Notes and Worksheet:

Root-finding as a first example of numerical solutions.

I want to start this course by showing the main purpose of numerical analysis, which is this:

> ***Numerical analysis:*** *Use simple computations, on many numbers, to accurately and quickly solve mathematical problems which you cannot otherwise solve. Understand how the algorithms work, so as to know when they apply, or even to make them better.*

Our textbook[1] starts with *how* one represents real numbers on a computer. While that is an important topic, it is definitely *not* the central theme of numerical analysis. These notes start with a more representative topic, namely root-finding. (This topic reappears in Chapter 4 of the text.)

Numerical methods can be implemented by-hand or with a basic calculator, as people did through the 1960s, but now we expect to program electronic computers to do all the tedious work for us. (Electrical currents running through rocks—silicon chips—doing arithmetic!) Thus some basic computer programming is needed, and these notes get started on that, too.

These notes will ask you to write drafts of computer programs. Over the semester you will get used to writing such drafts, called "pseudocodes". Here "pseudocode" is a synonym for "recipe for doing specific arithmetic or function evaluation on numbers". A pseudocode is effective if a human reader knows what operations to do and in what order. For these notes, please make your best guess about what to put in a pseudocode, and scribble that, and don't worry about details. Please wing it!

During the semester you will also write functional MATLAB programs. Here in these notes I'll help you write MATLAB in parts **G**–**I** at the end.

For all the parts of this worksheet we will work together in class. However, please *write down your own work for all parts* in order to get full participation credit.

1. ROOT-FINDING

Consider the following root-finding problem:

> PROBLEM. Given a continuous real-valued function f of a real variable, find an input r so that $f(r) = 0$.

We call r a "root" or a "solution". Note that a root might not exist or there might be many roots.

A. The function f might have 0, 1, 2, 3, ..., or infinitely-many roots. Make three simple sketches for cases with 0, 2, and infinitely-many solutions, respectively. (*There are many correct solutions.*)

[1] Driscoll & Braun, *Fundamentals of Numerical Computation*, SIAM Press 2018.

Here are two examples of root-finding problems which *cannot be solved by algebra*. We will solve them numerically after we have built some algorithms:

I. Where does the graph $y = e^x$ cross the circle of radius 2 which is centered at the origin?

II. The Kepler equation—see the Wikipedia page for Kepler's equation—relates the mean anomaly M, the eccentric anomaly E, and the eccentricity e:

$$M = E - e \sin E.$$

Here, all you need to know is how this equation is used. For a given elliptical orbit, e.g. of an asteroid around the sun, the eccentricity e is known, and, from the time, one knows the mean anomaly M. Then one solves Kepler's equation for E. Then, from other equations, which you can look-up if you want, the desired position of the asteroid is computed using the E value.

B. Explain using a graph why we expect there to be exactly two solutions to problem *I*. Rewrite the problem in the form $f(x) = 0$. (*Hint.* f is the difference of two functions.) Find a closed interval for x which includes the solution(s) and such that $f(x)$ is defined and continuous.

C. Rewrite problem *II* in the form $f(E) = 0$. Leave M and e as constants. (*Hint.* Easy.)

D. Now thinking more abstractly, suppose f is any continuous function on $[-1, 1]$ for which $f(-1) = -1$ and $f(1) = 1$. Sketch an example. Argue based on mathematical ideas that there is at least one root for any such f.

E. (*The most important part.*) Brainstorm on different methods for how you would solve the root-finding problem for a function as in part **D**. Assume only that f satisfies the conditions in **D** and that the value $f(x)$ is available for each x in $[-1, 1]$. (Do not make any other assumptions about the formula for f. Don't try to find a root by mere algebra.) Your goal is an accurate approximation of a root. Feel free to propose guess-and-check schemes. Try to describe your preferred algorithm using pseudocode, i.e. as a casual/vague computer program with minimal details.

F. Again consider the part **D** problem. One solution strategy starts by looking at the value and sign of $f(0)$ and deciding whether there must be a solution in $[-1, 0]$ or in $[0, 1]$. Then one can look at the value of f in the middle of that interval and split again, and so on. This is called *bisection*. Write a pseudocode for this algorithm.

G. (*I will help with all remaining parts, as we do them in class.*) Turn your pseudocode for bisection (part **F**) into a running MATLAB program `bisection.m` with signature

```
r = bisection(f,a,b)
```

This program will solve the root-finding problem for a continuous function $f(x)$ on an interval $[a, b]$ for which $f(a)$ and $f(b)$ have opposite signs. In the leading comments of the code, put problem *I* as a test example which allows you to check that your program works.

H. Also use `bisection.m` to solve problem *II* for the values $e = 0.7$ and $M = \pi/6$. See part **C** for the form of the function $f(E)$. (*Hint.* A graph of $f(E)$ will show an interval $[a, b]$ on which f changes sign.)

I. There exist *much* faster root-finding methods than bisection. For example there is Newton's method, which you learned in calculus. Do the following steps:

(i) Give a sketch which derives the Newton formula $x_{n+1} = x_n - f(x_n)/f'(x_n)$.
(ii) Write a pseudocode. What is the input? When should you stop?
(iii) Turn the pseudocode into a running MATLAB program `newton.m`.
(iv) Solve problems *I* and *II* using the program.