ECE 147C HW 1 Solution

Table of Contents

A Simulink block that models a pendulum with viscous friction is provided.

Part 1.

Use the Simulink block (without noise) to estimate the system's impulse response $\hat{h}(k)$ using both the impulse and step response methods.

What is the largest value α of the input magnitude for which the system still remains within the approximately linear region of operation?

Hint: to make sure that the estimate is good you can compare the impulse responses from the step method and impulse-response method.

```
clc; clear all; close all;
```

Setup

```
model_name = 'P2_mod';  % modified to work in Prob 2.2 as well

Ts = 0.01;         % Sampling period [sec/sample]
T  = 100;          % Length of simulation [sec]
K  = T/Ts;         % Length of simulation [samp]
noise = 0.0001;    % Noise variance in sim
noiseOn = 0;       % Noise disabled in part 1
t = 0:Ts:T;        % Time vector [sec]
alpha = 0.1;       % Input amplitude (guess initial value of $\alpha$)
```

Impulse response

```
alpha_step = 0;          % Configure the model for an impulse.
alpha_impulse = alpha;   % Configure the model for an impulse.
alpha_sine = 0; w = 0;   % Configure the model for an impulse.
sim(model_name);        % Simulate the system to get output y

himpulse = y/alpha;      % imp. resp. must be rescaled by the input amplitide

clearvars -EXCEPT model_name Ts T K noise noiseOn t alpha himpulse
```

Step response

```
alpha_step = alpha;      % Configure the model for a step.
alpha_impulse = 0;       % Configure the model for a step.
```

```
alpha_sine = 0; w = 0;    % Configure the model for a step.
sim(model_name);          % Simulate the system to get output y
```

To compute the impulse response from a step, we have

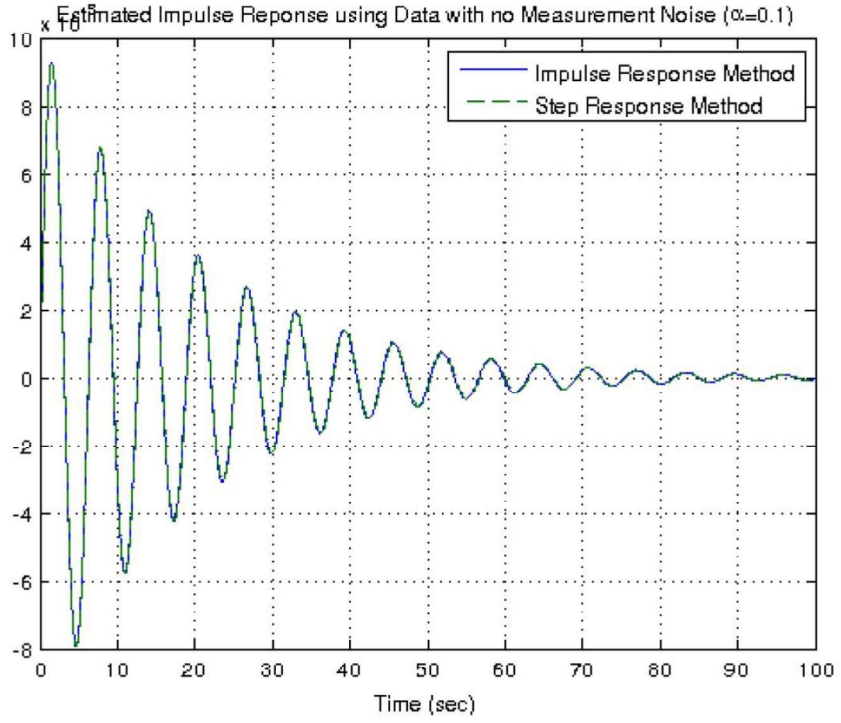
$$\hat{h}(k) = \frac{y(k) - y(k-1)}{\alpha} \qquad \forall k \in \{1, \ldots, N\}$$

and we assume $y(0) = 0$.

```
hstep = (y - [0; y(1:end-1)] ) / alpha;
```

Plot the two step responses.

```
figure
plot(t,himpulse,'-',t,hstep,'--'); grid on; xlabel('Time (sec)');
title(['Estimated Impulse Reponse using Data with '...
       'no Measurement Noise (\alpha=' num2str(alpha) ')']);
legend('Impulse Response Method','Step Response Method',...
        'Location','NorthEast');
```

