Name: ______________________________

1. (15 points) Let $f(x)$ be written as

$$f(x) = (x - x^*)^m q(x), \quad q(x^*) \neq 0,$$

where x^* is a **root of multiplicity** $m > 1$. In such a case scenario, Newton's method may converge when $m > 1$ but **not** quadratically. Note also that Newton's method can be written as $x_{n+1} = g(x_n)$ where

$$g(x) = x - \frac{f(x)}{f'(x)},$$

which is called the **iteration function**.

(a) (5 points) Write out the **iteration function** $g(x)$ for Newton's method in this case (it will involve $q(x)$ and $q'(x)$).

(b) (10 points) Show that $g'(x^*) = 1 - 1/m \neq 0$, and **explain why** this implies only **linear** convergence of Newton's method in the case of multiple roots.

Solution:

(a) From Newton's method written as $x_{k+1} = g(x_k)$ with $k = 0, 1, \ldots,$ we know that

$$g(x) = x - \frac{f(x)}{f'(x)} \Rightarrow g(x) = x - \frac{(x - x^*)^m q(x)}{m(x - x^*)^{m-1} q(x) + (x - x^*)^m q'(x)},$$

or, by simplifying the fraction therein, we arrive at

$$\boxed{g(x) = x - \frac{(x - x^*) q(x)}{m q(x) + (x - x^*) q'(x)}}. \tag{1}$$

(b) Next, we take the derivative of Eq. (1) with respect to x and obtain (after some algebra)

$$g'(x) = 1 - \frac{mq^2(x) + (x - x^*)^2 q'^2(x) - (x - x^*)^2 q(x) q''(x)}{\left[mq(x) + (x - x^*) q'(x)\right]^2}. \tag{2}$$

Thus, the evaluation of Eq. (2) at $x = x^*$ yields to

$$g'(x^*) = 1 - \frac{1}{m} \neq 0, \tag{3}$$

since $m > 1$. Note also, that $q(x^*) \neq 0$.

Finally, and based on Eq. (3), we conclude that Newton's method converges **linearly**. To see this, we perform a Taylor expansion of $g(x)$ about x^*

$$\begin{aligned}
g(x) &= g(x^*) + g'(x^*)(x - x^*) + \frac{g''(\xi)}{2}(x - x^*)^2 \xRightarrow{(x = x_k)} \\
x_{k+1} &= x^* + g'(x^*)(x_k - x^*) + \frac{g''(\xi)}{2}(x_k - x^*)^2 \Rightarrow \\
\frac{x_{k+1} - x^*}{x_k - x^*} &= g'(x^*) + \frac{g''(\xi)}{2}(x_k - x^*) \Rightarrow \\
\lim_{k \to \infty} \frac{|x_{k+1} - x^*|}{|x_k - x^*|} &= |g'(x^*)|,
\end{aligned}$$

or, via Eq. (3)

$$\boxed{\lim_{k \to \infty} \frac{|x_{k+1} - x^*|}{|x_k - x^*|} = \left|1 - \frac{1}{m}\right|}. \tag{4}$$

The latter equation reveals that the convergence factor is given by $\rho = |1 - \frac{1}{m}|$ which is **less** than 1, $\forall m > 1$, thus, convergence of Newton's method **is guaranteed**. Furthermore, the rate of convergence is 1 which signals that Newton's method will converge **linearly** in this case.

2. (25 points) Show that in the case of a root of multiplicity m, the **modified Newton's method** given by

$$x_{n+1} = x_n - m \frac{f(x_n)}{f'(x_n)},$$

is quadratically convergent.

