**ENEL102, fall term 2017**

**Assignment 4**

**Solving nonlinear functions, Chapter 9**

**Due date: Oct 30**

This assignment is based on the material in section 9.1. Suggest you read through this first before attempting the assignment questions. As usual, fill in this template with your matlab code, output and analysis. Then submit your Word document on D2L.

………………………………………………………………………………………………………………………………..

**Q1.** Consider the single variable equation of



This function is known to have a single real solution of f(x)=0. Find this solution by writing an anonymous function for f(x) and then using this in fzero() to find the solution of f(x)=0. Generate a plot of f(x) showing the root location and thereby verifying the answer generated by fzero(). List the Matlab statements used and the answer

**(Matlab input)**

f41 = figure('Name', 'Assignment 4, Question 1');

figure(f41);

f = @(x) exp(0.45\*x)+2\*x.^3+2;

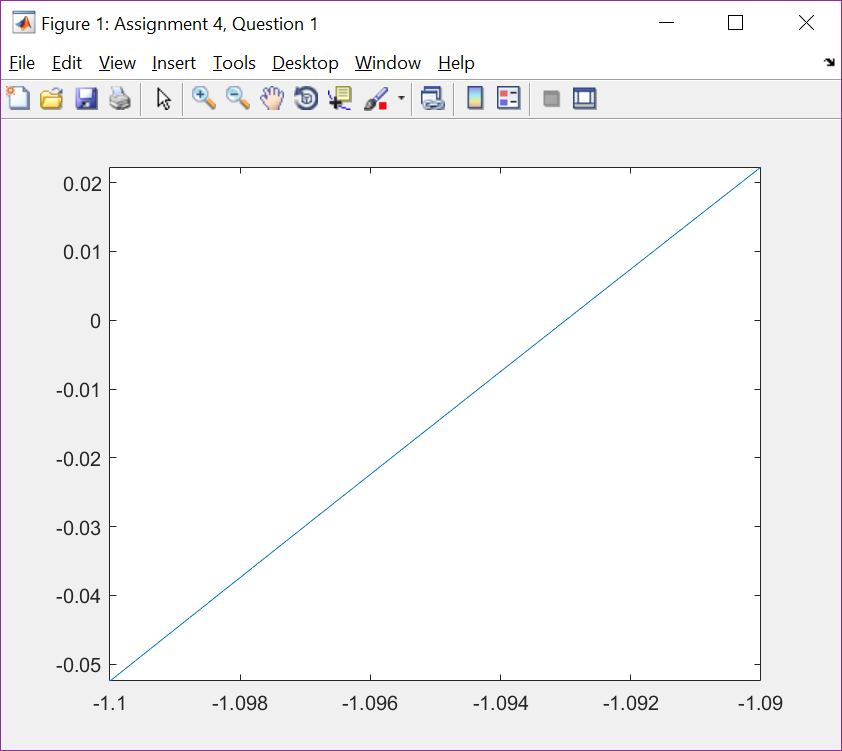
z = fzero(f,0)

fplot(f,[-1.1,-1.09]);

**(Matlab Response)**

z =

-1.0930



………………………………………………………………………………………………………………………………..

**Q2.** Consider the single variable equation of



This function is known to have three solutions of x that satisfy f(x)=0. That is three values of x satisfy this relation. Generate a plot such that these solutions are easily observed.

**(Matlab input)**

f42 = figure('Name', 'Assignment 4, Question 2');

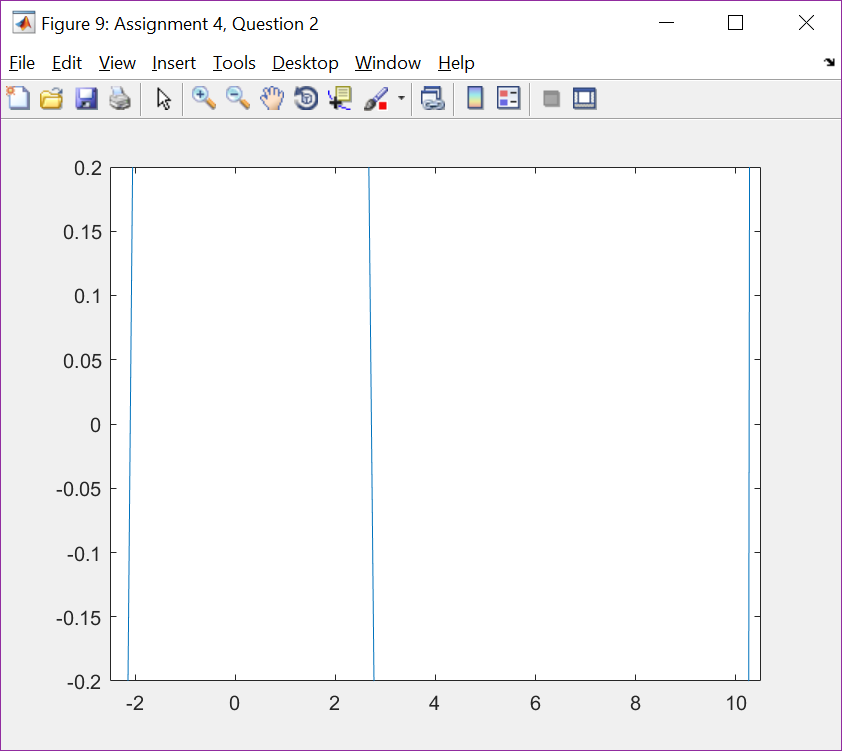
figure(f42);