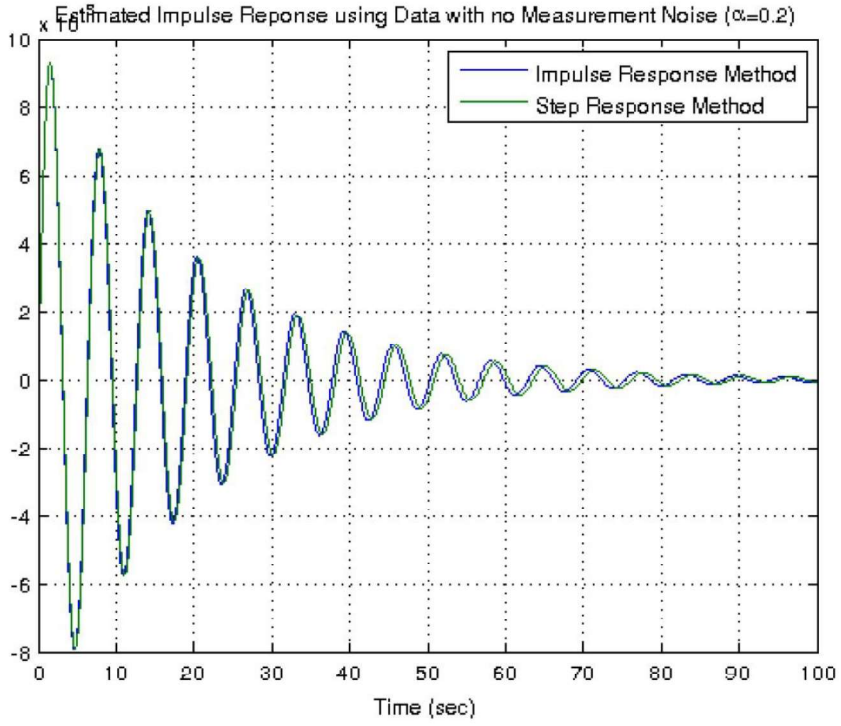



```
clearvars -EXCEPT model_name Ts T K noise noiseOn t alpha himpulse hstep
```

With alpha larger than 0.1, the two estimated impulse responses will start to deviate from each other:

```
alpha = 0.2;
alpha_step = 0;
```

```
alpha_impulse = alpha;
alpha_sine = 0; w = 0;
sim(model_name);
himpulse = y/alpha;
alpha_step = alpha;
alpha_impulse = 0;
alpha_sine = 0; w = 0;
sim(model_name);
hstep = (y - [0; y(1:end-1)] ) / alpha;

figure(2)
plot(t,himpulse,t,hstep); grid on; xlabel('Time (sec)');
title(['Estimated Impulse Reponse using Data with ' ...
       'no Measurement Noise (\alpha=' num2str(alpha) ')']);
legend('Impulse Response Method','Step Response Method',...
    'Location','NorthEast');
```



Part 2.

Turn on the noise in the Simulink block and repeat the identification procedure above for a range of values for α. For both methods, plot the error $\sum_k \|\hat{h}_\alpha(k) - h(k)\|$ vs. α, where $\hat{h}_\alpha(k)$ denotes the estimate obtained for an input with magnitude α. What is your conclusion?

Take the estimate $\hat{h}(k)$ determined in 1 for the noise-free case as the ground truth $h(k)$.

