Numerical Methods for Differential Equations
Homework 1

Due by 2pm Thursday 21 Sept. ***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. Another option is* `ubc.syzygy.ca`.

Problem 1. The `demo_01.m` programme from lectures solves the PDE $u_t = u_{xx}$ for $t \in [0, T_f]$ and $x \in [0, 1)$ using periodic boundary conditions $u(0) = u(1)$. Download this code from the course git repo (or 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. Say $k = 2$. Using $\Delta t = 0.25h^2$, compute the error at $t = 0.25$ (careful with the timestep, don't step too far![1]) 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.
3. Display the error versus h on a log-log plot for $h \in$ `hset` where `hset = 2.^(-(2:5))` (note m-file notation). Does it seem to be a straight line "in the limit"? Try again with `hset = 2.^(-(3:8))`. What is equation of the best fit line computed by Matlab/Octave's "polyfit" function?

Problem 2. Again with `demo_01.m`.

1. Suppose instead we want to solve $u_t = u_{xx}$ for $u(0) = 3$ and $u(1) = 0$. Modify the code to try to solve this problem. Ensure your grid *excludes* the points 0 and 1.
2. Describe in words where the "3" ends up in your code.
3. Try using the initial condition from Q1. Does this IC satisfy the boundary conditions? What happens in practice?
4. By hand, what is the "steady state" solution $U(x)$ of this differential equation? (That is, the solution in the $U(x) = \lim_{t\to\infty} u(t, x)$.)
5. Plot (as a function of time), the maximum difference between the discrete solution and the steady state. Displaying your results to at least 8 decimal digits, for what value of t (and which discrete time-step n) does this difference first drop below 10^{-5} using $h = 1/20$. Repeat with $h = 1/40$ and $h = 1/80$.

Problem 3. Use Newton's method to find a root of $x^2 - 2$, starting from $x_0 = 2$, showing the values of x_k to 15 digits.

However, it is impossible for Newton to give the exact answer in a finite number of steps. Why? Can you relate this the futility of "circle squaring"?

[1] "Say listen, don't you think you're bounding over your steps?"—Stan Laurel, 1932, *The Music Box*. Take a break and watch this: you're welcome.

Problem 4. Skim this blog post: https://access.redhat.com/blogs/766093/posts/2592591. Try the "`2.2*3.0`" example in Matlab/Octave, and some other programming languages if you have access. What happens? But the article's advice raises my hackles![2]

> The best mitigation is to stick to integer arithmetic whenever possible. The next best approach would be to use the decimal stdlib module which attempts to shield users from annoying details and dangerous flaws.

The vast majority of the time, we do not need to worry about the details, and "dangerous flaws" is just FUD (fear uncertainty and doubt).

In fact, there is really only one "annoying detail": every arithmetic operation makes a *relative error* of size ϵ_{mach} ("machine epsilon"). Based on the author's example, can you estimate what this value is, approximately?

Here's another experiment. In Octave/Matlab or Python, define "`a = 1.0 + 1e-2`". Now check if "`a == 1.0`"; of course not. But to 3 significant figures, what is the largest ϵ you can add to 1 and still get 1 (that is, the largest value of ϵ such that $1 + \epsilon = 1$)? What power of two does that seem to be?

If floating point is so dangerous, then surely our Bisection Method and Newton's Method codes cannot be expected to work... In `02_bisection.m`, how small can you make "`tol`" before some goes wrong? What goes wrong?

Here's an appeal-to-authority: JPL/NASA uses floating point (http://www.jpl.nasa.gov/edu/news/2016/3/16/how-many-decimals-of-pi-do-we-really-need). As they say[3] "trust, but verify": so check the calculation for the Voyager 1 example, reporting your answer to 4 significant figures.

We will study floating point in more detail in a later lecture.

Problem 6: Git Create an account at https://gitlab.math.ubc.ca. Spend some time reading about "Git". You may want to install "SourceTree" or another app in order to use Git on your own machine. On Mac/Linux you can also use command line tools.

Find instructor's notes (written in a combination of Markdown and LaTeX) and demos. Fork this repository. Clone it to your local computer.

Attach a screenshot of your web browser, showing your fork at gitlab.math.ubc.ca.

Attach another screenshot showing the clone on your local computer.

[2] "Someone is wrong on the internet": xkcd.com/386

[3] Obama, Reagan, ... Gorbachev?