f = @(x) exp(0.45\*x)-x.^2+4;

fplot(f,[-2.5,10.5]);

ylim([-1,1]);

**(Matlab Response)**



………………………………………………………………………………………………………………………………..

**Q3.** For the function given by **Q2** above find the three values of x that satisfy f(x)=0 using fzero(). Note that fzero() can only find one solution for a given initial guess for x. One way to ensure that we can find the three solutions is to use a set of initial conditions. Take the start points as x = -15,-14,…15 and use fsolve on each start point individually and store the point to which it converges. Show a plot of the output of fsolve as a function of the input guesses. Next superimpose the solutions of x onto a plot of f(x) verifying correctness of fzero().

**(Matlab input)**

f431 = figure('Name', 'Assignment 4, Question 3, Plot 1');

figure(f431);

f = @(x) exp(0.45\*x)-x.^2+4;

z = zeros(31,1);

for i = -15:15

z(i+16) = fzero(f,i);

end

plot(-15:15,z);

f432 = figure('Name', 'Assignment 4, Question 3, Plot 2');

figure(f432);

fplot(f,[-2.5,10.5]);

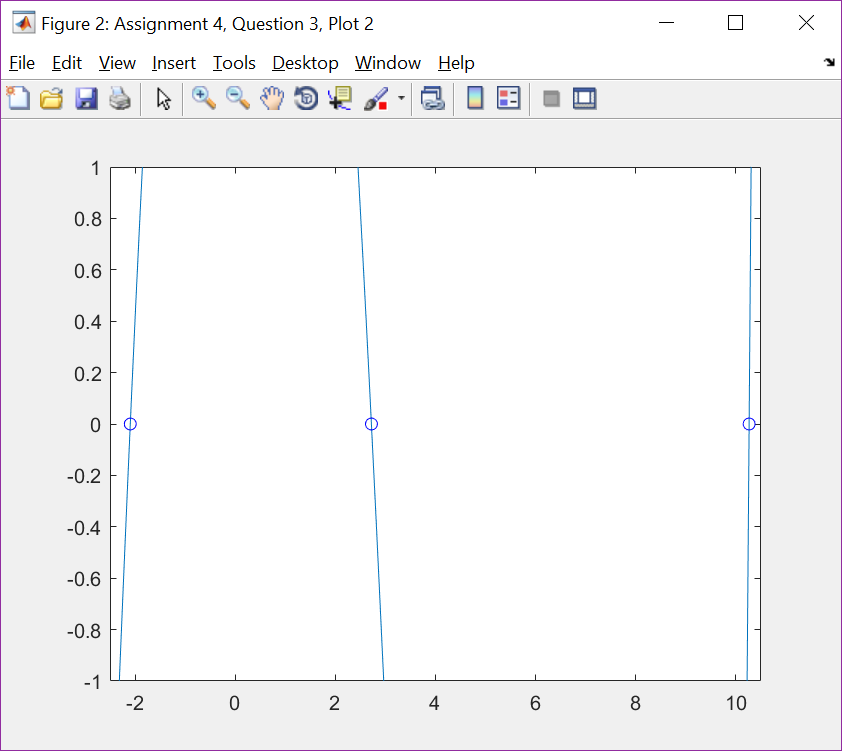
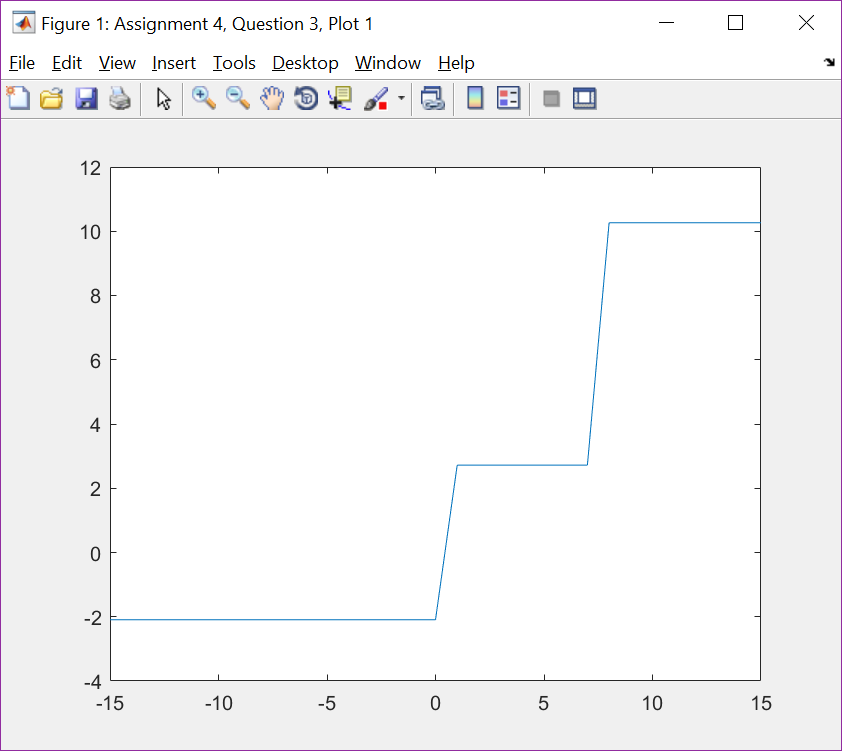
ylim([-1,1]);

