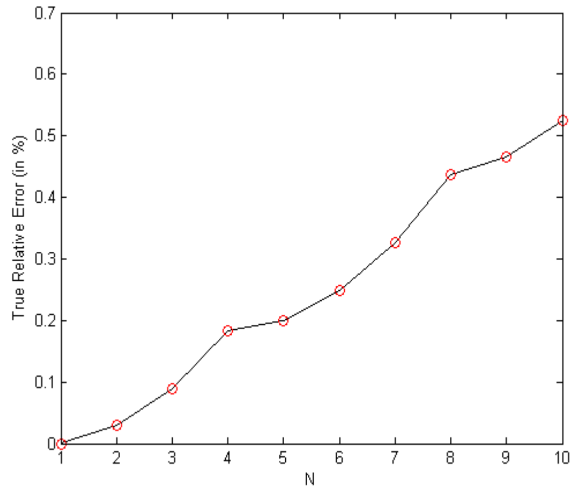
MAE384 HW1 Discussion

Prob. 1

We have already discussed this problem in some detail in class. The only point of note is that chopping, instead of rounding, should be applied to every step of the calculation when using the toy calculator. Those who applied rounding would find the error randomly fluctuating up and down. With chopping, the true relative error grows monotonically with N. Attached is a plot of TRE vs. N.



Prob. 2

The answer is 10010100011111, in binary.

Prob 3

There is only one positive solution for this equation. See attached sample solution version 1 for a solution by hand, and version 2 for a relevant Matlab code.

Prob 4

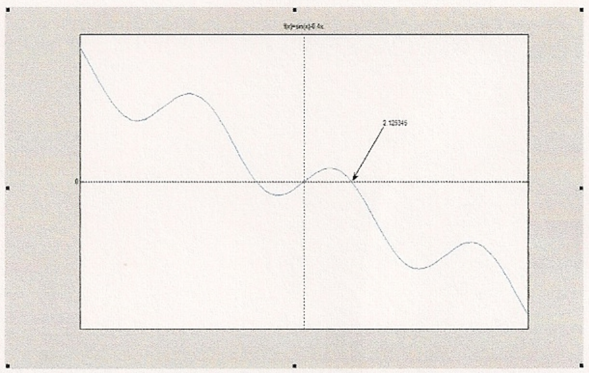
The equation has three solutions, $x = -2.9922...$, $x = 0.60898...$, and $x = 2.10935...$ Three versions of sample solutions are attached. Version 1 obtained the solutions by hand. Version 2 is included for its detailed discussion. Version 3 provides relatively simple Matlab codes, using only the commands that have been discussed in the class. (This is for those who find the codes in Version 2 mysterious.)

Prob 4, further discussion: Choosing an initial guess that's close to the desired solution usually leads to convergence into that solution. Since the initial "slope", $f'(x_1)$, where x_1 is the initial guess, determines the direction that the initial guess will move into, the points where $f'(x) = 0$ approximately mark the boundaries of the "basins of attraction"; Initial guesses that fall within the same "basin" will converge to the same solution (which itself is located within the basin). Since $f'(x) = 0$ at $x \approx -1.92$ and 1.37, we know approximately that an initial guess with $x_1 < -1.92$ leads to the solution of $x = -2.9922$; Initial guesses with $-1.92 < x_1 < 1.37$ lead to $x = 0.60898$, and those with $x_1 > 1.37$ lead to $x = 2.10935$. It's important to note that the demarcation of the basin of attraction obtained in this simple manner is only approximate but not exact. In the vicinity of where $f'(x) = 0$, Newton's method becomes nearly singular (it blows up if $f'(x_1) = 0$) and the outcome of the iterative process can be surprising. Indeed, in version 2 of our sample solution, we find "surprises" with the initial guesses $x_1 = -1.9, -1.8, -1.7, -1.6,$ and -1.5 (which are located in the vicinity of $x = -1.92$), and another surprise with $x_1 = 1.3$ (located in the vicinity of $x = 1.37$). This behavior is very common to Newton's method. (Recall the beautiful "fractal" picture for the solution of $z^3 - 1 = 0$ that we briefly discussed in class.)

