This assignment requires you to implement Newton’s method and bisection for rootfinding in Matlab and try them on several functions. You’ll need to pass function references to Matlab. We’ll discuss this in class, and you can get more information by typing “help feval” in Matlab.

1. Write a Matlab function `newton.m` to implement Newton’s method. Use the following lines as the first lines of your code; they describe the input and output arguments you should use for your function, as well as the various termination tests:

   ```matlab
   function [x,ierr] = newton(f, fd, x0, epsi, delta, maxf, maxstep)
   % Newton’s method for evaluating a root of the function
   % whose derivative is given in fd.
   %
   % calling sequence: [x,ierr] = newton(@f, @fd, x0, epsi, delta, maxf, maxstep)
   %
   % f = name of the matlab function that evaluates f
   % fd = name of the matlab function that evaluates f’
   % x0 = initial guess for the root x
   % epsi = terminate (with ierr=0) if two successive x values differ by less than this amount
   % delta = terminate (with ierr=0) if |f| drops below this value
   % maxf = maximum number of iterations; terminate with ierr=2 if exceeded
   % maxstep = terminate (with ierr=1) if step is longer than this value
   %
   % the routine prints out iteration number, x, and f at each iteration
   %
   % You will find Matlab functions fq1.m and fdq1.m which evaluate the function f(x) = e^x - 3x^2 and its derivative on the course web site. Write a Matlab code q1.m that calls your newton routine to find roots of this function, starting at the points x_0 = -0.5 and x_0 = 4. (Choose appropriate values for the other input arguments.) Your code q1.m should print out the final values of x and ierr returned by your Newton routine.

2. You will find Matlab functions fq2.m and fdq2.m to evaluate the function f(x) = tan x - x and its derivative on the course web site. Write code q2.m that calls your Newton routine to find roots of this function, starting from the values x_0 = 0.1 and x_0 = 4.9, and two other starting points of your own choosing. Comment on the performance of the Newton code on this example—its rate of convergence, whether it terminated with an
approximate root or an error, etc. (It may help if you draw graphs of \( \tan x \) and \( x \), to figure out approximately where the true roots of \( f(x) \) lie.)

3. Write a Matlab function to implement the bisection algorithm, using the following as the initial lines of your code:

```matlab
function [x,ierr] = bisection(f, a, b, tol, tolf)
%
% use bisection to calculate a root of f.
%
% calling sequence: [x,ierr] = bisection(@f, a, b, tol)
%
% f = name of the matlab function to evaluate function f
% a,b = initial endpoints of the interval. Must have a<b, with
%       f(a) and f(b) different signs (otherwise return ierr=1)
% tol = terminate when length of interval falls below this value
% tolf = terminate if |f(x)| falls below this value, where x is some
%       point evaluated during the algorithm
%
% return x = final estimate of the root.
% ierr = error code (=0 if an approximate root is found)
%
Write a code q3.m that tests your bisection code on the function \( f(x) = \tan x - x \) defined in question 2, calling with \( a = 3.5 \) and \( b = 4.5 \) and \( tol \) and \( tolf \) both set to \( 10^{-6} \). Print out the final values of \( x \) and \( ierr \) returned by your bisection code.

Hand in hard copies of your codes newton.m, bisection.m, q1.m, q2.m, and q3.m and output together with your written answers. Put your codes in a directory called homework2. From the parent directory of homework2, run the following command:

```bash
handin -c cs416-1 -a hwk2 -d homework2
```