PHYS 410: Computational Physics Fall 2019
Homework 1
Due: Monday, September 23, 11:59 PM

PLEASE report all bugs, comments, gripes etc. to Matt: `choptuik@physics.ubc.ca`

Problem 1

Implement a hybrid algorithm that uses bisection and Newton's method to locate a root within a given interval $[x_{\min}, x_{\max}]$. Assuming that you code in MATLAB, your top-level algorithm should be implemented as a function with the header

```
function x = hybrid(f, dfdx, xmin, xmax, tol1, tol2)
```

where the arguments to the routine are defined as follows:

```
%  f:    Function whose root is sought.
%  dfdx: Derivative function.
%  xmin: Initial bracket minimum.
%  xmax: Initial bracket maximum.
%  tol1: Relative convergence criterion for bisection.
%  tol2: Relative convergence criterion for Newton iteration.
```

The single output argument is given by

```
%  x:    Estimate of root.
```

Given the initial bracket (interval) $[x_{\min}, x_{\max}]$ such that

$$f(x_{\min})f(x_{\max}) < 0$$

your implementation should perform bisection until the root has been localized to a *relative* accuracy of `tol1`. Your code should then perform Newton iterations until the root has been determined to a *relative* tolerance of `tol2`.

Note that in MATLAB functions can be passed to other functions as arguments (e.g. `f` and `dfdx` above) using *function handles*, as in the following:

```
function fx = f(x)
   fx = cos(x)^2;
end

function val = caller(f, x)
   val = f(x);
end

result = caller(@f, 2.0)
```

Here, `result` will be assigned the value $\cos(2)^2$. In brief, to pass a function to another function, simply prepend a `@` to the function name in the argument list.

Test your implementation by determining all roots of the function

$$f(x) = 128x^8 - 256x^6 + 160x^4 - 32x^2 + 1$$

in the interval $[-1, 1]$.

I leave it to you to determine how to choose the initial intervals for `hybrid`, but a brute force approach will suffice. Also, your solution may comprise more than one function—i.e. more functions than `hybrid` alone.

Problem 2

Implement a d-dimensional Newton iteration. Again, assuming that you are coding in MATLAB, your implementation should be in the form of a function with header

```
function x = newtond(f, jac, x0, tol)
```

where the input arguments are defined by

```
% f:    Function which implements the nonlinear system of equations.
%       Function is of the form f(x) where x is a length-d vector, and
%       returns length-d column vector.
% jac:  Function which is of the form jac(x) where x is a length-d vector, and
%       which returns the d x d matrix of Jacobian matrix elements.
% x0:   Initial estimate for iteration (length-d column vector).
% tol:  Convergence criterion: routine returns when relative magnitude
%       of update from iteration to iteration is <= tol.
```

and the output argument is

```
% x:    Estimate of root (length-d column vector)
```

Use your implementation to find a root of the system

$$x^2 + y^3 + z^4 = 1$$

$$\sin(xyz) = x + y + z$$

$$x = yz$$

in the vicinity of $(x, y, z) = (3, -2, -1)$.