Use the impulse response from Part 1 for reference. Re-compute it and save it.

```
alpha_step = 0;
alpha_impulse = 0.1;
alpha_sine = 0; w = 0;
sim(model_name);
h = y/alpha_impulse;
save my_ir.mat h;

clearvars -EXCEPT model_name Ts T K noise h
```

Turn on the noise:

```
noiseOn = 1;
```

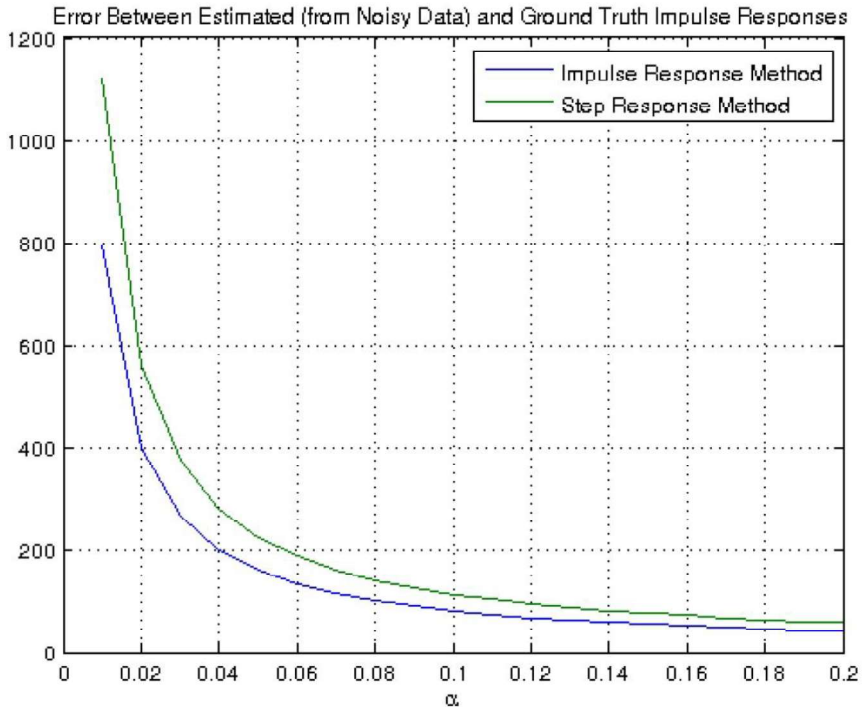
For a range of values of alpha,

1. Compute impulse response for that α
2. Compute error between it and our baseline
3. Compute step response for that α
4. Compute error between it and our baseline

```
alphas = 0.01:0.01:0.2;

for k = 1:length(alphas)
    alpha = alphas(k);

    % impulse
    alpha_step = 0;
    alpha_impulse = alpha;
    alpha_sine = 0; w = 0;
    sim(model_name);
    himpulse = y/alpha;
    EHI(k) = norm(himpulse-h,1);

    % step
    alpha_step = alpha;
    alpha_impulse = 0;
    sim(model_name);
    hstep = (y - [0; y(1:end-1)] ) / alpha;
    EHS(k) = norm(hstep-h,1);
end
```

Plot results.

```
plot(alphas,EHI,alphas,EHS); grid on; xlabel('\alpha');
title(['Error Between Estimated (from Noisy Data) ' ...
       'and Ground Truth Impulse Responses']);
legend('Impulse Response Method','Step Response Method','Location','NorthEast');
```



As expected, in presence of noise, a larger alpha will result in a better estimation (less error). In addition, one can see that the impulse response method always ends up with better results.

Homework 1 Solutions, Problem 2.2 (Correlation Method)

Table of Contents

We're using the correlation method to estimate the transfer function of a system by computing individual points on its Bode plot.

Part 1.

Estimate the system's frequency response $\widehat{H(e^{j\Omega})}$ at a representative set of (log-spaced) frequencies Ω_i using the correlation method.

Select an input amplitude α for which the system remains within the approximately linear region of operation.

