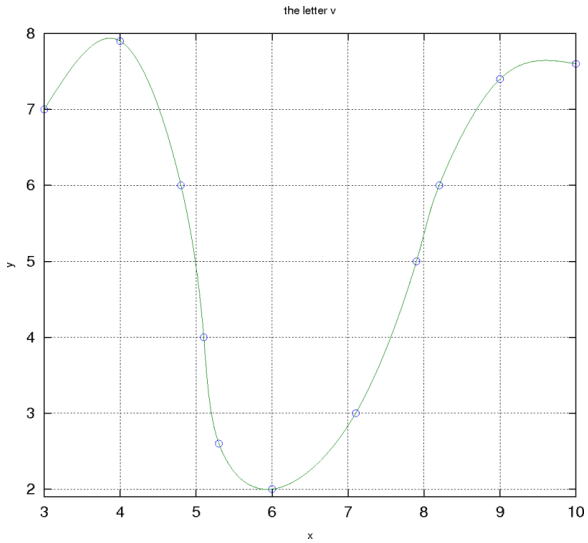

FIGURE 1. The script letter "v" by cubic splines through by-hand identified points. This is "`figure(2)`" from the code `letterv.m`.

Problems 7.2, exercise 1: Somewhat tedious, but the method is clear. Write down the Lagrange polynomials $\ell_0(x), \ldots, \ell_3(x)$ and integrate them to find $A_0, \ldots, A_3$, respectively, in the integration rule

$$\int_0^1 f(x)\,dx \approx A_0 f(0) + A_1 f(\frac{1}{3}) + A_2 f(\frac{2}{3}) + A_3 f(1).$$

Here is an alternate way that exploits our available tool, MATLAB/OCTAVE. The rule should be able to integrate cubic polynomials exactly. (*Why?*) This gives 4 equations in the unknown

A_i:

$$1 = \int_0^1 1\,dx = A_0 + A_1 + A_2 + A_3,$$
$$1/2 = \int_0^1 x\,dx = 0A_0 + (1/3)A_1 + (2/3)A_2 + 1A_3$$
$$1/3 = \int_0^1 x^2\,dx = 0A_0 + (1/9)A_1 + (4/9)A_2 + 1A_3$$
$$1/4 = \int_0^1 x^3\,dx = 0A_0 + (1/27)A_1 + (8/27)A_2 + 1A_3$$

This system of equations has numerical solution:

```
>> format rat
>> [1 1 1 1; 0 (1/3) (2/3) 1; 0 (1/9) (4/9) 1; 0 (1/27) (8/27) 1] \ [1 (1/2) (1/3) (1/4)]'
ans =
        1/8
        3/8
        3/8
        1/8
```

So

$$\int_0^1 f(x)\,dx \approx \frac{1}{8}\left[f(0) + 3f(\frac{1}{3}) + 3f(\frac{2}{3}) + f(1)\right].$$

Problems 7.2, exercise 14: Similar to previous.

Problems 7.2, exercise 31: This is fairly similar to **Exercise 2** below. The error term on page 482 is

$$-\frac{(b-a)}{12}h^2 f''(\xi).$$

The derivatives of the integrand are

$$f'(x) = 1 - 2xe^{-x^2}, \qquad f''(x) = (-2 + 4x^2)e^{-x^2}.$$

The maximum of $|f''(x)|$ on the interval is bounded:

$$|f''(x)| = |-2 + 4x^2|e^{-x^2} \le 14e^{-1} = 5.15$$

(*This is quite pessimistic because the magnitude of f'' is in fact less than 1 on the interval* $[1, 2]$. *But it suffices.*) Require the error to be smaller than the given number:

$$\left|\frac{(b-a)}{12}h^2 f''(\xi)\right| = \frac{h^2}{12}|f''(\xi)| \le \frac{5.15h^2}{12} \le 0.5 \times 10^{-7}.$$

This equivalent to $h \le \sqrt{1.165 \times 10^{-7}} = 3.4 \times 10^{-4}$ which says $n = 1/h \ge 2929.8$ so I get $n = 2930$ subintervals.

