ELEC 3004 – Systems: Signals & Controls

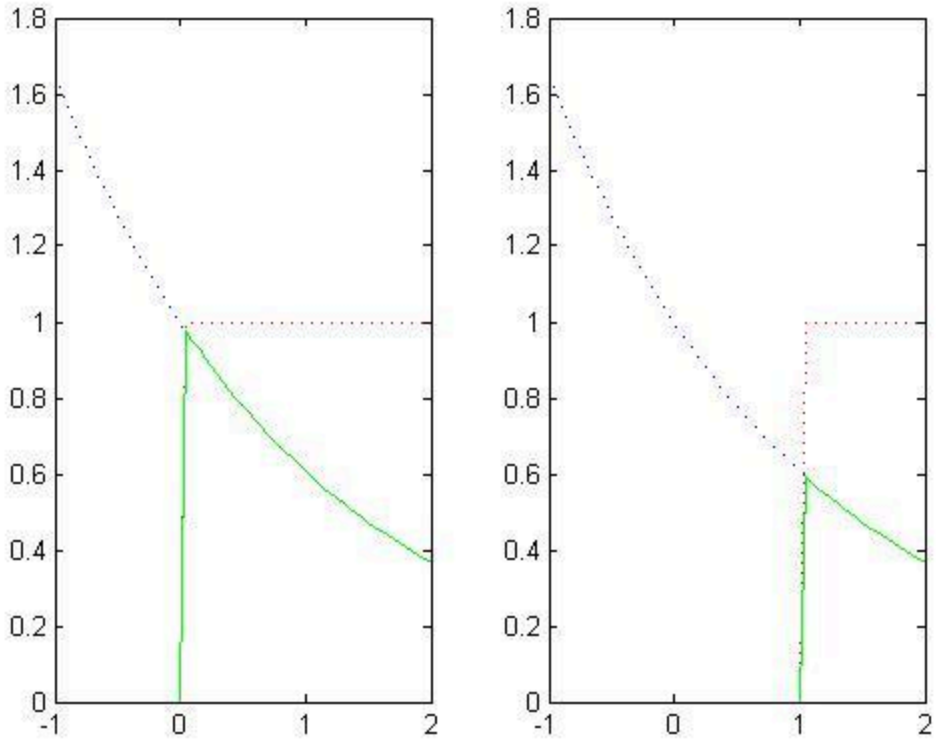
Matlab - Random Tidbits!

Things to help you with the first assignment!

How to construct a step function function:

```
s = -1; f = 2; n = 0.05; % where s is the start_time, f is the finish_time and
n is the time interval step size

t = s:n:f;          % set up the vector of time values
g = exp(-t/2);

len = (0-s)/n + f/n;  % find the length from the start and finish values
u = ones(1, len + 1); % make a long enough vector for u(t) filled with ones
u(1:abs(s)/n + 1) = 0; % make the part before t = 0, equal to 0

subplot(1,2,1)
plot(t, g);
hold on
plot(t, u, 'r-');
plot(t, u.*g, 'g:'); % don't forget to use the 'dot' operator when multiplying
vectors together!

% This part is to implement a time-shifted step function, i.e. u(t - T)
subplot(1,2,2)
numSec = 1;
shift = numSec*1/n; % the number of seconds you want to shift by multiplied by
the number of elements in 1 second (based on the stepsize 'n')
u2 = ones(1, (f-s)/n+1);
u2(1: abs(s)/n + 1 + shift) = 0;
% put -shift, if the shift is negative, i.e. u(t + T)

plot(t,u2, 'r-');
hold on
plot(t, g);
plot(t,u2.*g, 'g:');
```

The output appears below! Note that the green line is the unit step multiplied by the exponential function. You can also look at the Matlab command heaviside() and dirac().