Important: Write MATLAB scripts to automate the procedure of taking as inputs the simulation data $u(k)$, $y(k)$ (from a set of .mat files in a given folder) and producing the Bode plot of the system.

```
clc; clear all; close all;
```

___Setup

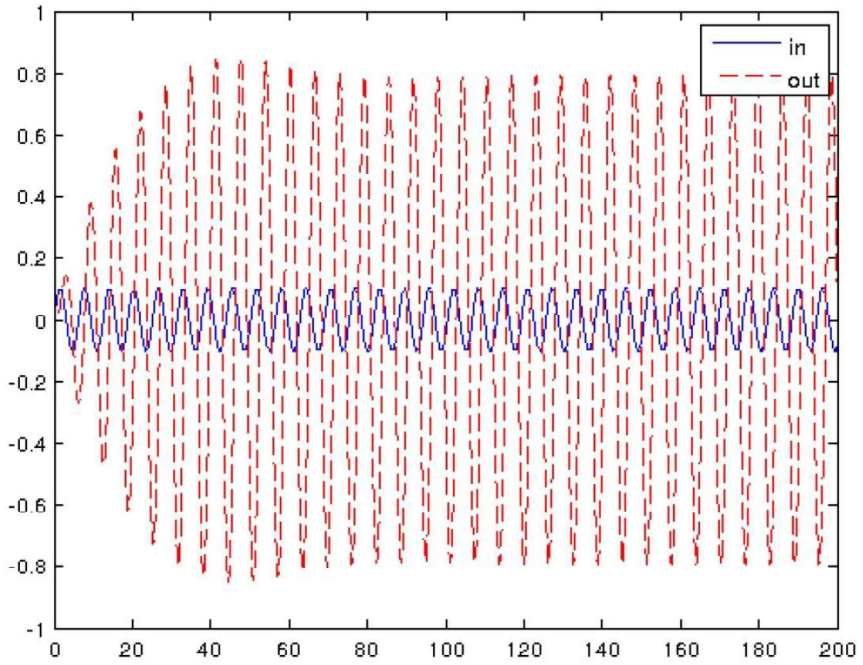
```
model_name = 'P2_mod';
Ts = 0.01;        % Sampling period [sec/sample]
T = 200;          % Duration of simulation [sec]
K  = T/Ts;         % Length of simulation [samp]
noise = 0.0001;  % Noise std dev (used in later parts)
t = 0:Ts:200;     % Time vec [sec]
j = sqrt(-1);


alpha = 0.1;      % This is a good amplitude from Prob 2.1.
```

```
alpha_step = 0;  % Configure the model for sine input.
alpha_impulse = 0;
alpha_sine = alpha;

noiseOn = 0;
```

___Example with a single frequency

Here is an example of the correlation method with $\omega = 1$ rad/sec.

```
w = 1;                % Freq. of cts-time sine block input [rad/sec].
w_eyeballed = w;      % Save w/ distinctive name for later.
f = w/(2*pi);         % Freq. of cts-time sine block input [hz].
Omega = w*Ts;         % Freq. of disc-time zero-orded held sine block input

sim(model_name);

% Estimate gain and phase from the plot:
figure
plot(t,u,'b-',t,y,'r--'); legend('in','out')
```



From this plot, estimate the gain and phase of the TF at this frequency.

