Assignment #7

Due Friday, 8 November 2024, at the start of class

This Assignment is based mostly on Chapter 10 of the textbook,[1] with the exception of problem **P8** which is from Chapter 8. Please read all of sections 10.1–10.5. You can skip sections 10.6 and 10.7. Reading section 9.2 about Richardson extrapolation would also be a good idea, but I will go over this idea in class; it relates only to Romberg integration in section 10.5.

These expectations always apply to homework:

1. Please put the problems in the order they appear below.
2. When you use MATLAB/etc., show the commands you used along with the results.
3. Please keep a clear distinction between codes, input commands, and computed results and/or figures.
4. Other than the text you write, please minimize use of paper. For example, computer outputs and figures do not need extra space.

Do these exercises from the textbook:

CHAPTER 10

- Exercise 2 on page 248
- Exercise 3 on page 249
- Exercise 6 on page 249

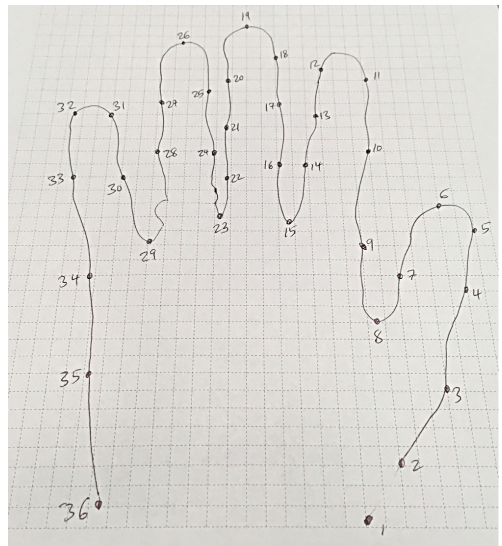
Do these additional problems:

P8. **(a)** Trace your hand on a grid and mark 25 to 40 (roughly) equally-spaced points along the trace, and store them into 1D Matlab arrays (i.e. vectors). Let n be the number of points. Here are two ways to do this; pick one:

- Find/print some grid paper and trace the outline of your hand on it. Mark the points by hand and enter the x and y coordinates in an editor.
- Open a big Matlab figure window. Put your hand on the screen. Use a Matlab command like `[x,y] = ginput(35)`, and click mouse points.

[1]Greenbaum & Chartier, *Numerical Methods: Design, Analysis, and Computer Implementation of Algorithms*, Princeton University Press 2012).

The scaling of the grid and/or figure will not matter. At this point my result looked like the figure at right with $n = 36$ points.



(b) Compute and plot the cubic spline interpolant of your data, as a **parameterized curve** $(x(t), y(t))$. The indexing of the points 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. Compute $x(t)$ using the Matlab `interp1()` function, with the last argument as `'spline'`. Do the same thing for $y(t)$. For plotting you will need to generate a fine grid of t values on the interval $[1, n]$. Only plot the (x, y) values in the main figure. Also generate two more figures, smaller is fine,[2] for the functions $x(t)$ and $y(t)$. Make sure to label all axes appropriately. Other than entering the data for the points (x_k, y_k), your Matlab program should only be a few lines.

P9. **(a)** By using a substitution, compute the exact value of the integral

$$\int_{-1}^{1} x^2 \sin(-5x^3 + 1)\, dx.$$

Plot the integrand $f(x) = x^2 \sin(-5x^3 + 1)$ on the interval $[-1, 1]$.

(b) Based on formula (10.6) in Section 10.2, write a composite Simpson's rule code, making it into a convenient function[3] like

```
z = mysimpsons(f,a,b,n)
```

Using the result from **(a)**, compute the numerical error when approximating the same integral for $n = 10, 20, 40, 80, 160, 320$ points. Report this in a small table.

(c) I posted a very short code called `clenshawcurtis.m` on the Codes tab. It is the best possible Matlab implementation of the method described in Section 10.4. Use it to compute the error when approximating the same integral, again for $n = 10, 20, 40, 80, 160, 320$ points.

(d) Use `semilogy` to make a plot of the computed errors for the 2 above methods, in parts **(b)** and **(c)**, versus n. That is, put n on the x axis, and the errors on the y axis, with logarithmic scaling of the errors. Briefly describe what you see, and why it is this way.

---

[2] This is a good application for `subplot`.

[3] My code `mytrap.m` on the Codes tab is a model for how to write your Simpson's method code.