Assignment #8

Due Wednesday 10 November, 2021 at the start of class.

Submit on paper or by email: `elbueler@alaska.edu`

Exercise 5.2.1. Do parts (a) and (b) only.

Exercise 5.2.3. Use Theorem 5.2.2.

Exercise 5.2.4.

Exercise 5.3.4. Do parts (a) and (b) only. Feel free to use `interp1()` for this.

Exercise 5.6.1. Do parts (a) and (c) only.

P6. *This problem replaces Exercise 5.1.1, and is a good starting-point for this Assignment! Each part involves generating a plot; turn in that plot. Please also turn in the code which generated the plot and the uniform error estimate.*

Suppose we want to interpolate the function $f(x) = \tanh(x)$ using the following seven nodes t_i and points y_i:

```
>> t = [-4 -2 0 1 2 4 6];
>> y = tanh(t);
```

(a) Generate a well-labeled plot of the (high-degree) polynomial interpolant $p_a(x)$ of this data using `polyfit()` and `polyval()`, also showing $f(x)$ and the data points in the same figure:

```
>> c = polyfit(t,y,6);
>> xx = -4:.01:6;  yy = tanh(xx);            % for plotting
>> plot(xx,polyval(c,xx), xx,yy, t,y,'ko')
>> legend('p_a(x)','f(x)=tanh(x)','interpolation points')
>> xlabel x,  ylabel y
```

Using the same 1001 evaluation points `xx`, use one additional line of Matlab to accurately estimate the uniform error estimate $\|f - p_a\|_\infty$.

(b) Let $p_b(x)$ be the piecewise-linear interpolant of $f(x)$ using the same seven points (t_i, y_i). Using the same plotting style as in part **(a)**, generate a new well-labeled plot of $p_b(x)$ using `interp1()`, plus $f(x)$ and the interpolation points. Again, accurately estimate $\|f - p_b\|_\infty$.

(c) Let $p_c(x)$ be the cubic spline interpolant of $f(x)$ using the same seven points (t_i, y_i). Using the same plotting style, generate a well-labeled plot of $p_c(x)$ using `interp1()`, and accurately estimate $\|f - p_c\|_\infty$.

(d) Of the three graphs, I think $p_c(x)$ looks the most like $f(x)$, but the fit is still not great. However, it is clear how to add two more interpolation nodes to get a much better fit. Do so. That is, regenerate the plot of a new cubic spline interpolant $\tilde{p}_c(x)$ through nine points, including the existing seven. Use the same plot style as usual. Compute the uniform error $\|f - \tilde{p}_c\|_\infty$ and confirm it is greatly reduced.

(e) Going back to $p_a(x)$, the polynomial interpolant in part **(a)**, recompute it using the same nine points as in part **(d)**, and plot the result $\tilde{p}_a(x)$ in the usual style. Compute $\|f - \tilde{p}_a\|_\infty$. Did it get better?

P7. *This problem replaces 5.1.4.*

Define

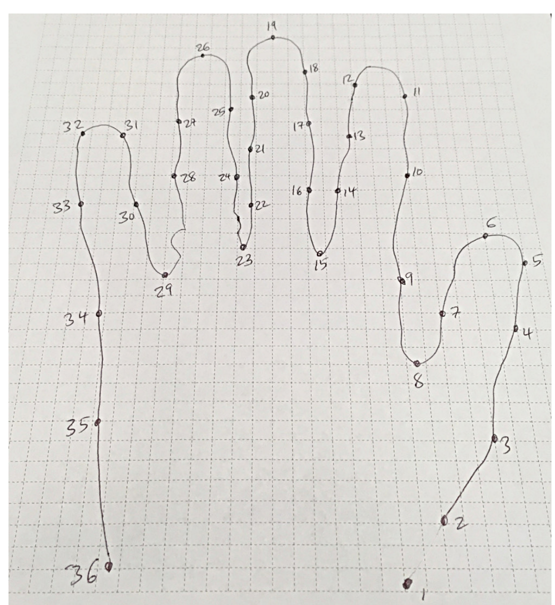
$$q(x) = a\frac{x(x-1)}{2} - b(x-1)(x+1) + c\frac{x(x+1)}{2}.$$

(a) Show that q is a polynomial interpolant through the points $(-1, a)$, $(0, b)$, $(1, c)$.

(b) What important properties do the three functions $f_1(x) = \frac{x(x-1)}{2}$, $f_2(x) = -(x-1)(x+1)$, and $f_3(x) = \frac{x(x+1)}{2}$ have? What should we call these functions?

P8. Find some grid paper with roughly 1/4 inch grid and trace the outline of your hand on it. (*I googled "printable grid paper," etc.*) Add 30 to 50 *roughly* equally-spaced points along the outline, generally including tips of fingers and saddle points between fingers. (*At this point my result looked like the figure below, with $n = 36$ points. You can read values off this graph if you want, and you'll get a picture of* my *hand, but yours is more fun!*) Type into the Matlab (or other) editor, so you only have to do it once, the (x_k, y_k) locations of each point, for $k = 1, \ldots, n$, choosing coordinates on the grid paper in some manner.

Now the idea is to get an interpolant which is a **parameterized curve** $(x(t), y(t))$. The indexing can be regarded as t-values, namely $t_k = k$ for $k = 1, \ldots, n$. The function $x(t)$ interpolates all the pairs (t_k, x_k) and $y(t)$ interpolates all the (t_k, y_k) pairs.



Plot the interpolating parameterized curve $(x(t), y(t))$ in the x, y plane using the Matlab `interp1()` function (twice). For plotting you will need to generate a fine grid of t values on the interval $[1, n]$. Turn in the plot and your code.

(*Only plot the (x, y) values in the main figure, but feel free to generate separate figures for the functions $x(t)$ and $y(t)$; this is optional. Other than the data for the points (x_k, y_k), your Matlab program should only be a few lines.*)