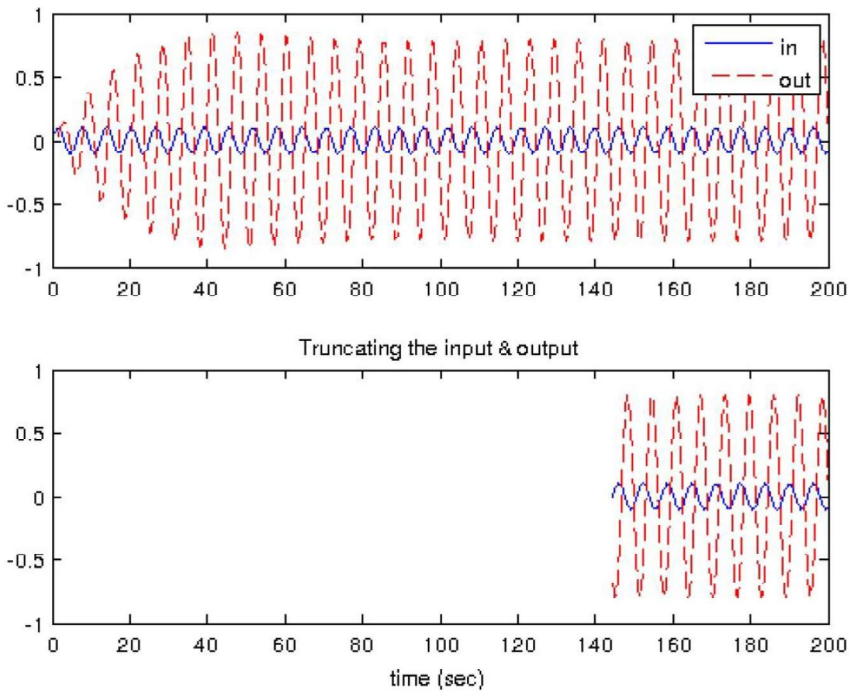
```
% amp: input amplitude: .1 -> output amplitude .791
gain_eyeballed = .791/.1      % abs of H(e^{j*Omega}) (unitless)
```

```
% phase: input peak at 158.7 sec -> output peak at 160.9 sec
phase_eyeballed = (160.9-158.7)*w   % angle of H(e^{j*Omega}) (rad)
```

```
gain_eyeballed =

    7.9100


phase_eyeballed =

    2.2000
```

Now we perform the correlation method. First, throw away all but the last 50 seconds. Make sure you round to an integer of the period so that the cropped input has the same phase as the non-cropped input.

```
% Time near an integer number of cycles:
tpeak = floor( (T-50)*f ) / f;  % seconds

% Convert from to samples:
kpeak = round(tpeak/Ts);  % samples

% Verify this captures about 50 seconds and starts at the same phase as
% the original input

figure
subplot(2,1,1); plot(t,u,'b-',t,y,'r--'); legend('in','out');
subplot(2,1,2); plot( t(kpeak:end), u(kpeak:end), ...
    'b-',t(kpeak:end),y(kpeak:end),'r--'); set(gca,'xlim',[0,T])
xlabel('time (sec)')
title('Truncating the input & output')
```



Truncate the output:

```
ycrop = y(kpeak:end);
```

Correlation method:

```
x = exp(-j * Omega * [1:length(ycrop)] );
Kw = 1/length(ycrop)*(x*ycrop);
H = Kw*2/alpha;

abs(H)
angle(H)
```

```
ans =

    8.0315


ans =

    2.4494
```

How compares to the eyeballed gain & phase?

```
gain_eyeballed
phase_eyeballed
```

```
gain_eyeballed =

    7.9100


phase_eyeballed =

    2.2000
```

Pretty close.

```
clearvars -EXCEPT model_name Ts T K noise t j alpha noiseOn ...
    w_eyeballed gain_eyeballed phase_eyeballed ...
    alpha_step alpha_impulse alpha_sine
```