Solution, Prob 3 version 1 (by hand) Thanks to Jason Kmon

3.

First the equation was rewritten in terms of *f(x)*. The function was then plotted in *Matlab* in order to determine how many positive non-zero solutions were present. According to the graph there was only one solution that met the criteria. When using the bisection method: an interval around the zero of the function is selected, then a midpoint of that interval is found, the half of the segment that the zero is on then becomes the next set of boundaries. The process is continued until the midpoint value comes within the acceptable range of the exact answer (±0.05).

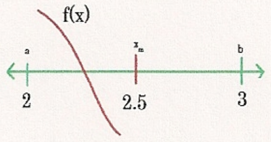


$$\sin(x) = 0.4x$$

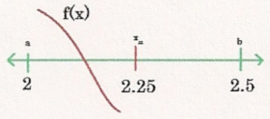
$$f(x) = \sin(x) - 0.4x \quad \Rightarrow \quad [2,3] \text{ interval}$$

$$f(x) = 0$$

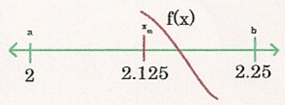
$$x_m = \left(\frac{a+b}{2}\right) \qquad f(a)\times f(b) < 0 \quad x_m = b \qquad f(a)\times f(b) > 0 \quad x_m = a$$



$$x_1 = \left(\frac{2+3}{2}\right) = 2.5 \qquad f(2) = 0.1093 \qquad f(3) = -0.4013 \qquad f(2)\times f(3) < 0 \therefore x_m = b$$



$$x_2 = \left(\frac{2+2.5}{2}\right) = 2.25 \quad f(2) = 0.1093 \qquad f(2.25) = -0.1219 \quad f(2)\times f(3) < 0 \therefore x_m = b$$



$$\boxed{x_3 = \left(\frac{2+2.25}{2}\right) = 2.125 \approx 2.125345}$$

Solution, Prob 3 version 2 (Matlab code by HPH)

The following is a barebone program that chooses [1, 3] as the initial interval and performs 5 iterations. The number of iteration (5) is pre-determined by the consideration that the length of the interval is halved after each iteration. Solving $2/2^N < 0.1$ yields the minimum value of N as 5. (The "0.1" here is 2 x 0.05, since we allow +/- 0.05 as the tolerance.) Note that at the end of the process we pick the mid-point as the final solution.

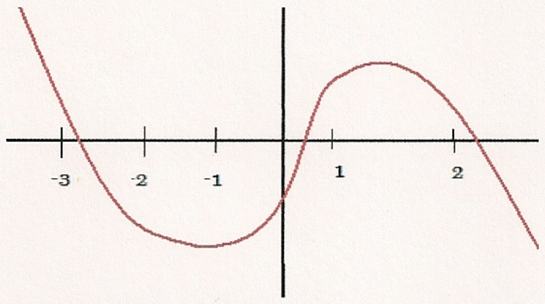
```
xleft = 1;
xright = 3;
for k = 1:5
    xm = (xleft+xright)/2;
    if (sin(xm)-0.4*xm)*(sin(xleft)-0.4*xleft) < 0
        xright = xm;
    else
        xleft = xm;
    end
end
xm = (xleft+xright)/2;
fprintf('solution is %12.6f \n',xm)
```

This simple program ignores the possibility that the mid-point may coincides with the exact solution. A minor modification would easily take that into account. This is left to you as an exercise.