Solution: Assume that $m > 1$, so there is a multiple root, say x^*, of multiplicity m. From Question 1, we have

$$f(x) = (x - x^*)^m q(x),$$

where $q(x^*) \neq 0$ and

$$g(x_n) = x_n - m\frac{f(x_n)}{f'(x_n)},$$

and denote $e_n = x_n - x^*$. Taylor expanding leads to

$$\begin{aligned} e_{n+1} &= x_{n+1} - x^* \\ &= g(x_n) - x^* \\ &\approx g(x^*) + g'(x^*)e_n + \frac{g''(x^*)e_n^2}{2} - x^* \\ &\approx g'(x^*)e_n + \frac{g''(x^*)e_n^2}{2}, \end{aligned}$$

with $g(x^*) = x^*$. Since $g'(x^*) = 0$ (see, Eqs. (2) and (3) of Question 1 and multiply the fraction appearing therein with m) then

$$e_{n+1} = \frac{g''(x^*)e_n^2}{2}.$$

Since the error is approximately squared at each iteration, the convergence **is** quadratic provided that $g''(x^*) \neq 0$. Indeed, direct computation of $g''(x)$ yields

$$\begin{aligned} g''(x) = \frac{1}{(mq(x) + (x - x^*)q'(x))^3} \times \Big\{ &(x - x^*)^2 q'(x)^2 \left[2mq'(x) + (x - x^*)q''(x)\right] \\ &+ mq(x)^2 \left[2q'(x) + (x - x^*)\left(q'''(x)(x - x^*) + 4q''(x)\right)\right] + q(x)(x - x^*)\times \\ &\left[-4mq'(x)^2 + (x - x^*)q'(x)\left(q'''(x)(x - x^*) - 3mq''(x)\right) - 2(x - x^*)^2 q''(x)^2\right] \Big\}, \end{aligned}$$

and thus

$$g''(x^*) = \frac{2q'(x^*)}{m^2 q(x^*)} \neq 0,$$

since $q(x^*) \neq 0$ from the original assumption. However, note that this condition imposes that $q'(x^*) \neq 0$ (otherwise, if $q'(x^*) = 0$, Newton's method would converge **at least cubically**!).

3. (15 points) Suppose

$$x + \ln x = 0, \quad x > 0.$$

Implement the **secant method** in MATLAB (or in **any** programming language) and **find the root** of the above equation. Use $x_0 = 0.5$, $x_1 = 0.6$ and $|x_{k+1} - x_k| < 10^{-10}$ as a convergence criterion. In addition, use your function from Question 3 employing Newton's method and repeat the calculation with same initial guess x_0 and convergence criterion as before. Attach your code for the secant method and provide MATLAB outputs for both cases. **Which method converges faster?** Briefly explain.

Solution: An implementation of secant's method is given in the following script named as "secant.m":

```
function [ x, k, fout ] = secant( func, x0, x1, atol, nmax )
%
% On input:
%           a)  func: is the nonlinear function f(x).
%           b)   x0 : initial guess.
%           c)   x1 :        >>    .
%           d)  atol : absolute tolerance.
%           e)  nmax : maximum number of iterations allowed.
%
% On output:
%           a)    x : is the root of f(x).
%           b)    k : number of iterations required to achieve atol.
%           c)  fout : residuals stored at each k.

       k = 0;                         % Iteration index.
   iflag = 0;                         % For stopping purposes.
      xk = rand;                      % Just to start the while loop.
     res = xk - x0;                   % Compute the absolute error.
     fx0 = feval(func,x0);            % Evaluate the function at x_{0}.
     fx1 = feval(func,x1);            % Evaluate the function at x_{1}.
     while abs(res)>atol              % Do-while loop.
        fout(k+1) = fx0;              % Store the residual in each step.
          xk = x1 - fx1 *     ...
                ( x1 - x0 ) / ...
               ( fx1 - fx0 );         % Secant method.
          res = xk - x0;              % Compute the "new" absolute ...
              error.
          x0 = x1;                    % Substitutions.
          x1 = xk;                    %      >>      .
         fx0 = feval(func,x0);        % Evaluate the function at x_{0}.
         fx1 = feval(func,x1);        % Evaluate the function at x_{1}.
         k = k + 1;                   % Increase iteration index.
         if(k == nmax)                % Check whether we reached the
             iflag = -1;              % maximum number of iterations!
```

```
                break;                % If yes, then we stop secant.
            end
        end
        if(iflag==0)                  % On successful exit,
       fout(k) = feval(func,xk);      % return the root!
       xout(k) = xk;
            x = xk;
        else
            disp(' maximum number of iterations reached!');
            x = [];
            k = [];
        end
    end
```

