# Assignment #7

## Due Thursday, 2 November at the start of class

Please read sections 2.5, 4.8, 4.11, 5.1, 5.2, and 5.3 of the textbook (J. Epperson, *An Intro. to Numerical Methods and Analysis*, 2nd edition).

**Section 2.5, pages 74–78:** For these problems I recommend first turning Algorithm 2.5 into an (easy!) MATLAB code with first line "`function z = trap(f,a,b,n)`."

- **Exercise 1.** A great problem for verifying your code. Note that "How small does $h$ have to be ..." requires additional thoughts and writing, not just running the code. (Ditto for following problems.)
- **Exercise 6.**
- **Exercise 8.**
- **Exercise 9.** This technique is called "linear extrapolation to $h = 0$".
- **Exercise 14.**

**Section 5.1, pages 265–266:**

- **Exercise 1.**
- **Exercise 3.**

**Section 5.2, pages 190–191:**

- **Exercise 3.** The corrected trapezoid rule is "$T_n^C(f) = \dots$" on page 267.

**P8.** (*This problem gives the flavor of splines independently of the textbook. The splines here are less good-looking, but slightly simpler, than those covered in section 4.8 of the textbook. This problem is stated so as to make the indexing in* MATLAB *easy.*)

Suppose we have distinct nodes $\{x_i\}_{i=1}^{n+1}$ which are in order so that $x_i < x_{i+1}$. Suppose we have any data values $\{y_i\}_{i=1}^{n+1}$. I claim we can build $n$ quadratic polynomials

$$p^k(x) = a^k + b^k(x - x_k) + c^k(x - x_k)(x - x_{k+1}),$$

for $k = 1, \dots, n$, on each subinterval $[x_k, x_{k+1}]$, so that

(i) $p^k(x_k) = y_k$ for each $k = 1, 2, \dots, n$,
(ii) $p^k(x_{k+1}) = y_{k+1}$ for each $k = 1, 2, \dots, n$, and
(iii) $(p^{k-1})'(x_k) = (p^k)'(x_k)$ for each $k = 2, 3, \dots, n$.

Note that (i) corresponds to $n$ equations, (ii) to $n$ equations, and (iii) to $n-1$ equations, for a total of $3n - 1$ equations. On the other hand there are $3n$ unknowns, namely the coefficients $a^1, \dots, a^n, b^1, \dots, b^n, c^1, \dots, c^n$.

To uniquely determine a spline (curve) through the data therefore requires one more condition. One has to choose something; I choose

(iv) $c_1 = 0$

so that $p^1(x)$ is actually a linear polynomial. (*Any choice here is awkward.*)

**(a)** For concreteness, suppose $n = 3$. Write down the nine equations in nine unknowns which determine the polynomials $p^1, p^2, p^3$. Explain how to solve these equations systematically, first by finding all the $a^k$, then all the $b^k$, and then by setting up a simple system for just the $c^k$. This system can then be solved by forward substitution.

**(b)** For the particular nodes $x_i = i$, $i = 1, 2, 3, 4$, and data $y_i = \exp(-x_i^2)$, set up and solve the system. Use MATLAB as needed. Print the nine coefficients $a^k, b^k, c^k$.

**(c)** Write a MATLAB code with first line

```
function [a, b, c] = quadspline(x,y)
```

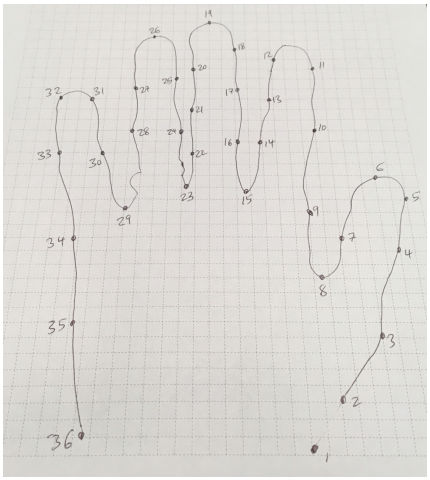
which implements the above strategy for any $n$. It will check that the input arrays have the same length. It will get $n$ from `length(x)`. It will return arrays of coefficients; it does not plot anything. Test your code on the part **(b)** problem.

**(d)** Finally, write a separate code

```
function plotqs(x,y,a,b,c)
```

which plots the quadratic spline defined by the lists of coefficients `a,b,c`. Note that `x,y` are the same nodes and data which go into `quadspline.m`, while `a,b,c` are the outputs of `quadspline.m`. Your code will choose a relatively fine grid (e.g. at least 20 points) on each subinterval $[x_k, x_{k+1}]$ so as to plot $p^k(x)$ on that interval. Test your code on the part **(b)** problem.

**P9.** Start this problem by getting some grid paper with roughly 1/4 inch grid. (I googled "printable grid paper," etc.) Your hand will fit on a letter size piece easily. Trace the outline of your hand. Add 30 to 50 *roughly* equally-spaced points along the outline, especially including tips of fingers and saddle points between fingers. Number these points to keep track. Let $n$ be the number of points minus one.



At this point my result looked like the figure at right, with 36 points and $n = 35$.

Write down, in an editor so you only have to do it once, the $(x_k, y_k)$ locations for each point. The numbering can be regarded as $t$-values, namely $t_k = k$ for $k = 1, \ldots, n+1$. Now we have two functions $x(t)$ and $y(t)$, both sampled, which define a parametric curve for the outline. That is, $x(t)$ is approximated by pairs $(t_k, x_k)$ and $y(t)$ by $(t_k, y_k)$.

Now use the built-in functions `spline` and `ppval` to draw a cubic spline that is the outline of your hand. You will apply `spline` twice, first to the $(t_k, x_k)$ points and then to the $(t_k, y_k)$. However, when you plot you only plot the $(x, y)$ values.