Solution, Prob 4 version 1 (by hand) Thanks to Jason Kmon

Newton's method utilizes derivatives of the function in order to approximate the zero of a function. The derivative at an initial guess is computed. From the derivative the point where the tangent line crosses the x-axis becomes the next guess. This process continues until $|x_N - x_{N-1}| < 0.01$.

$$e^{-x} - x^2 + 3x = 2$$
$$f(x) = e^{-x} - x^2 + 3x - 2$$
$$f'(x) = -e^{-x} - 2x + 3$$



$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

Solution 1

$x_1 = -4$

$f(-4) = 24.59815$

$f'(-4) = -43.59815$

$$x_2 = -4 - \frac{24.59815}{-43.59815} = -3.43579$$

$|-3.43579 - -4| > 0.01$

$f(-3.43579) = 6.94391$

$f'(-3.43579) = -21.18436$

$$x_3 = -3.13579 - \frac{6.94391}{-21.18436} = -2.80801$$

$|-2.80801 - -3.43579| > 0.01$

$f(-2.80801) = -1.73205$

$f'(-2.80801) = -7.96087$

$$x_4 = -2.80801 - \frac{-1.73205}{-7.96087} = -2.91679$$

$|-2.91679 - -2.80801| > 0.01$

$f(-2.91679) = -0.77616$

$f'(-2.91679) = -9.64828$

$$x_5 = -2.91679 - \frac{-0.77616}{-9.64828} = -2.99723$$

$|-2.99723 - -2.91679| > 0.01$

$f(-2.99723) = -0.05489$

$f'(-2.99723) = -11.03552$

$$x_6 = -2.99723 - \frac{-0.77616}{-9.64828} = -3.00220$$

$|3.00220 - 2.99723| < 0.01$

Solution 2

$x_1 = 1$

$f(1) = 0.36787$

$f'(1) = 0.63212$

$$x_2 = 1 - \frac{0.36787}{0.63212} = 0.41803$$

$|0.41803 - 1| > 0.01$

$f(0.41803) = -0.26231$

$f'(0.41803) = 2.22469$

$$x_3 = 0.41803 - \frac{-0.26231}{2.22469} = 0.53594$$

$|0.53594 - 0.41803| > 0.01$

$f(0.53594) = -0.09423$

$f'(0.53594) = -1.34300$

$$x_4 = 0.53594 - \frac{-0.09423}{-1.34300} = 0.60610$$

$|0.60610 - 0.53594| > 0.01$

$f(0.60610) = -0.00358$

$f'(0.60610) = 1.24233$

$$x_5 = 0.60610 - \frac{-0.00358}{1.24233} = 0.60898$$

$|0.60898 - 0.53594| > 0.01$

$f(0.60898) = -1.127 \times 10^{-5}$

$f'(0.60898) = 1.23813$

$$x_6 = 0.60898 - \frac{-1.127 \times 10^{-5}}{1.23813} = 0.60899$$

$|0.60899 - 0.60898| < 0.01$

(Continued to next page)

(Continued from previous page)