___Automate over many frequencies

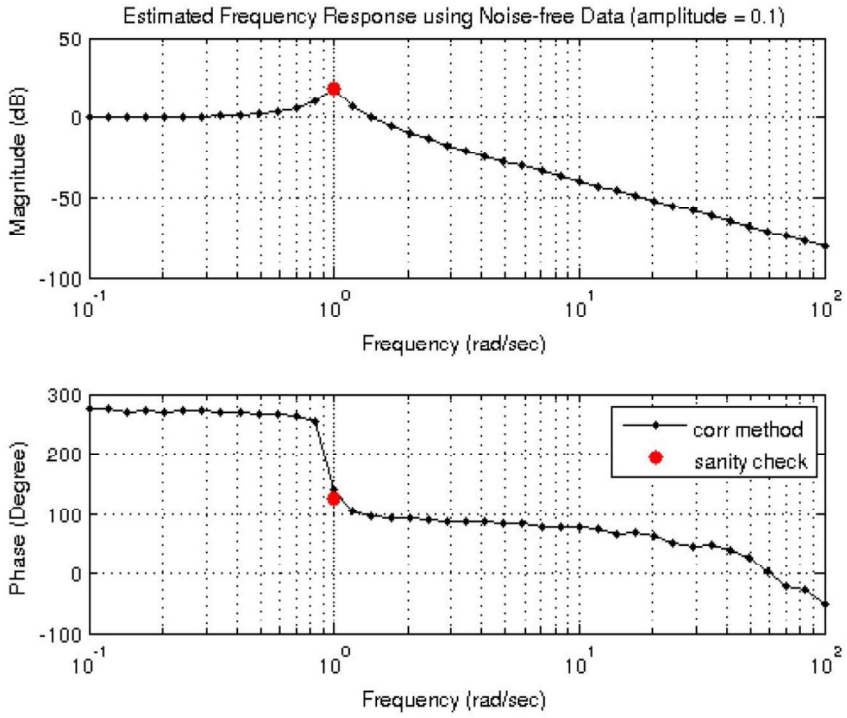
For each freq, do the corr method.

```
num_freqs = 40;
freqs = logspace(-1,2,num_freqs); % List of freqs to try [rad/sec]

H = zeros(size(freqs));

for i = 1:length(freqs)
    w = freqs(i);        % Freq. of cts-time sine block input [rad/sec].
    f = w/(2*pi);        % Freq. of cts-time sine block input [hz].
    Omega = w*Ts;        % Freq. of disc-time zero-orded held sine block input

    sim(model_name);

    % Time near an integer number of cycles:
    tpeak = floor( (T-50)*f ) / f;  % seconds

    % Convert from to samples:
    kpeak = round(tpeak/Ts);  % samples

    % Truncate the output:
    ycrop = y(kpeak:end);

    % Correlation method:
    x = exp(-j * Omega * [1:length(ycrop)] );
    Kw = 1/length(ycrop)*(x*ycrop);
    H(i) = Kw*2/alpha;

end
```

___Plot results

```
Hdb = 20*log10(abs(H));       % gain of H in dB
Hdeg = phase(H)*180/pi+360;   % phase of H in degrees
```

```
gain_eyeballed_db = 20*log10(gain_eyeballed);
phase_eyeballed_deg = phase_eyeballed*180/pi;

figure;
subplot(2,1,1); semilogx(freqs,Hdb,'k.-'); grid on;
xlabel('Frequency (rad/sec)'); ylabel('Magnitude (dB)');
title(['Estimated Frequency Response using Noise-free Data (amplitude = ' num2str(
subplot(2,1,2); semilogx(freqs,Hdeg,'k.-'); grid on;
xlabel('Frequency (rad/sec)'); ylabel('Phase (Degree)');

subplot(2,1,1); hold on; semilogx(w_eyeballed,gain_eyeballed_db,'r.','markersize',
subplot(2,1,2); hold on; semilogx(w_eyeballed,phase_eyeballed_deg,'r.','markersize
legend('corr method','sanity check')
```



Part 2.

Turn on the noise in the Simulink block and repeat the identification procedure above for a range of values for α. Plot the error $\sum_i \left\| \widehat{H_\alpha(e^{j\Omega_i})} - H(e^{j\Omega_i}) \right\|$ denotes the estimate obtained for an input with magnitude α. What is your conclusion?

Take the estimate $\widehat{H(e^{h\Omega})}$ determined in Part 1 for the noise-free case as the ground truth $H(e^{j\Omega})$.

```
clearvars -EXCEPT model_name Ts T K noise j H alpha_impulse ...
```

```
    alpha_step alpha_sine

Htrue = H; clear H;

noiseOn = 1;
alphas = [0.001:0.02:0.19 0.2];

tf_error = zeros(size(alphas));
```