Exercise 1: **(a)** Keeping the name x_1 for the center point $(a+b)/2$, the Lagrange polynomials simplify to

$$\ell_0(x) = \frac{(x-x_1)(x-b)}{(a-x_1)(a-b)} = \frac{2}{(b-a)^2}\left(x^2 - (x_1+b)x + x_1 b\right),$$
$$\ell_1(x) = \frac{(x-a)(x-b)}{(x_1-a)(x_1-b)} = -\frac{4}{(b-a)^2}\left(x^2 - (a+b)x + ab\right),$$
$$\ell_2(x) = \frac{(x-a)(x-x_1)}{(b-a)(b-x_1)} = \frac{2}{(b-a)^2}\left(x^2 - (a+x_1)x + ax_1\right).$$

The right-hand expressions are one of several forms in which the Lagrange polynomials are relatively easy to integrate. The result of integration is given in the problem statement. These integrals were needed to compute the coefficients A_0, A_1, A_2 in Simpson's rule, $\int_a^b f(x)\,dx \approx \sum_{i=0}^2 A_i f(x_i)$.

(b) I'll give some detail here:

$$\begin{aligned}\int_a^b (x-a)(b-x)\,dx &= -\int_a^b x^2-(a+b)x+ab\,dx = -\left[\frac{x^3}{3}-(a+b)\frac{x^2}{2}+abx\right]_a^b \\ &= -\frac{b^3}{3}+(a+b)\frac{b^2}{2}-ab^2+\frac{a^3}{3}-(a+b)\frac{a^2}{2}+a^2b = \frac{1}{3}(a-b)(a^2+ab+b^2)+\frac{1}{2}(b-a)(a^2+b^2) \\ &= \frac{1}{6}(b-a)\left[-2a^2-2ab-2b^2+3a^2+3b^2\right] = \frac{1}{6}(b-a)(b-a)^2 = \frac{(b-a)^3}{6}\end{aligned}$$

This integral was necessary to complete the error formula for the trapezoid rule,

$$\int_a^b f(x)\,dx = \frac{b-a}{2}\left[f(a)+f(b)\right]-\frac{f''(\xi)}{12}(b-a)^3.$$

Exercise 2: The error formula for the composite Simpson's rule on page 484 says

$$\int_a^b f(x)\,dx = S_n - \frac{b-a}{180}h^4 f(4)(\xi)$$

for S_n the composite Simpson's approximation and $h=(b-a)/n$. In our case $f(x)=\pi+\sin(3x)$ so $f^{(4)}(x)=81\sin(3x)$. Thus

$$\left|\int_0^{10} f(x)\,dx - S_n\right| = \frac{10-0}{180}h^4\left|81\sin(3\xi)\right| \le \frac{9}{2}h^4,$$

using the standard inequality $|\sin\theta|\le 1$. Recalling $h=10/n$, we want to choose

$$\frac{9}{2}h^4 \le 10^{-8} \quad\Longleftrightarrow\quad \frac{9}{2}\frac{10^4}{n^4} \le 10^{-8} \quad\Longleftrightarrow\quad n^4 \ge 4.5\times 10^{12} \quad\Longleftrightarrow\quad n \ge 1456.475.$$

But n must be an even integer, so $n=1458$ is the smallest value we expect to get that accuracy.

Does it get that accuracy? On the one hand, the exact answer is

$$\int_0^{10} \pi+\sin(3x)\,dx = 10\pi+\frac{1}{3}(1-\cos(30)).$$

On the other hand,

```
>> f=@(x) pi + sin(3*x);
>> n=1458; h=10/n; xi=0:h:10;
>> Sn = (h/3) * sum([1 repmat([4 2],1,n/2 -1) 4 1] .* f(xi))
Sn =     31.6978427195495
>> exact = 10*pi+(1/3)*(1-cos(30))
exact =  31.6978427192687
>> abs(Sn - exact)
ans = 2.80756751180888e-10
```

Thus the error of 2.8×10^{-10} is smaller than 10^{-8}.

Why the gap? That is, why is it that when we used $n=1458$ the error was quite a bit less than 10^{-8}? Because of "$|\sin(3\xi)|\le 1$", which is true but not precise. In fact I find that the smallest n for which the error is less than 10^{-8} is $n=598$; this comes from a MATLAB/OCTAVE *program-at-the-command line:*

```
>> for n=8:2:1458 ...
> h=10/n; xi=0:h:10; Sn=(h/3)*sum([1 repmat([4 2],1,n/2 -1) 4 1].*f(xi));
> if abs(exact-Sn)<1e-8, n, break, end
> end
n =       598
```

Witness the power of a super-calculator ... eh?

Exercise 3: From the statement of the problem I immediately was able to set up equations for the coefficients. In particular, as with cubic splines, for $j = 0, \ldots, n-1$ let

$$h_j = t_{j+1} - t_j \qquad \text{and} \qquad m_j = \frac{y_{j+1} - y_j}{h_j}.$$

The requirement $Q_j(t_j) = y_j$ implies $a_j = y_j$. The other two requirements immediately become

$$\begin{aligned} y_j + b_j h_j + c_j h_j^2 &= y_{j+1} && j = 0, \ldots, n-1, \\ b_j + 2c_j h_j &= b_{j+1} && j = 0, \ldots, n-2. \end{aligned}$$

This is $2n-1$ equations in the $2n$ unknowns b_j, c_j (for $j = 0, \ldots, n-1$), so the additional requirement $c_0 = 0$ is enough to hope for a unique solution.