How to model sampling of a continuous signal, particularly sampling close to or less than the Nyquist rate:

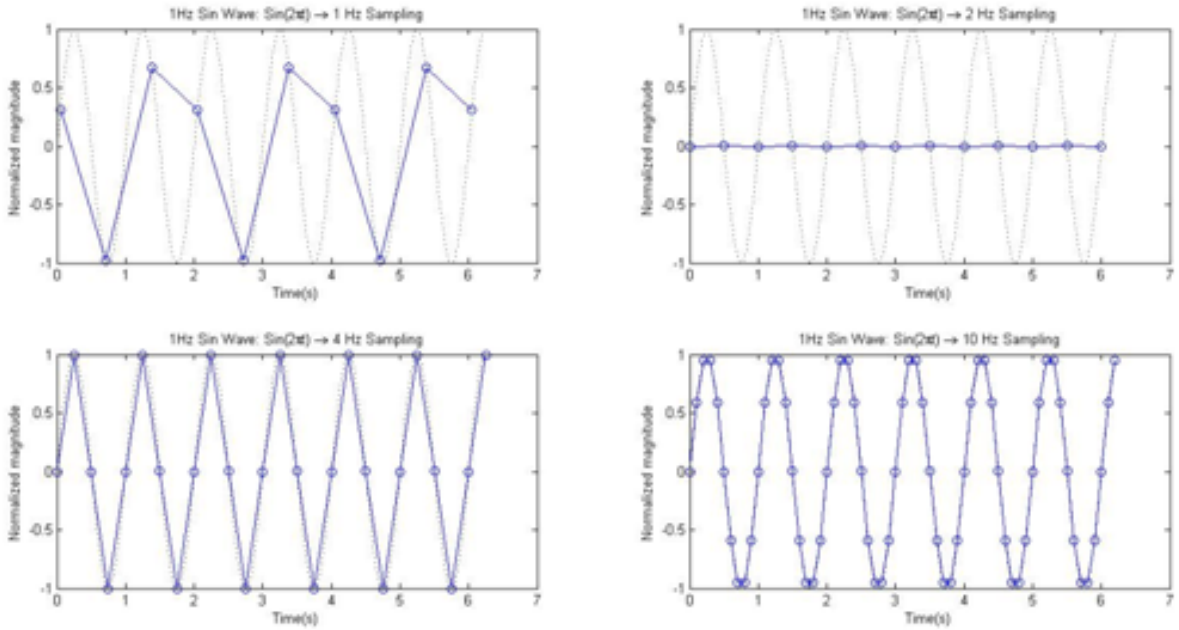
```
% Once you know the frequency of the signal, just modify the step_size, 'n' %
to reflect the sampling rate you want.
freq = 1;
tfine = 0:1e-2:2*pi;  % step size of 0.01 or 100 Hz (1/100)
sfine = sin(2*pi*freq*tfine);  % sine wave with 1Hz frequency


% Plot with 1.5Hz sampling with 1/20s delay
tshift = 1/20;  % to introduce delay in the sampling, pick the time delay you
want (tshift)
tcourse = (0:1/1.5:2*pi) + tshift; % then add it to the time vector
scourse = sin(2*pi*freq*tcourse);
subplot(2,2,1)
plot(tfine, sfine,'k:', tcourse, scourse, 'bo-')


% Plot with 2Hz sampling
tcourse = 0:1/2:2*pi; % for 1Hz wave, use n = 0.5 or 1/2
scourse = sin(2*pi*freq*tcourse);
subplot(2,2,2)
```

```
plot(tfine, sfine,'k:', tcourse, scourse, 'bo-')
% once, you've seen what this plot looks like, for something cool, try adding %
in a 1/4 s time delay to see what happens

% Plot with 4Hz sampling
tcourse = 0:1/4:2*pi;  % change to 1/4 here
scourse = sin(2*pi*freq*tcourse);
subplot(2,2,3)
plot(tfine, sfine,'k:', tcourse, scourse, 'bo-')

% Plot with 10Hz sampling
tcourse = 0:1/10:2*pi; % change to 1/10 here
scourse = sin(2*pi*freq*tcourse);
subplot(2,2,4)
plot(tfine, sfine,'k:', tcourse, scourse, 'bo-')
```

The output should look like the Figure below.