Solution 3

$x_1 = 2$

$f(2) = 0.13534$

$f'(2) = -1.13534$

$x_2 = 2 - \frac{0.13534}{-1.13534} = 2.11921$

$|2.11921 - 2| > 0.01$

$f(2.11921) = -0.01329$

$f'(2.11921) = -1.35855$

$x_3 = 2.11921 - \frac{-0.01329}{-1.35855} = 2.10943$

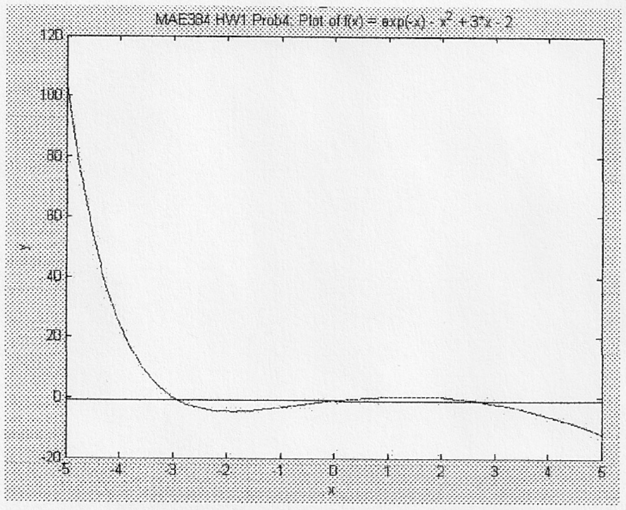
$|2.1943 - 2.11921| < 0.01$

$\boxed{X_1 = -3.00220 \qquad X_2 = 0.60899 \qquad X_3 = 2.10943}$

Choosing an initial guess on different sides of a maxima or minima alters which solution Newton's Method approaches. The side facing the value that's sought will yield that particular solution. Also choosing an initial guess which is very close to the solution will produce the satisfactory result more quickly. The first iteration takes the largest leap towards the solution, so if it is close to the intended value fewer calculations are needed.

Solution, Prob 4 version 2 (Matlab, next 3 pages) Thanks to Susanna Young
(Fast forward to the end to see the discussion. For simpler Matlab codes, see version 3)

Step 1: Plot the function in order to make an educated guess for the initial estimate of the solution
