MATH-131 (Numerical Methods for Scientists and Engineers) — Worksheet 3

Semester: Spring 2019, Instructor: Nicholas Knight

Due Feb. 12 at 2359. *Please remember to cite your sources, including collaborators.*

Deliverable: Submit a Live script titled `worksheet3.mlx` via CatCourses (under Assignments). Divide this file into sections, one for each of the following questions, plus an extra (final) section containing all the function definitions. Document each function definition to explain the input and output arguments.

1. Implement the bisection method (textbook §2.1). Your function should have the following signature,

   ```
   function r = bisection(f, a, b, tol, maxits)
   ```

   where `f` is a function handle and `a`, `b`, `tol`, `maxits`, and `r` are numbers. Your implementation can assume that

   - `-Inf` $<$ `a` $<$ `b` $<$ `Inf`,
   - sign(`f(a)`) $\neq$ sign(`f(b)`)
   - `f` is continuous on [`a`, `b`].
   - `tol` is nonnegative, and
   - `maxits` is a positive integer.

   Your implementation should estimate the absolute error as one-half the width of the current search interval, and terminate once this estimate no longer exceeds `tol`. Additionally, your implementation should perform at most `maxits` iterations, and should report a warning to the user if your error estimate still exceeds `tol` after `maxits` iterations. (This matches the functionality in the textbook's implementation, Algorithm 2.1.)

   (a) Test your code by using it to approximate the root π of $x \mapsto \sin(x)$. (Pick `a` and `b` to 'bracket' this root.) Find values of `tol` and `maxits` so that the six leading digits of `r` are correct (3.14159).

   (b) Use your code to try to find a root of $x \mapsto \tan(x)$ on the interval $[1, 2]$. Explain what happens.

   (c) *Challenge (optional/ungraded)* Implement bisection using recursion (vs. iteration).

   (d) *Challenge (optional/ungraded* The textbook claims that it is preferable (in floating point) to compute the midpoint as `c = a+(b-a)/2`, rather than `c = (a+b)/2`. Find an interval [`a`, `b`] where this makes a difference, and explain what might go wrong when using the latter formula.

2. Read `doc fzero` to learn how to use MATLAB's `fzero` routine. This is an accelerated version of bisection called the Brent-Dekker algorithm, not covered in the textbook (but referenced briefly in §2.7.)

   (a) Try it out on the previous example: `fzero(@sin, [a b])`, replacing `a` and `b` with the ones you used. How many digits are correct?

   (b) Try out `fzero`'s `Display` option:

   ```
   fzero(@tan, [1 2])
   fzero(@tan, [1 2], optimset('Display','iter'))
   ```

   (*Note:* the second command may help you complete Question 1(b).)

   (c) Try out `fzero`'s `PlotFcns` option:

   ```
   fzero(@sin, [3 4], optimset('PlotFcns',@optimplotfval));
   ```

   (d) Add an optional sixth input argument to your function `bisection`, so that

   ```
   bisection(f, a, b, tol, maxits, 'iter')
   ```

   outputs all approximations to the root, not just the last one — see Part (b) — and so that

   ```
   bisection(f, a, b, tol, maxits, 'plot')
   ```

   generates a plot of the iteration history — see Part (c). (You'll need to use `varargin` and `strcmp`.)