Numerical Methods for Differential Equations
Homework 1

Due by 2pm Tuesday 27 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 *is one possible approach.*

Problem 1. 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![1]

> 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[2] "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 2. Use Newton's method to find a root of $x^2 - 2$, starting from say $x_0 = 1$, 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 to the futility in squaring the circle?

Problem 3. Learn about the "Secant Method" and "Brent's method". What is the convergence rate of Brent's Method (give a citation) and how does it compare to Bisection and Newton?

Implement Newton's Method and Brent's Method. Use both to solve for a root of

$$f(x) = 10 \exp(\sin x) - x,$$

in the interval $x \in [5, 7.8]$. Use 7.5 as an initial guess for Newton's Method. Use a tolerance of `1e-13` (on the x value, sometimes called "xtol").

Display the sequence of iterations for each method. How many iterations does each method take? Also, run Newton with an initial guess of $x_0 = 7.8165$. Describe what happens and why, with the help of a plot.

Updated: use 7.5 as initial guess for Newton, and tolerance specified.

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

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

Problem 4: Optimization challenge. Consider the function

$$f(x, y) = x\, e^{-r^2} \cos(3(\theta + r))$$

where $r = \sqrt{x^2 + y^2}$ and $\theta = \tan^{-1}(y/x)$.[3] Consider a unit-area equilateral triangle in $\mathbb{R}^3$. If the vertices of the triangle must lie on the surface $z = f(x, y)$, what is the minimum angle the normal vector of the triangle can make with the x–y plane?

Be sure to explain clearly what you have done and why you think you've found the minimum. Aim for at least six digits of accuracy.

You could use GNU Octave, MATLAB, Ipopt, or some other software. You do not need to write your own optimization code (unless you want to). You could also try the NEOS Server (http://www.neos-server.org/neos) which allows you to upload optimization problems using a web interface.

Problem 5: 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.

[3] Please use caution when computing $\tan^{-1}$: the `atan2` function is your friend.