In particular, take the first of the above equations and solve it for b_j:

$$b_j = \frac{y_{j+1} - y_j - c_j h_j^2}{h_j} = m_j - c_j h_j.$$

Substitute this into the second set of equations, for each of b_j and b_{j+1}, to get

$$m_j - c_j h_j + 2c_j h_j = m_{j+1} - c_{j+1} h_{j+1}$$

or, equivalently,

$$h_j c_j + h_{j+1} c_{j+1} = m_{j+1} - m_j$$

for $j = 0, \ldots, n-2$. This last is a set of $n-1$ equations in the n unknowns c_j, $j = 0, \ldots, n-1$, so the additional condition $c_0 = 0$ is needed.

We can now write a matrix equation for the c_j:

$$\begin{bmatrix} 1 & & & & \\ h_0 & h_1 & & & \\ & h_1 & h_2 & & \\ & & \ddots & \ddots & \\ & & & h_{n-2} & h_{n-1} \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ c_{n-1} \end{bmatrix} = \begin{bmatrix} 0 \\ m_1 - m_0 \\ m_2 - m_1 \\ \vdots \\ m_{n-1} - m_{n-2} \end{bmatrix}$$

This is a lower-triangular and bidiagonal system. As long has $h_1, \ldots, h_{n-1}$ are non-zero, which they are merely because we require distinct nodes, the solution c_j is unique. Once c_j are found we can find b_j immediately.

The solution can be found by fairly-obvious forward substitution, as in the first "`for`" loop in the following code:

qspline.m

```
function yi = qspline(t,y,xi)
% QSPLINE  yi = qspline(t,y,xi)
% Computes not-particularly-natural quadratic spline through points
%              t = t(1),...,t(n+1)
%              y = y(1),...,y(n+1)
% and evaluates at the x-values in xi, to give result yi.   Requires
% t(1) < t(2) < ... < t(n+1) and xi in [t(1),t(n+1)], and finds
% quadratic polynomials in this form:
%     Q_i(x) = y_i + b_i (x - t_i) + c_i (x - t_i)^2
% Try:  >> x=0:3; y=exp(x); xp=0:0.01:3; yq = qspline(x,y,xp);
%       >> plot(x,y,'o',xp,exp(xp),xp,yq)
%       >> legend('data','underlying function','quadratic spline')
% Compare to ncspline.m for natural cubic splines.

t = t(:);  y = y(:);          % force t and y to be column vectors
% check that inputs make sense
if (length(t) ~= length(y)), error('t,y must be vectors of same size'),  end
n = length(t) - 1;
if n <= 0, error('must have at least two points to produce spline'), end
```

```
if any(xi < t(1)) | any(xi > t(n+1))
  error('xi must be in interior (interval [t(1),t(n+1)])')
end

h = diff(t);                  % gaps:   h(i) = t(i+1) - t(i)
m = diff(y) ./ h;             % slopes: m(i) = (y(i+1) - y(i)) / h(i)

% directly solve bidiagonal system for coeffs c_i, by forward substitution
c = zeros(size(h));   % c_0 = 0 already
for j=2:n
  c(j) = ( m(j)- m(j-1) - h(j-1)*c(j-1) ) / h(j);
end

% determine other coeffs from c:
b = m - h .* c;

% evaluate quadratics; requires finding interval in which xi lies
xi = xi(:);
yi = zeros(size(xi));
for k = 1:length(xi)
  i = 1;
  while xi(k) > t(i+1)
    i = i + 1;
  end
  % now xi(k) is in [t(i),t(i+1)], so apply Hoerner to evaluate Q_i(x):
  dx = xi(k) - t(i);
  yi(k) = y(i) + dx * (b(i) + dx * c(i));
end
```

The "geometric consequences" of the choice $c_0 = 0$ are that the left-most quadratic is a straight line, which is somewhat ugly, while the "algorithmic consequence" is already mentioned: the system of equations is lower triangular. Another, possibly better choice is to require $c_0 = c_1$, implying extra smoothness at the first interior node. This condition can be easily dealt-with, but it does break the literal lower-triangularness of the system. But both of these suggestions are obviously flawed in that they are asymmetric, because they treat the left end of the spline differently from the right. And, at this point, I don't have a really great suggestion ...