Using this command: fplot ('exp(-x)-x^2+3*x-2',[-10,50])



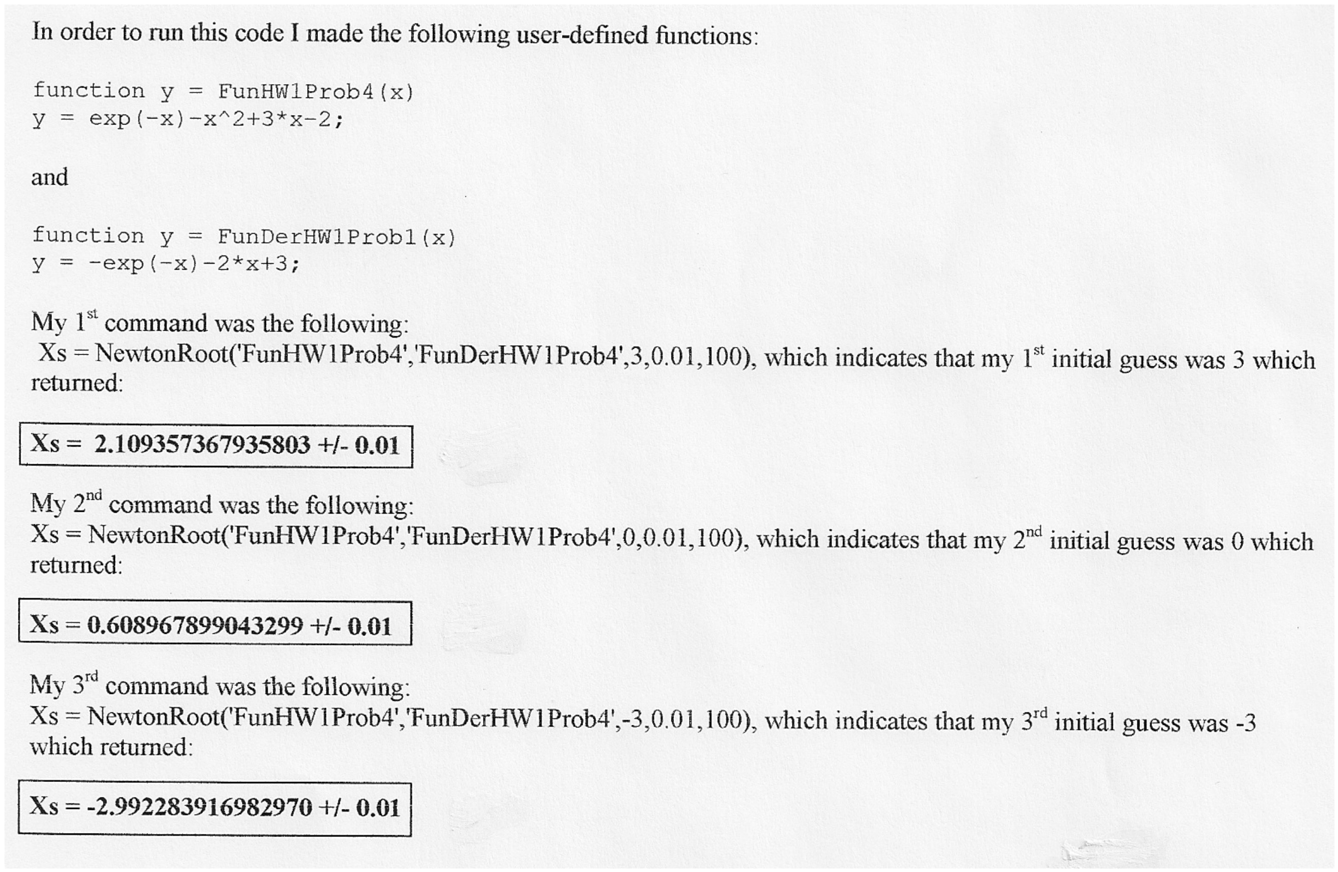
Based on the graph above it seems clear that there are 3 solutions to this function. Therefore I will perform the following code from the textbook with three different initial solution estimates.

```
function Xs=NewtonRoot(Fun, FunDer, Xest, Err, imax)
%NewtonRoot finds the root of Fun = 0 near the point Xest using Newton's
%method
%Input variables:
%Fund Name(string) ofa funtion file that calculates Fun for a given x.
%FunDer  Name(string) of a function file that calculates the derivative of
%   Fun for a given x
%Xest Initial estimate of the solution
%Err Maximum error.
%imax Maximum number of iterations
%Output variable:
%Xs Solutions

for i=1:imax
        Xi=Xest - feval(Fun,Xest)/feval(FunDer,Xest);
        if abs((Xi - Xest)/Xest)<Err
            Xs=Xi;
            break
        end
        Xest=Xi;
end
if i==imax
    fprintf('Solution was not obtained in %i iterations.\n',imax)
    Xs=('No answer');
end
```

