

```

Editor - /Users/kcedrone/Dropbox/2.671/S2018/Lab 2 - Filtering/inp_out_PSD.m
inp_out_PSD.m x +
1  % Read input file
2  [filename, filepath] = uigetfile('*.wav', 'Select input file...'); % Opens a dialog box so you can choose
3  input_file = fullfile(filepath, filename); % Creates the filename
4  [inp, Fs] = audioread(input_file);
5
6  %% Pre-processing
7  dt = 1.0/Fs; % Time between discrete samples
8
9  %% Filter creation
10 fc_Hz = 300.0; % Cut-off frequency in Hz
11 tau = 1.0/(2*pi*fc_Hz);
12 s = tf('s');
13
14 % Create transfer function of filter(s)
15 filt_type_label='1st order RC low-pass filter'; % For plotting purposes
16 tf_LPF = 1.0/(tau*s + 1.0); % 1st order RC Low pass filter
17 d_filt = c2d(tf_LPF, dt); % Convert continuous transfer function to discrete
18
19 [tf_num, tf_denom] = tfdata(d_filt, 'v'); % Separate into numerator and denominator
20
21 %% Butterworth filter
22 % As an alternative to creating it with transfer function,
23 % you could create a discrete filter directly:
24 % If you un-comment the following lines, this filter will overwrite the
25 % previous filter.
26 % fc_Hz = 300.0;
27 % fc_norm = fc_Hz/(Fs/2.0); % Normalized cutoff frequency
28 % filt_type_label='Butterworth'; % For plotting purposes
29 % filt_order = 4;
30 % [tf_num, tf_denom] = butter(filt_order, fc_norm, filt_type);
31
32 %% Apply the filter
33 outp = filter(tf_num, tf_denom, inp);
34
35

```

User selects a data file, in this case a .WAV audio file

I choose to name the outputs of audioread() inp and Fs

Analysis steps of creating a filter, and filtering the input data

Alternate analysis step is commented out to prevent it from running

```

35
36 %% Plot results
37 hFig = figure('Name', 'PSD Comparison');
38 set(hFig, 'Units', 'Normalized', 'OuterPosition', [0, 0.01, 0.9, 0.9]);
39
40 % PSD plot for input signal
41 ax1 = subplot(1,2,1);
42 [p, f] = pwelch(inp, [], [], [], Fs);
43 plot(f, p, 'b-');
44 grid on;
45 xlabel('Frequency (Hz)');
46 ylabel('PSD amplitude (V^2/Hz)');
47 xlim([1 10000]);
48 set(gca, 'TickDir', 'out');
49 title('Input PSD', ' ');
50 improvePlot;
51
52 % PSD plot for output signal
53 ax2 = subplot(1,2,2);
54 [p, f] = pwelch(outp, [], [], [], Fs);
55 plot(f, p, 'b-');
56 grid on;
57 xlabel('Frequency (Hz)');
58 ylabel('PSD amplitude (V^2/Hz)');
59 title(filt_type_label, 'Output PSD', ' ');
60 xlim([1 10000]);
61 improvePlot;
62 set(gca, 'TickDir', 'out');
63
64 linkaxes([ax1 ax2], 'xy'); % Link the axes so x and y axis limits are synced
65
66 % Clean up all the font-sizes
67 axis_handles=findobj(gcf, 'type', 'axe'); % Grab the axes handle(s)
68
69 % Iterate over all axes handle(s)
70 for ax = axis_handles
71     set(ax, 'FontSize', 16) % Change font size
72 end
73
74
75 %% Save figure as PNG
76 print(hFig, 'input_output_PSD_comparison', '-dpng');
77

```

Plot the results

Clean up the plot

Save the plot as .png file in the present working directory