For each alpha, perform the correlation method to approximate the TF.

```
for m=1:length(alphas)

    alpha = alphas(m);
    alpha_step = 0;  % Configure the model for sine input.
    alpha_impulse = 0;
    alpha_sine = alpha;

    % Copy-pasta from Part 1.

    num_freqs = 40;
    freqs = logspace(-1,2,num_freqs); % List of freqs to try [rad/sec]

    H = zeros(size(freqs));

    for i = 1:length(freqs)
        w = freqs(i);        % Freq. of cts-time sine block input [rad/sec].
        f = w/(2*pi);        % Freq. of cts-time sine block input [hz].
        Omega = w*Ts;        % Freq. of disc-time zero-orded held sine block input

        sim(model_name);

        % Time near an integer number of cycles:
        tpeak = floor( (T-50)*f ) / f;  % seconds

        % Convert from to samples:
        kpeak = round(tpeak/Ts);  % samples

        % Truncate the output:
        ycrop = y(kpeak:end);

        % Correlation method:
        x = exp(-j * Omega * [1:length(ycrop)] );
        Kw = 1/length(ycrop)*(x*ycrop);
        H(i) = Kw*2/alpha;

    end

    % End copy-pasta.

    tf_error(m) = sum(abs(H-Htrue));

end
```

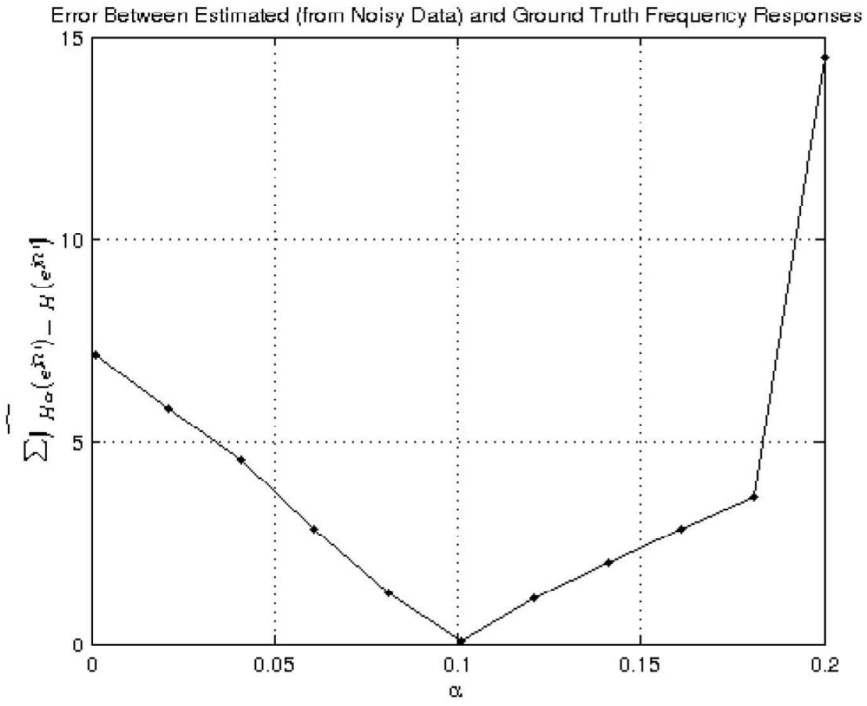
Expected behavior:

- For small alpha, the noise will dominate so the identified TF will deviate from the noiseless model from Part 1.

- For large alpha, the system will be nonlinear and the identified TF will deviate from the noiseless model from Part 1.

Plot results.

```
figure;
plot(alphas,tf_error,'k.-'); grid on;
xlabel('\alpha');
ylabel('$\sum_i\|\widehat{H_\alpha(e^{j\Omega_i})}-H(e^{j\Omega_i})\|$',...
    'interpreter','latex');
title(['Error Between Estimated (from Noisy Data) ' ...
    'and Ground Truth Frequency Responses']);
```



Up to 0.1, a larger alpha will result in a better estimation (less error). After that, error will increase due to invalid linearity assumption.

Part 3.

