clc;

clear all;

close all;

instrreset;

%% Parameters to set

T = 2; % Total sampling time in seconds

fs = 1000; % sampling frequency target in Hz

f\_min=1; % minimum frequency to be characterized in Hz

f\_max=30; % maximmum frequency to be characterized in Hz

f\_step=1; % frequency step in Hz

%%

f\_vec=[f\_min:f\_step:f\_max]; % initialize vector for frequency sweep

transfer\_vec=zeros(length(f\_vec),1); % initialize transfer function vector

for j=1:length(f\_vec) % loop over each frequency to be tested

% create the serial objects

% you must replace the port name with the port on your machine

% you can find this through the arduino interface (tools->port)

% the baud rate must match what you selected in your serial read ...

% Ardino code

% the first serial object, s\_gen, corresponds to the Arduino acting ...

% as your signal generator

% the second serial object, s\_read, corresponds to the Arduino ...

% acting as your oscilloscope

% open object for your signal generator arduino

% this segment sets the frequency of the generated signal

s\_gen = serial('COM7','BaudRate',115200);

fopen(s\_gen); % open the serial connection

pause(5); % pause for 5 seconds while the serial object is opened

fprintf(s\_gen,'%s',int2str(f\_vec(j))); % write the signal frequency

fclose(s\_gen); % close serial object

pause(2);

% open object for your "read" arduino

s\_read = serial('COM18','BaudRate',115200);

fopen(s\_read); % opens said serial object

flag=0; %set flag for timer

i=1; % set sample counter

dt\_set=1/fs; % set time step target

timer=0; % initialize timer

L=T\*fs\*2; % oversized vector length

time=zeros(L,1); % initialize time vector

A=zeros(L,1); % initialize amplitude vector

A\_ref=zeros(L,1);

waittime=2; %set initial wait time before sampling in seconds

t=0; % initialize time variable

ind=0; % initialize index variable

ind\_ref=0;

a=0; % initialize amplitude variable

a\_ref=0;

dump=''; % initialize text dump variable

out=''; % initialize serial output string variable

tic; % start timer

while toc<waittime % read and dump serial data until wait time is reached

dump=fscanf(s\_read);

end

while flag==0

out=fscanf(s\_read); % reading the serial port

ind\_ref=find(out==',',1); % find index of comma

ind=find(out==';',1); % find index of semi-colon

a=str2double(out(1:ind\_ref-1)); % get amplitude of output signal

% get amplitude of reference signal

a\_ref=str2double(out(ind\_ref+2:ind-1));

t=str2double(out(ind+2:end))/1E6; % get time

% condition to take sample at set sampling rate

if (t-timer)>dt\_set

time(i) = t; % establishing time steps for sampling frequency

A(i)=a; % add signal amplitude to signal amplitude vector

A\_ref(i)=a\_ref; % add reference amplitude to vector

timer=time(i);

i=i+1;

% condition to end loop when end time is reached

if t>(T+time(1))

flag=1;

end

end

end

fclose(s\_read); % closes serial port for "read" arduino

reps=i-1; % get number of samples acquired

time = time(1:reps)-time(1); % setup a vector for time

% convert serial amplitude to voltage

voltage = 5/1023\*A(1:reps);

% convert serial amplitude to voltage for reference signal

voltage\_ref = 5/1023\*A\_ref(1:reps);

% find the average time interval between samples

dt\_avg = mean(diff(time));

fs\_avg=1/dt\_avg; % calculate the average sampling frequency from dt\_avg

%% Create plot

figure(01); % setup figure 01

% plot time vs. voltage for output signal

subplot(221)

plot(time, voltage(1:reps),'-o','LineWidth',2,'MarkerSize',4);

xlabel('time (s)'); % x-axis label name

ylabel('voltage (V)'); % y-axis label name

ylim([min(voltage)-abs(0.1\*max(voltage)) ...

max(voltage)+abs(0.1\*max(voltage))]); % set y plot range

title(['f\_{s,average}=' num2str(round(fs\_avg)) ' Hz']); % set title as sampling rate

% get current plot axes, set font and line width

set(gca,'FontSize',22,'LineWidth',2);

set(gcf, 'units', 'normalized'); % set plot sizing to normalized units

% plot time vs. voltage for reference signal

subplot(222)

plot(time, voltage\_ref(1:reps),'-ok','LineWidth',2,'MarkerSize',4);

xlabel('time (s)'); % x-axis label name

ylabel('voltage (V)'); % y-axis label name

ylim([min(voltage\_ref)-abs(0.1\*max(voltage\_ref)) ...

max(voltage\_ref)+abs(0.1\*max(voltage\_ref))]); % set y plot range

title('Reference signal'); % set title as sampling rate

% get current plot axes, set font and line width

set(gca,'FontSize',22,'LineWidth',2);

set(gcf, 'units', 'normalized'); % set plot sizing to normalized units

% calculate Power spectral density (PSD) for the output signal

[PSD,f\_psd] = periodogram(voltage-mean(voltage),rectwin(reps),...

reps,fs\_avg,'onesided');

FT=sqrt(PSD); % convert Arduino PSD to Fourier magnitude

[FT\_max,ind\_max]=max(FT); % find the maximum of the output signal FT

% calculate Power spectral density (PSD) for the reference signal

[PSD\_ref,f\_psd] = periodogram(...

voltage\_ref-mean(voltage\_ref),rectwin(reps),...

reps,fs\_avg,'onesided');

FT\_ref=sqrt(PSD\_ref); % convert Arduino PSD to Fourier magnitude

% find the maximum of the reference signal FT

[FT\_max\_ref,ind\_max\_ref]=max(FT\_ref);

% calculate the transfer function value at this frequency

% e.g. the output divided by the input

transfer\_vec(j)=FT\_max/FT\_max\_ref;

% plot the FT spectra for the output and reference signals

subplot(223)

plot(f\_psd,FT\_ref,'k-o',f\_psd(ind\_max\_ref),FT\_max\_ref,'gx',...

f\_psd,FT,'b-o',f\_psd(ind\_max),FT\_max,'rx',...

'LineWidth',2,'MarkerSize',4);

xlabel('frequency (Hz)'); % x-axis label name

ylabel('|FT|'); % y-axis label name

title(['FFT of AC signals (f\_{in}=' int2str(f\_vec(j)) ' Hz)']); % set title as sampling rate

xlim([0 f\_max\*2]);

% get current plot axes, set font and line width

set(gca,'FontSize',22,'LineWidth',2);

set(gcf, 'units', 'normalized'); % set plot sizing to normalized units

% plot the transfer function

subplot(224)

plot(f\_vec,transfer\_vec,'r-o','LineWidth',2,'MarkerSize',4);

xlabel('frequency (Hz)'); % x-axis label name

ylabel('|FT|\_{max}/|FT|\_{max,ref}'); % y-axis label name

title('Transfer Function'); % set title as sampling rate

xlim([f\_min f\_max]);

% get current plot axes, set font and line width

set(gca,'FontSize',22,'LineWidth',2);

set(gcf, 'units', 'normalized'); % set plot sizing to normalized units

% set position of figure on screen [distance from left, top, width, height]

set(gcf, 'Position', [0.1, 0.1, .6, 0.8]);

drawnow;

pause(2);

end

% Be sure to change filenames if you don’t want to overwrite your data!

save('MAE170\_lab4','f\_vec','transfer\_vec');

% save frequency and gain to mat file

csvwrite('MAE170\_lab4.csv',[f\_vec',transfer\_vec]);

% save frequency and gain to csv file

saveas(gcf,'MAE170\_lab4');

% save figure