Numerical Methods for Differential Equations
Homework 2

Due by 2pm Thursday 13 Oct. ***Please submit your hardcopy at the start of lecture.*** *The answers you submit should be attractive, brief, complete, and should include program listings and plots where appropriate. The use of "publish" in Matlab/Octave is one possible approach.*

Problem 1 (Convergence studies). Download `demo_01.m` from the course git repo (or some website).

1. What is the exact solution of the PDE satisfying periodic boundary conditions on $x \in [0, 1)$ with $u_0 = \sin(k\pi x)$ for k even.

2. You might recall in Lecture 1, your instructor was a bit confused[1] by the result when $k = 1$. Discuss the relevance of your solution when k is odd.

3. Back to $k = 2$. Compute the error at $t = 0.25$ (careful with the timestep, don't step too far![2]) Measure the error in the vector max norm ("$\|\cdot\|_\infty$"). Make a table of errors showing that the error decreases like $O(h^2)$ where h is the spatial grid size.

4. Display the error versus h on a log-log plot. Why is this a straight line? What is the slope of the line?

5. *Feeding your inner functional-analyst:* Repeat using the vector 2-norm. What happens? Consider a vector norm that looks like $h^\alpha \|\cdot\|_2$; can you find a reasonable and justifiable value for α? Hint: consider the error as a continuous function and think about the functional L^2 norm. Why could we ignore this for L^∞ and vector max norm?

Problem 2. Construct (by hand or with a computer algebra system) the Lagrange interpolating polynomial for the data

x	0	1	3
f	4	10	0

Make a plot of the resulting interpolant as well as a separate plot of the of the three cardinal polynomials (on the same axes).

Problem 3. If $S \in \Pi_{n-1}$ interpolates f at $x_0, x_1, \ldots, x_{n-1}$ and $T \in \Pi_{n-1}$ interpolates f at $x_1, x_2, \ldots, x_n$, prove that

$$\frac{(x - x_0)T(x) - (x - x_n)S(x)}{x_n - x_0}$$

interpolates f at $x_0, x_1, \ldots, x_{n-1}, x_n$.

Problem 4. Let $w(x) = \prod_{k=0}^{n} (x - x_k)$. Show that the Lagrange interpolating polynomial p_n to f at $x_0, x_1, \ldots, x_n$ can be written as

$$p_n(x) = w(x) \sum_{k=0}^{n} \frac{f(x_k)}{(x - x_k)w'(x_k)}.$$

[1] I know its hard to isolate this to just one instance!

[2] "Say listen, don't you think you're bounding over your steps?"—Stan Laurel, 1932. Watch this for "work": you're welcome.

Find an "clever form of 1" and divide by it, cancel some things and hence derive the *barycentric formula for Lagrange interpolation*:

$$p_n(x) = \frac{\displaystyle\sum_{k=0}^{n} \frac{w_k}{(x-x_k)} f(x_k)}{\displaystyle\sum_{k=0}^{n} \frac{w_k}{(x-x_k)}},$$

where $w_k = \frac{1}{w'(x_k)}$.

Problem 5. In the previous question, can you think of two things you might expect to go wrong when implementing these formulae on a computer? Despite this, it turns out they are completely well-behaved although this was only proven recently [Higham, *The numerical stability of barycentric Lagrange interpolation*, 2004], see also the survey article [Berrut & Trefethen, *Barycentric Lagrange interpolation*, 2004].

Write a software implementation of the barycentric formula for Lagrange interpolation. The inputs to your function should be a vector of points (these are the nodes x_k) and another scalar x (the point at which to interpolate). It should *not* take the $f(x_k)$ as inputs. Your function should return an vector/list of the *weights* β_k so that

$$p_n(x) = \sum_{k=0}^{n} \beta_k f(x_k).$$

Note: your code should work if x happens to be equal to one of the nodes x_k. Use it to evaluate the interpolant of the data given in Problem 2 at $x = 0.32$, $x = 2.999$, and $x = 1$. Display the resulting approximation to $f(x)$ as well as the weights β_k for each case, to at least 12 decimal places.

Problem 6: Git Since Homework 1, the central `math405_2016` repo has probably moved on from your forked: your instructor has hopefully posted new demos and lecture notes. We need to "fast forward" your fork and clone.

1. (Optional, might help with next part). On your clone (on your local computer) figure out how to add a new "remote" called "upstream" and set it to your instructors repo. For example, on the command line it might look like:

   ```
   $ git remote -v
   origin https://gitlab.math.ubc.ca/yourstuff/etc/math405_2016.git (fetch)
   origin https://gitlab.math.ubc.ca/yourstuff/etc/math405_2016.git (push)
   upstream https://gitlab.math.ubc.ca/cbm/math405_2016.git (fetch)
   upstream https://gitlab.math.ubc.ca/cbm/math405_2016.git (push)
   ```

2. Figure out how to "fast forward" your "master" branch (on your local clone) and on your gitlab fork (on the `https://gitlab.math.ubc.ca` website).

3. Attach a screenshot showing that you have the file `05_finite_diff.md` in your local clone.

4. Attach a screenshot showing recent parts of "`git log`" (either from command line or equivalent in a GUI).