```
% % Nicole Deyerl
% % MATH 6321 (Dan Reynolds)
% % 9/2/16
% % Homework 1, Problem 2 (Computational)
% % This script finds the roots of the lagrange interpolant of the points given
% % by problem 2 of homework 1.  It’s a script which uses Newton’s method to find
% % the roots of this particular function.

clear;
% lagrange interpolant of 4 data points
f = @(x) (-4/5 + 4/495 + 8/9)*x.^3 + (-28/5+8/495 + 52/9)*x.^2 +...
    (-31/5 + 13/1485 + 116/27)*x + (-119/135 + 14/13365 + 136/243);
% derivative of lagrange interpolant
fprime = @(x) 3*(-4/5 + 4/495 + 8/9)*x.^2 + 2*(-28/5+8/495 + 52/9)*x +...
    (-31/5 + 13/1485 + 116/27);

% Newton’s method (reference text: Numerical Analysis by Sauer from
% previous course)
n = 10; %number of steps to use in newton solver
x = zeros(1,n+1); %set up solution vector

% at most 3 roots -> 3 Newton solves with 3 initial guesses x0 = x(1)
x(1) = -1; % "x0"
for i = 1:n % start with "x0"
    x(i+1) = x(i) - f(x(i))/fprime(x(i)); % newton’s method
end

fprintf(’The first root is %d \n’,x(n+1));

x(1) = -3;% "x0"
for i = 1:n
    x(i+1) = x(i) - f(x(i))/fprime(x(i));
end

fprintf(’The second root is %d \n’,x(n+1));

x(1) = -10;% "x0"
for i = 1:n
    x(i+1) = x(i) - f(x(i))/fprime(x(i));
end

fprintf(’The third root is %d \n’,x(n+1));
```

```
% % Nicole Deyerl
% % MATH 6321 (Dan Reynolds)
% % 9/2/16
% % Homework 1, Problem 3
% % This script finds the root of a nonlinear system of equations given by
% % homework 1.  It uses a relative tolerance of 10^06 and an absolute tolerance
% % of 10^-10, with an initial guess of (1,2).
% %
clear;

% functions f1 and f2
f1 = @(x,y) x.^2 + y.^2 -4;
f2 = @(x,y) x*y - 1;

% Jacobian functions
Df1 = @(x,y) 2*x;
Df2 = @(x,y) 2*y;
Df3 = @(x,y) y;
Df4 = @(x,y) x;

n = 10;
norms = 1; % initialize error tolerance
normx = 1; %initialize relative error denominator
x = zeros(2,n); % solution matrix (made assuming while loop doesnt take more
                  % than n iterations
p = zeros(2,1); % intermediate solution vector
f = zeros(2,1); % function value vector
Df = zeros(2,2); % jacobian value vector

x(:,1) = [1,2]; % initial condition "x0"=(1,2)


for i=1:n
    f = [ f1(x(1,i),x(2,i)); f2(x(1,i),x(2,i))]; % vector f(x(i))
    Df = [Df1(x(1,i),x(2,i)), Df2(x(1,i),x(2,i)); Df3(x(1,i),x(2,i)), ...
        Df4(x(1,i),x(2,i))]; % matrix Df(x(i))
    p(:,1) = Df\(-f); % solve Df*p =-f
    x(:,i+1) = x(:,i) + p(:,1); % solve for next x-value

    norms = max(abs(x(1,i+1)-x(1,i)),abs((x(2,i+1)-x(2,i)))); % diff between steps
    normx = max(abs(x(1,i)),abs(x(2,i))); % magnitude of previous step

    if(norms <= (10e-10)) && (norms <= (10e-6)*normx) % exit loop when converg./error
        break;                                         % threshold is met
    end

end

% print solution
fprintf('--------------------\n');
fprintf('Absolute tolerance threshold = 10e-10\nRelative tolerance threshold = 10e-6\n');
fprintf('final atol = %.5e  final rtol = %.5e\n',norms,norms*normx);
fprintf('soln = (x,y) = (%.13d, %.13d)\n',x(1,i),x(2,i)); %x0 has index 1, + i-1 time
                                                           %steps -> soln has index i
```