Then, the main driver consists of the following code which calls "secant.m":

```
clearvars; close all; clc; format long;
   func = @(x) x+log(x);
     x0 = 0.5; x1 = 0.6; atol = 1.e-10; nmax = 100;
 [ xstar, k, fout ] = secant( func, x0, x1, atol, nmax );
```

and the corresponding MATLAB output follows:

```
        x_{k}                   f(x_{k})
------------------------------------------------------------
  0.500000000000000  -0.193147180559945e+00
  0.600000000000000   0.089174376234009e+00
  0.568413897526397   0.003508464725863e+00
  0.567120282313471  -6.357732002681971e-05
  0.567143306843229   4.540927101004399e-08
  0.567143290409997   5.876410469340954e-13
  0.567143290409784  -1.110223024625157e-16
```

On the other hand, the MATLAB output using Newton's method follows:

```
         x_{k}                   f(x_{k})
------------------------------------------------------------
  0.500000000000000  -0.193147180559945e+00
  0.564382393519982  -0.007640861009083e+00
  0.567138987715060  -1.188933308848839e-05
  0.567143290399369  -2.877842408821607e-11
  0.567143290409784  -1.110223024625157e-16
```

From the above results it can be discerned that Newton's method converges faster. In particular, 7 iterations are required for secant method while 5 ones for Newton's one. Note that both methods converged within *machine precision*!

4. (15 points) Assume the following **fixed point iterations** ($x_{k+1} = g(x_k)$):
   (a) (5 points) $x_{k+1} = -16 + 6x_k + \frac{12}{x_k}$ with $x^* = 2$
   (b) (5 points) $x_{k+1} = \frac{2}{3}x_k + \frac{1}{x_k^2}$ with $x^* = 3^{1/3}$
   (c) (5 points) $x_{k+1} = \frac{12}{1+x_k}$ with $x^* = 3$

   Note that x^* above corresponds to the respective **fixed point**.

   Then, which of the above iterations **will converge** to the fixed point x^* indicated above, provided that $x_0 \approx x^*$, i.e., the initial iterate x_0 is sufficiently close to x^*? If it does converge, then find the order of convergence.

Solution:

(a) Here, we have that $g(x) = -16+6x+\frac{12}{x}$ and $g(2) = 2$, thus, x^* is indeed a fixed point. Furthermore, $g'(x) = 6 - \frac{12}{x^2}$ and this way, $g'(2) = 3$. Since, $|g'(x)| < 1$ is required, we conclude that for given x_0 the iteration is **not** guaranteed to converge.

(b) In this case, $g(x) = \frac{2}{3}x + \frac{1}{x^2}$ and $g(3^{1/3}) = 3^{1/3}$, so, x^* is a fixed point. Next, $g'(x) = \frac{2}{3} - \frac{2}{x^3}$ leading to $g'(3^{1/3}) = 0$. The latter suggests *at least* quadratic convergence. To examine this finding, let us compute $g''(x) = \frac{6}{x^4}$ where $g''(3^{1/3}) \neq 0$. Thus, we conclude that for $x_0 \approx x^*$, the order of convergence **is** quadratic.

(c) Finally, $g(x) = \frac{12}{1+x}$ and indeed $g(3) = 3$ holds, suggesting that $x^* = 3$ is a fixed point. Subsequently, $g'(x) = -\frac{12}{(1+x)^2}$ and $g'(3) = -\frac{3}{4} \neq 0$. Furthermore, note that $|g'(3)| < 1$ which itself suggests that the iteration will converge **linearly** in this case.