hold on;

plot(z(5),0,'ob',z(18),0,'ob',z(31),0,'ob');

hold off;

**(Matlab Response)**

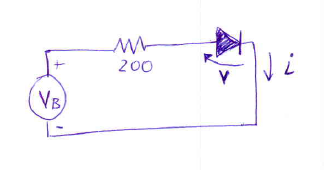


………………………………………………………………………………………………………………………………..

**Q4** Consider a series circuit of a battery of voltage Vb, connected to a series circuit consisting of a diode with a current-voltage relation of



and a resistor of 200 ohms. The diode is connected such that it is forward biased as shown in the diagram.



Plot the current through the diode (i) as a function of the voltage across it (v).

Now using fzero(), find the voltage across the diode when the supply battery is one volt.

**(Matlab input)**

f44 = figure('Name', 'Assignment 4, Question 4');

figure(f44);

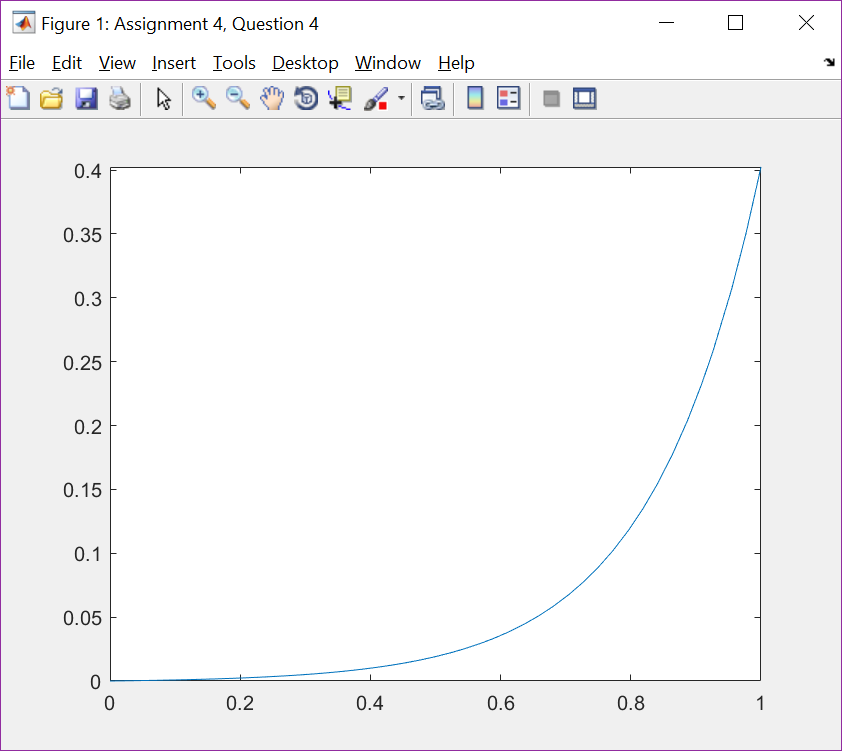
i = @(v) 0.001\*(exp(6\*v)-1);

fplot(i,[0,1]);

f = @(x)0.001\*(exp(6\*(1-200\*x))-1)-x;

i=fzero(f,1)

