PHYS 410: Computational Physics Fall 2024
Homework 1
Due: Wednesday, October 2, 11:59 PM
PLEASE report all bugs, comments, gripes etc. to Matt: `choptuik@physics.ubc.ca`

General Notes

1. Refer to `https://laplace.phas.ubc.ca/410` and `https://laplace.phas.ubc.ca/410/Homework.html` for general information concerning homeworks/projects and the writeups that must accompany the submission of each.
2. Your solution to each problem must include at least one script which, when run, will call the functions you write to generate your results: i.e. the TA should have to do nothing more than type the name of a script in order to reproduce the results you describe in your writeup. Ensure that you include the names of all such scripts in your writeup.
3. Your functions and scripts do not have to be extensively documented. For functions, an initial comment block that describes the input and output arguments is recommended, and you are free to use the comments below for that purpose.
4. Roots should be calculated to a minimum of 10^{-12} *relative precision*.
5. The grading rubric weights 40% of the grade to elements related to the writeup for the homework, and 60% for elements related to the actual code.
6. Recall that you are welcome to discuss this and other assignments with your classmates, but that *any work you submit, including your writeup and any and all source code, must be original to you.*

Problem 1

Implement a hybrid algorithm that uses bisection and Newton's method to locate a root within a given interval $[x_{\min}, x_{\max}]$. 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 (takes one argument).
%  dfdx: Derivative function (takes one argument).
%  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`. Use the last root estimate from the bisection method as the initial estimate for the Newton algorithm.

The functions `f` and `dfdx` are to be hard coded for each separate nonlinear function $f(x)$; i.e. do *not* attempt to write some general routines using, e.g., MATLAB's symbolic manipulation capabilities.

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(some_fcn, x)
   val = some_fcn(x);
end

>> result = caller(@f, 2.0)

result =

    0.1732
```

Here, `@f` is a function handle and `result` is 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

$$f(x) = 512x^{10} - 5120x^9 + 21760x^8 - 51200x^7 + 72800x^6 - 64064x^5 + 34320x^4 - 10560x^3 + 1650x^2 - 100x + 1$$

in the interval $[0, 2]$.

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. Your implementation should be in the form of a function with MATLAB 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
%        which 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 for the
%        nonlinear system defined by f.
%  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

$$x^2 + y^4 + z^6 = 2$$

$$\cos(xyz^2) = x + y + z$$

$$y^2 + z^3 = (x + y - z)^2$$

in the vicinity of $\mathbf{x}^{(0)} = (x, y, z) = (-1.00, 0.75, 1.50)$.

As for the first question, the functions `f` and `jac` are to be hard-coded for the specific nonlinear system given above.