So far, all of the sampling rates were uniformly spaced. If you want to mimic asynchronous sampling, try the following:

```
% For logspace sampling!
tmpcourse = 0:0.125:pi/2;
tcourse = logspace(-1.5,pi,100);
tcourse = [tcourse, tcourse + pi];

% For random sample perturbation!
tshift = (pi/60);
randscale = 0.5;
tcourse = (0:0.5:(2*pi-0.5*randscale));
tcourse = tcourse+randscale*rand(size(tcourse));
```

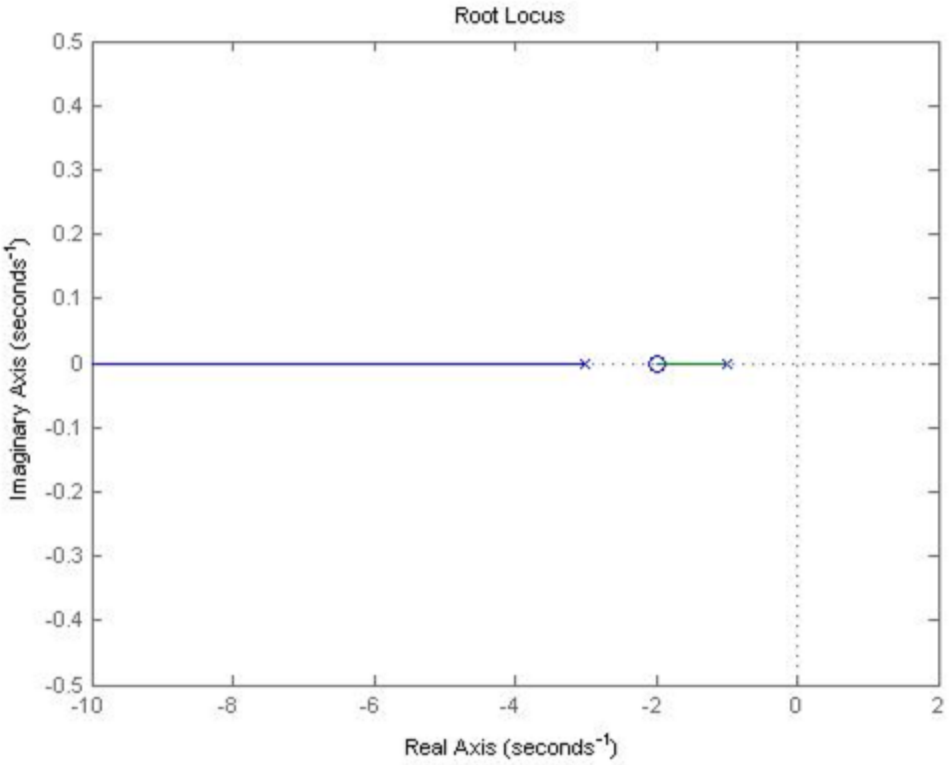


To get the title and axis labels, use the following format. Putting this text block after every subplot command, gives that specific subplot the labels you've written.

```
title('Name of function here'); % \rightarrow gives the right arrow symbol
xlabel('Time(s)');  % label for the x axis
ylabel('Normalized magnitude'); % label for the y axis
```

How to plot the root locus of a system from its transfer function:

```
% The system is from Assignment 0: (2s + 4)/(s^2 + 4s + 3)
num = [2 4];
den = [1 4 3];
sys2 = tf(num, den); % remember this Matlab tute 0 - make a transfer function!
residue(num, den) % returns the numerator values of the fraction expansion
roots(den) % returns the roots of the polynomial in the denominator
rlocusplot(sys2);
```


Root Locus
0.5
0.4
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
-0.4
-0.5
-10
-8
-6
-4
-2
0
2
Imaginary Axis (seconds-1)
Real Axis (seconds-1)