**(Matlab Response)**



………………………………………………………………………………………………………………………………..

**Q5** You can get information regarding the convergence of fzero() by setting the optimset() to display the output. Use this to **a)** Determine the number of function calls required to find the solution in **Q4**. **b)** Show the command for optimset that changes the function residual tolerance from the default of 1e-4 to 1e-8. **c)** Run the diode current problem again in **Q4** with this tighter tolerance and explain why no additional function calls are required for this particular problem when the function tolerance is changed.

**(Matlab input)**

% part a

options = optimset('Display','iter');

i=fzero(f,1,options)

% part b

options = optimset(options,'TolFun',1.0e-8);

i=fzero(f,1,options)

**(explanation)**

The bounding constraint on the solution is not the function tolerance, but the tolerance on x. The function tolerance is being met in both cases, and it only continues because it has not met the x tolerance, which doesn’t change.

………………………………………………………………………………………………………………………………..

**Q6** Consider the intersection of a circle and line given by the functions



By inspection, determine the two points of intersection

**(sol)**

………………………………………………………………………………………………………………………………..

**Q7** Use fsolve() to determine the two points of intersection. Show the Matlab code to do this.

**(Matlab input)**

f = @(x) [x(1)^2 + x(2)^2 - 1; x(1) - x(2)];

options = optimset('Display','off');

fsolve(f,[1;1],options)

fsolve(f,[-1;-1],options)

**(Matlab Response)**

ans =

0.7071

0.7071

ans =

-0.7071

-0.7071

………………………………………………………………………………………………………………………………..

**Q8** Consider a matrix with unknown coefficients of a,b,c that is given by the model of



Lab measurements show that A has the values of



Find the values of a b and c using fsolve().

**(Matlab input)**

f = @(x) [x(1)\*x(2)-4;x(1)\*x(3)-3;x(1)\*x(2)+x(3)-5];

fsolve(f,[0;0;0])

**(Matlab Response)**

ans =

3.0000

1.3333

1.0000

………………………………………………………………………………………………………………………………..

**Q9** Consider the spiral function generated by the parametric equations



for t>0. Also there is the parabolic function of



The objective is to determine the points of intersection of the spiral and parabola in the x-y plane that have a magnitude of less than 10. Use whatever numerical means to find these and list them out. List your program with outputs and sufficient comments that the TA can understand what you are doing.

**(Matlab input)**

options = optimset('MaxFunEval',1000,'Display','off');

% Set up options

% larger MaxFunEval makes getting an answer more likely

% Display off turns off fsolve text output

f = @(v) [2+v(1)^2-v(2);v(3)\*cos(2\*v(3))-v(1);v(3)\*sin(2\*v(3))-v(2)];

% v = [x;y;t]

% f = [

% 2+x^2-y

% t\*cos(2\*t)-x

% t\*sin(2\*t)-y

% ];

% when f is equal to the zero vector, we are on both the parabola and

% the spiral.

a = [0;0;0];

% for saving the last solution

for t = 1:0.1:11

% temp variable for starting position

n=fsolve(f,[t\*cos(2\*t);t\*sin(2\*t);t],options);

% solve the system, starting from a point on the spiral, that

% progresses clockwise around the spiral

if norm(a-n) > 0.01

% if this solution is substantially different from the last

% solution

if norm(f(n)) < 0.01

% and this solution puts the function value very close to 0

disp('----------------------------------------------');

x = n(1), y = n(2)

% display the solution information

a = n;

% save the solution

end

end

end

**(Matlab Response)**

----------------------------------------------

x =

1.2436

y =

3.5466

----------------------------------------------

x =

-1.3648

y =

3.8628

----------------------------------------------

x =

2.1383

y =

6.5722

----------------------------------------------

x =

-2.2086

y =

6.8780

----------------------------------------------

x =

2.7716

y =

9.6819

----------------------------------------------

x =

-2.8207

y =

9.9564