Math 563, Spring 2019
Last updated: 1/8/20

HOMEWORK 0
MATH 563, SPRING 2020

DUE MONDAY, JAN. 13

General Note: Some parts (on extensions, open-ended questions, tangents and more) are worth 'additional credit' (AC). You are expected to do some of these parts (about a third) throughout the course, as they often are an opportunity to engage with the material in depth. However, skipping some will not negatively impact your grade.

On preparation: Note that you do **not** need to include the problem statements in your solutions. Using LaTeX is not required but suggested (plots/data are easier in TeX).

On testing code: Your code may be tested against data I create. Thus, the code should work on generic inputs, not hard-coded to work for the given problems. The data, of course, will be similar to what is given (numbers tweaked). Supplying a working example with your code (or noting what doesn't work) helps the testing process.

Reading: Read the following (book sections refer to Quarteroni unless otherwise noted).

- The Guidelines for Computational Work document on Piazza.
- Section 2.4 of Quarteroni (a more detailed version of the short intro from the lecture notes; we'll revisit the themes there throughout the course)
- Complete the office hours poll posted to Piazza

Code to turn in: `newt_polyval` from C1. Submission to Sakai (details to follow).

Problems without code

S1 (Survey). Some questions for the start of the course...

a) Describe, briefly, which programming languages you know and in how much depth. Have you written code for scientific computing?

b) Do you have prior experience with numerical methods (either in courses or in practice)? Is there something you've seen that you'd like to understand better?

c) What do you hope to get out of the course? Are there applications - practical, research or in courses - of particular interest?

Q1 (Some interpolant properties). For all parts, let $x_0, \cdots x_n$ be distinct nodes, let $f_j = f(x_j)$ and let $\ell_j(x)$ be the Lagrange basis polynomial.

a) Let $\ell(x) = (x - x_0)(x - x_1)\cdots(x - x_n)$. What is the interpolating polynomial through the points $x_0, \cdots x_{n-1}$ for $\ell(x)$?

b) Show that Lagrange interpolation is a linear operator. That is, if f_1 and f_2 are functions, the Lagrange interpolant (for given nodes) of $c_1 f_1 + c_2 f_2$ is the same linear combination of the interpolants for f_1 and f_2 (i.e. $p = c_1 p_1 + c_2 p_2$).

c) Show that $\sum_{i=0}^{n} \ell_i(x) = 1$. *Hint: use uniqueness.*

d) (AC) Use part (c) to derive the relative error bound

$$\|p_n(x) - f(x)\|_\infty \leq 2M_n \|f(x)\|_\infty, \quad M_n = \max_x \left(\sum_{i=0}^{n} |\ell_i(x)| \right)$$

where $\|f\|_\infty = \max_{x \in [a,b]} |f(x)|$. *Hint: write* $f = f(x) \cdot 1$ *and replace 1 with the sum in (c).*

Q2 (A uniqueness proof).
a) Suppose f and f' are given at points $x_0, \cdots, x_n$ and let $H(x)$ be the 'Hermite interpolating polynomial' that agrees with f and f' at the nodes. What is its degree? Explain (non-rigorously).[1]

b) Show that the Hermite interpolant from (i) is unique.
Hint: Write $p(x) = (x - x_0)(x - x_1)\cdots(x - x_n)q(x)$ and show that $p = p' = 0$ at $x_0, x_1, \cdots, x_n$.

Q3 (Cubic Hermite interpolant). Suppose you are given the values of a function $f(x)$ and its derivative at $x = 0$ and $x = 1$. A polynomial interpolant matching these values can be constructed, which is the **cubic Hermite interpolating polynomial** (CHIP).

a) Let $p(x)$ and $q(x)$ be the cubic polynomials such that

$$p(0) = p'(0) = 0, \quad p(1) = 1,\ p'(1) = 0,$$
$$q(0) = q'(0) = 0, \quad q(1) = 0,\ q'(1) = 1.$$

Find p and q by writing them as $ax^3 + bx^2 + cx + d$ and solving directly.

b) Find the two other elements of the basis for CHIPs on $[0, 1]$ (with non-zero p or p' at $x = 0$). *Hint: consider* $p(1 - x)$.

[1]Not to be confused with the 'Hermite polynomial', which is something else (we'll get to that later!).

1. Problems with code

We'll have more code write in HW 1; this is just a warm-up of sorts, and a way to test the submission process.

C1. Horner's method for computing a polynomial

$$p(x) = a_0 + a_1 x + \cdots + a_n x^n$$

proceeds as follows. Write in 'nested' form, factoring out x's as much as possible:

$$p(x) = a_0 + x(a_1 + x(a_2 + \cdots x(a_{n-1} + x(a_n))\cdots)).$$

For instance when $n = 3$,

$$p(x) = a_0 + x(a_1 + x(a_2 + x(a_3))).$$

Then the polynomial is evaluated from inside out. For the $n = 3$ case, it would be

$$\begin{aligned} y &\leftarrow a_3, \\ y &\leftarrow a_2 + xy, \\ \vdots &\leftarrow \quad \vdots \end{aligned}$$

(*The problem:*) Suppose you are given the coefficients $c_0, \cdots, c_n$ in an array `c` for the polynomial (which we'll see is another form of the interpolant),

$$\begin{aligned} p(x) &= \sum_{i=0}^{n} c_i \prod_{j=0}^{n-1} (x - x_j) \\ &= c_0 + c_1(x - x_0) + c_2(x - x_0)(x - x_1) + \cdots + c_n(x - x_0)\cdots(x - x_{n-1}). \end{aligned}$$

Adapt Horner's method to this case and write a routine `newt_polyval(nodes,c,xvals)` that evaluates $p(x)$ given nodes $x_0, \cdots, x_n$ at a set of input values `xvals` (returning an array `pvals`).

For example, `newt_polyval([1, 5],[3, 2],[1, e])` would evaluate $p(x) = 3 + 2(x - 1)$ and return `[3, 3 + 2(e-1)]`. Note that the value of the nodes are used up to x_{n-1}.