ELEC 3004 – Systems: Signals & Controls

Matlab - What does that even look like?

As you've learnt, signals can be odd or even, periodic or non-periodic. Most of the time you can tell straight away what the signal will look like and hence what its characteristics are. Sometimes, however, you may see a function that you cannot readily identify without a lot of intuition or a good piece of software to graph it with.

First, some basics:

```
% Set the time – t = start:step_size:finish
% The smaller your step_size the more continuous the function will look when it's plotted
t = 0:0.01:2*pi;   % this makes a vector that is (2*pi – 0)/0.01 elements long.

% Pick a simple function, e.g. cos(t). The output is a vector the same length as t.
f = cos(t);

% Make a pretty picture!
plot(t, f);
```

Now for some slightly more complex functions. Say you have e^{st} where $s = \sigma + j\omega$.. If both σand $j\omega$ are not equal to zero, you will need to plot an exponential and cosine multiplied together, i.e. $e^{\sigma t}cos(\omega t)$ .You can do this with the following command:

```
% Using the same t vector as before
sig = -3;  omega = 2;  % you can put things on the same line if you separate them with a semi-colon

% the dot here is very important, it allows element-wise multiplication since exp and cos make
% separate vectors. Without it, Matlab throws an error.
g = exp(sig*t) .* cos(omega*t);
plot(t, g);
```

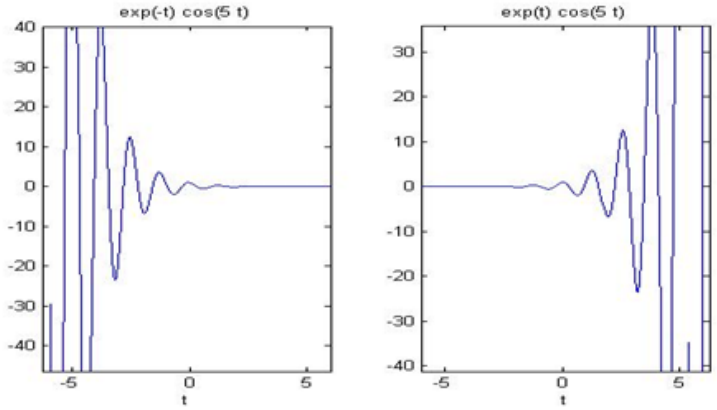
What if you'd like to compare the outcome of having positive and negative sigma, σ? That's where subplot comes in handy.

```
% Subplot takes the form: subplot(numOfRows, numOfCol, currentPos), e.g. subplot(1,2,1) makes
% two plots side by side and subplot(2,1,1) makes two plots stacked on top of each other.  Try the
% following:

subplot(1,2,1);
ezplot('exp(-t)*cos(5*t)');     % note: you don't need the dot because ezplot takes a string and
% interprets the variables (t in this case) for you – no need to set up a vector of time values!
subplot(1,2,2);
```

ezplot('exp(t)*cos(5*t)'); % added the 5*t to make it extra jiggly!

This is the result! Notice how the function with a negative sigma value is the one that is stable?



Lastly, you may want to plot a discrete function. Essentially, every plot in Matlab is discrete (since it is a digital program) but the 'sampling rate' (step size between time values) is so high that the functions look continuous. So in that sense, to make a function that looks more like a discrete function, you set your own step size to be a bit larger than usual and use the command 'stem()'.

```
% A 'sampled' cosine wave
n = 0:0.2:2*pi;
f = cos(n);
stem(n, f);
```

You can plot several of these together on one plot using the command 'hold on', like this:

```
% Add this bit to the end of the code snippet above for a discrete cosine wave
hold on;
f2 = sin(n);
stem(n, f2, 'r-');  % to make a red sine wave
```

The result is this! Now that you can combine waves on the same graph, you can use this method for graphing the superposition of two linear signals!