### ELEC 5705 RF Systems Design: Assignment #1

Due Oct. 12th, 2017 (late assignments will be penalized at 25% per day)

The goal of this assignment will be to become familiar with Matlab and some of its capabilities for doing system level RF analysis. To that end we will begin by creating a model for a nonlinear amplifier. Create a function for your amplifier which takes as inputs: the input voltage, the gain of the amplifier, the IIP2, the IIP3, the NF, and the time step between the points in the input voltage array.

Your amplifier must be able to model nonlinear behaviour and have realistic noise performance. Once you have constructed your amplifier you should create a test bench and simulated the following:

1. The transient response of an amplifier with a 5dB noise figure, 15dB gain, and a -20dBm IIP3.
2. Determine the input and output SNR for this amplifier. Produce power spectral density plots of the frequency.
3. Produce a plot of the output power of the IM3 products, and fundamental power versus input power for an input with two tones. Determine the 1dB compression point of the amplifier.

Reports should be short and include plots and results with some discussion of what they mean. A maximum limit of 10 pages for each report but they can be shorter.

Some helpful hints follow:

The wgn command in matlab is likely to come in handy. The trick is getting it to have the right amplitude for the right bandwidth. For instance:

wgn (1, length(Vin), NoiseFloor + 10\*log10(1/(2\*tstep)), R, 'dBm')

should produce a time domain noise voltage array with the same number of points as Vin. tstep is the time spacing of points in Vin and the output. NoiseFloor is the noise power in a 1Hz bandwidth of the noise source you are trying to model.

To make ffts have a low numerical noise floor in matlab it is necessary to pick the length of your arrays carefully. For instance:

%Set the total time for the simulation. And the step time.

tstop = 100e-6;

tstep = 5e-10;

%Create the time variable.

time= 0:tstep:tstop-tstep;

will result in low numerical noise and will be much better than:

time= 0:tstep:tstop;

If you are running a lot of time based simulations, often you will want to see the frequency response. Here is a function that may help:

function [ freq pwr ] = fft\_pwr( tstep, voltage, R )

% This fuction is going to take in voltage array and give a

% scalled fft. The voltage elements must be equally spaced in time.

% The tstep input is the time spacing of the voltage points.

% It is a single real number. The input R specifies the resistance the

% power is to be calculated for. The default value is 50ohms if this

% input is ommited.

% The output will be an array of frequency in MHz and one of power in dBm.

clearvars freqP freqN freq pwr z x y w j;

%Determine the frequency step knowning the total time of the simulation.

freqstep = 1/tstep/length(voltage);

% The reference impedance is assumed to be 50ohms if it isn't specified.

if nargin < 3

 R = 50;

end

%Create the x axis data points knowning how many voltage points we have.

%This will be slightly different if we have an even or odd number of array

%points.

%if even.

if (length(voltage)) == (2\*round(length(voltage)/2))

 freqP = 0:freqstep:freqstep\*(length(voltage)/2-1);

 freqN = -freqstep\*(length(voltage)/2):freqstep:-freqstep;

 freq = [freqN freqP];

%if odd.

else

 freqP = 0:freqstep:freqstep\*(length(voltage)/2-1);

 freqN = -freqstep\*round(length(voltage)/2):freqstep:-freqstep;

 freq = [freqN freqP];

end

 %Scale to MHz.

freq = freq ./1e6;

% Compute the fft.

z = abs(fft(voltage));

%Scale the fft to give the peak amplitude of the waveform.

z = z/length(z);

z = z \* 2;

w = fftshift(z);

%Now convert the voltage into dBm.

pwr = 10\*log10(w.^2/(2\*R))+30;

end