5. (30 points) Write a MATLAB script which can identify the **three roots** of

$$e^x - 2x^2 = 0,$$

   using **fixed-point** iterations with $|x_{k+1} - x_k| < 10^{-10}$ as a convergence criterion. Note that plotting will help here. Furthermore, **explain** your choices for the $g(x)$ utilized in order to ensure convergence and attach your MATLAB script.

Solution: The graph of the function $f(x) = e^x - 2x^2$ is shown in Fig. 1. We immediately notice the existence of three zeros of $f(x)$. Since we want to apply fixed point iterations of the form of $x_{k+1} = g(x_k)$, we should choose the function $g(x)$ such that $|g'(x)| < 1$, that is, convergence is guaranteed on the respective intervals. To this end, we notice that some possible choices of $g(x)$ follow

$$e^x - 2x^2 = 0 \Rightarrow x = \ln(2x^2), \quad \text{and} \tag{5}$$
$$e^x - 2x^2 = 0 \Rightarrow x^2 = \frac{1}{2}e^x \Rightarrow x = \pm\frac{1}{\sqrt{2}}e^{x/2}. \tag{6}$$

Furthermore, let us focus on the interval $[-4, 4]$. Thus, we have the following outcomes:

1. $g(x) = \ln(2x^2)$ and $|g'(x)| = |2/x| < 1$ holds $\forall x \in [2.1, 4]$. This way, if $x_0 = 3$, we obtain $x^* = 2.617866613357755$.
2. $g(x) = \frac{1}{\sqrt{2}}e^{x/2}$ where $|g'(x)| = \left|\frac{1}{2\sqrt{2}}e^{x/2}\right| < 1$ holds $\forall x \in [-4, 2]$. Firing up the fixed point method for $x_0 = 2$ we obtain $x^* = 1.487962065730103$.
3. Finally, to trace the *negative* solution, we use Eq. (6) with the *minus* sign, that is, $g(x) = -\frac{1}{\sqrt{2}}e^{x/2}$, where the same convergence criterion holds as before. This way, we obtain $x^* = -0.539835276909246$ for $x_0 = -3$.

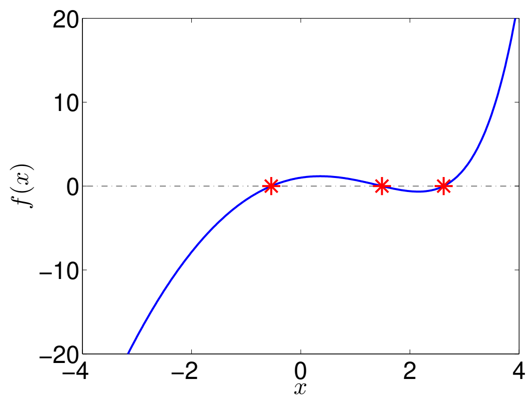

Figure 1: The graph of the function $f(x)$ of Question 5. Note that there exist three roots which are shown with red stars therein.

The three roots that we found are plotted in Fig. 1 with red stars and the MATLAB code that was used is shown next

```
clearvars; close all; clc; format long;
```

```

% Initial guess and iteration function:
     x0 = 3;
  gfunc = @(x) log(2*x.^2);
%  gfunc = exp(x/2)/sqrt(2);
%  gfunc = -exp(x/2)/sqrt(2);

% Fixed-point setup:
nmax = 100;
 tol = 10^-10;
for i = 1:nmax
    x1 = gfunc(x0);
 error = abs(x1-x0);
    x0 = x1;
    if(error<tol)
        break;
    end
    if(i==nmax)
        disp('maximum number of iterations reached!');
        pause;
    end
end
x0
```

Date: January 30, 2020

Mathematics Department, California Polytechnic State University, San Luis Obispo, CA 93407-0403, USA
Email address: echarala@calpoly.edu