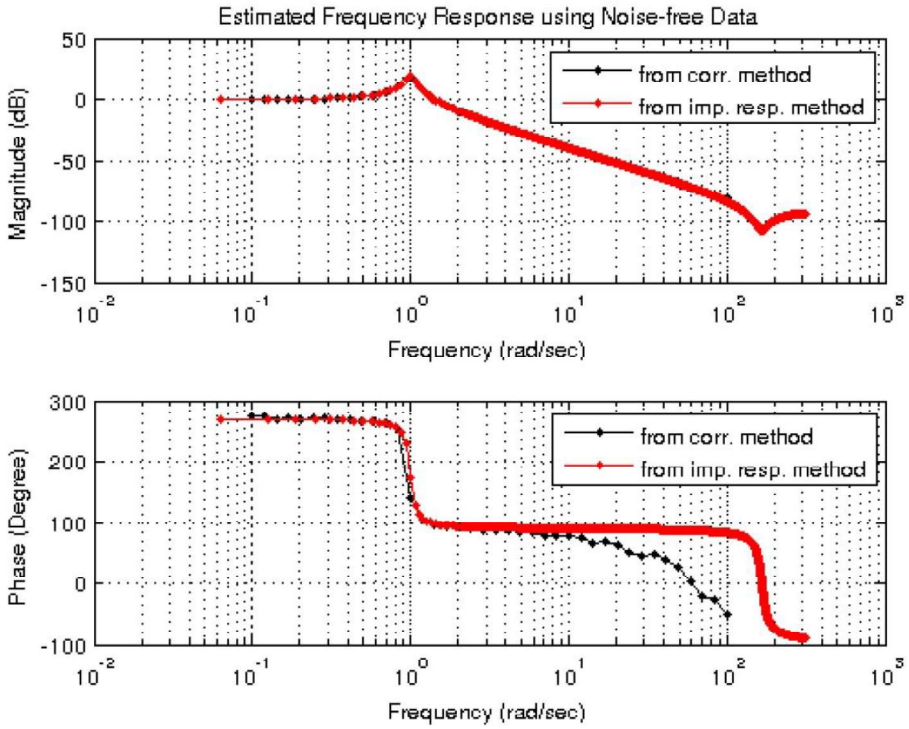
Compare your best frequency response estimate $H(e^{j\Omega})$ with the frequency response that you would obtain by taking the Fourier transform of the best impulse response $\hat{h}(k)$ that you obtained in Exercise 2.1.

Hint: Use the MATLAB command fft to compute the Fourier transform. This command gives you $H(e^{j\Omega})$ from $\Omega = 0$ to $\Omega = 2\pi$ so you should discard the second half of the Fourier transform (corresponding to $\Omega > \pi$).

```
clearvars -EXCEPT Htrue freqs Ts

load('my_ir.mat','h');

Hfft   = fft(h);
Hfft   = Hfft(1:round(length(Hfft)/2));  % take the first half (0 to pi)
N      = length(Hfft);
Omegas = 0:pi/(N-1):pi;    % the FFT's samples are index by rad/sample
w      = Omegas/Ts;        % convert FFT's indices to rad/sec


Hfft_db = 20*log10(abs(Hfft));
Hfft_deg = phase(Hfft)*180/pi+360-90;  % Guess: -90deg due to some idiosynchracy
```

Compare.

```
Hdb = 20*log10(abs(Htrue));       % gain of H in dB
Hdeg = phase(Htrue)*180/pi+360;   % phase of H in degrees

figure;

subplot(2,1,1); semilogx(freqs,Hdb,'k.-'); grid on;
hold on; semilogx(w,Hfft_db,'r.-');
legend('from corr. method','from imp. resp. method');
xlabel('Frequency (rad/sec)'); ylabel('Magnitude (dB)');
title(['Estimated Frequency Response using Noise-free Data']);

subplot(2,1,2); semilogx(freqs,Hdeg,'k.-'); grid on;
hold on; semilogx(w,Hfft_deg,'r.-');
legend('from corr. method','from imp. resp. method');
xlabel('Frequency (rad/sec)'); ylabel('Phase (Degree)');
```



As it can be seen, compared to the Fourier Transform of our estimated impulse response, the estimated frequency response is quite reasonable.