(Continued to next page)

(Continued from previous page)



In order to run this code I made the following user-defined functions:

```
function y = FunHW1Prob4(x)
y = exp(-x)-x^2+3*x-2;
```

and

```
function y = FunDerHW1Prob1(x)
y = -exp(-x)-2*x+3;
```

My 1st command was the following:
Xs = NewtonRoot('FunHW1Prob4','FunDerHW1Prob4',3,0.01,100), which indicates that my 1st initial guess was 3 which returned:

Xs = 2.109357367935803 +/- 0.01

My 2nd command was the following:
Xs = NewtonRoot('FunHW1Prob4','FunDerHW1Prob4',0,0.01,100), which indicates that my 2nd initial guess was 0 which returned:

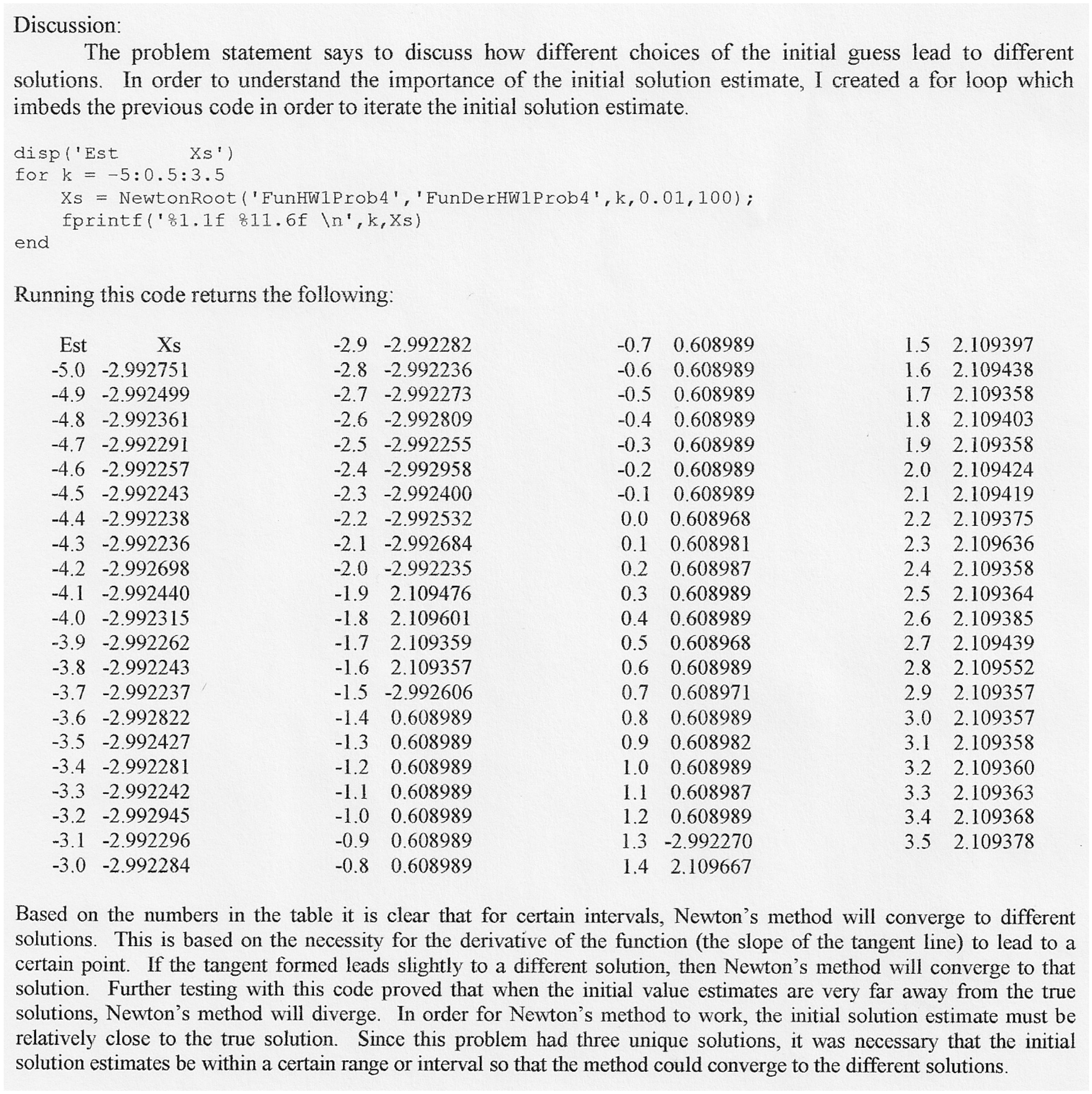
Xs = 0.608967899043299 +/- 0.01

My 3rd command was the following:
Xs = NewtonRoot('FunHW1Prob4','FunDerHW1Prob4',-3,0.01,100), which indicates that my 3rd initial guess was -3 which returned:

Xs = -2.992283916982970 +/- 0.01

(Continued to next page)

(Continued from previous page)



Discussion:

The problem statement says to discuss how different choices of the initial guess lead to different solutions. In order to understand the importance of the initial solution estimate, I created a for loop which imbeds the previous code in order to iterate the initial solution estimate.

```
disp('Est       Xs')
for k = -5:0.5:3.5
    Xs = NewtonRoot('FunHW1Prob4','FunDerHW1Prob4',k,0.01,100);
    fprintf('%1.1f %11.6f \n',k,Xs)
end
```

