Math 260, Fall 2020
Last updated: 10/2/20

HOMEWORK 6

DUE FRI. OCT. 9

Reading: Ready the "Notes on style and best practicves" posted to Piazza.

For a head start, also skim the GitHub slides and create a repository. You may want to then use it for this homework (optional here; required next week!).

Computational problems. *Note: you may want to refer to the trapezoidal rule example code from class. The 'ratio' calculations aren't required here, as you have exact solutions.*

Q1 (Simpson's rule). Implement Simpson's rule using N subintervals (as `simp(f,a,b,n)`. Then:

a) Test it on the integral

$$\int_0^2 (1-x^2)^2\,dx = \frac{46}{15}$$

using $n = 4, 8, \cdots, 2^M$ sub intervals. Calculate the error and make a log-log plot (use enough points that you get near machine precision for the largest n).

The error behaves like Cn^{-p} for a certain value of p. Identify it from your plot and add a reference line [see e.g. the `trapz.py` lecture code).

b) Now test it on the integral

$$\int_0^\pi \frac{1}{4\sin^2(2x)+1}\,dx \approx 1.404962946208145$$

Use equally spaced n-values $n = 2, 4, 6, \cdots, 200$. Make a log-log and semi-log plot and figure out if the error like Cn^{-p} or like r^n.

Note: 'like' here really means 'bounded by' - there may be variation under this upper bound (e.g. $\sin(x)/x^2$ *is* $O(x^2)$ *but oscillates). You should only look at the error where its above machine precision - you may have to cut n off from the suggested values. It may also help to 'thin out' the n values (skip by more than* 2*)*

Q2 (Adaptive integration). Review the given code for adaptive integration.

a) Using your Simpson's rule code, try computing the integral of

$$f(x) = 5e^{-50x} + \sin(x)$$

from $x = 0$ to $x = \pi$ (get the exact value by any means e.g. Wolfram alpha or by hand). Verify the adaptive method uses less points to get the same error with a tolerance of 10^{-5}.

b) Create an example where the situation is more extreme, e.g. the adpative method is better by a factor of 100 and an example where the adaptive method isn't much better .

Your "main" function should contain the code used to testat least part (b) (this can just be a print of the errors; you don't need plots).

Q3 (derivatives and rounding error). *Note: we did a similar example weeks ago, which you can find in the* `3_numerics` *folder.*

One formula for the third derivative[1] is the following:

$$f'''(x) \approx \frac{f(x+2h) - 2f(x+h) + 2f(x-h) - f(x-2h)}{2h^3}.$$

Suppose you need to approximate $f'''(x_0)$ where

$$x_0 = 1, \qquad f(x) = \cos 2x$$

as accurately as possible.

a) According to theory, the error satisfies

$$E \sim Ch^p \text{ as } h \to 0$$

for a certain value of p. Estimate this value numerically.

b) What, approximately, is the best choice of h and what is the minimum error? (note that you don't have to be very precise here - the right order of magnitude is acceptable). (question to think about: how does this differ from estimating an integral as accuratealy as possible, given your work in Q1?).

Have code that creates a plot or otherwise shows that your answer makes sense (but you can just put your answer in a comment).

c) (optional): When h is small enough, bad things happen. What is the behavior of the error like when h is small (in Big-O terms)? Does this makes sense?

[1] You can derive this by using the formula $(f(x+h) - f(x))/h$ to estimate f', then use the formula to estimate $f''(x) \approx (f'(x+h) - f'(x))/h$ and so on.