Running this code returns the following:

```
 Est      Xs           -2.9  -2.992282       -0.7   0.608989      1.5  2.109397
-5.0  -2.992751        -2.8  -2.992236       -0.6   0.608989      1.6  2.109438
-4.9  -2.992499        -2.7  -2.992273       -0.5   0.608989      1.7  2.109358
-4.8  -2.992361        -2.6  -2.992809       -0.4   0.608989      1.8  2.109403
-4.7  -2.992291        -2.5  -2.992255       -0.3   0.608989      1.9  2.109358
-4.6  -2.992257        -2.4  -2.992958       -0.2   0.608989      2.0  2.109424
-4.5  -2.992243        -2.3  -2.992400       -0.1   0.608989      2.1  2.109419
-4.4  -2.992238        -2.2  -2.992532        0.0   0.608968      2.2  2.109375
-4.3  -2.992236        -2.1  -2.992684        0.1   0.608981      2.3  2.109636
-4.2  -2.992698        -2.0  -2.992235        0.2   0.608987      2.4  2.109358
-4.1  -2.992440        -1.9   2.109476        0.3   0.608989      2.5  2.109364
-4.0  -2.992315        -1.8   2.109601        0.4   0.608989      2.6  2.109385
-3.9  -2.992262        -1.7   2.109359        0.5   0.608968      2.7  2.109439
-3.8  -2.992243        -1.6   2.109357        0.6   0.608989      2.8  2.109552
-3.7  -2.992237        -1.5  -2.992606        0.7   0.608971      2.9  2.109357
-3.6  -2.992822        -1.4   0.608989        0.8   0.608989      3.0  2.109357
-3.5  -2.992427        -1.3   0.608989        0.9   0.608982      3.1  2.109358
-3.4  -2.992281        -1.2   0.608989        1.0   0.608989      3.2  2.109360
-3.3  -2.992242        -1.1   0.608989        1.1   0.608987      3.3  2.109363
-3.2  -2.992945        -1.0   0.608989        1.2   0.608989      3.4  2.109368
-3.1  -2.992296        -0.9   0.608989        1.3  -2.992270      3.5  2.109378
-3.0  -2.992284        -0.8   0.608989        1.4   2.109667
```

Based on the numbers in the table it is clear that for certain intervals, Newton's method will converge to different solutions. This is based on the necessity for the derivative of the function (the slope of the tangent line) to lead to a certain point. If the tangent formed leads slightly to a different solution, then Newton's method will converge to that solution. Further testing with this code proved that when the initial value estimates are very far away from the true solutions, Newton's method will diverge. In order for Newton's method to work, the initial solution estimate must be relatively close to the true solution. Since this problem had three unique solutions, it was necessary that the initial solution estimates be within a certain range or interval so that the method could converge to the different solutions.

The above calculation shows how the final solution is related the initial guess. Note that this is accomplished by embedding the segment of Matlab code for a single solution within another do loop that steps through different initial guesses, in this case from -5 to 3.5 with an increment of 0.1. Notice the "surprising" outcomes with $x_1 = -1.9, -1.8, -1.7, -1.6$, and -1.5, and $x_1 = 1.3$. See general discussion in p. 1 for further explanation. -- HPH

Solution, Prob 4   version 3 (Simpler Matlab codes by HPH)

A barebone version for obtaining a single solution (this example chooses initial guess = 0.1 and performes 10 iterations):

```
x = 0.1;
for m = 1:10
   x = x - (exp(-x)-x^2+3*x-2)/(-exp(-x)-2*x+3);
   fprintf('solution is %12.7f  after %3i iteration(s) \n',x,m)
end
```

To aid the discussion, the following is an example that steps through a range of initial guesses (from -3 to 3 with an increment of 0.1) to obtain the corresponding final solutions. The outcome is similar to that in the discussion in version 2.

```
fprintf('     initial     final   \n')
for k = 1:51
    x = -3+(k-1)*0.1;
    xinitial = x;
    for m = 1:30
        x = x - (exp(-x)-x^2+3*x-2)/(-exp(-x)-2*x+3);
    end
    xfinal = x;
    fprintf('%12.6f %12.6f \n',xinitial,xfinal)
end
```