SEA Project Report 2

Yi Yang

09/26/2016

This week, I will use some data acquired from the experiment and compare the theoretical curves with the experimental curves.

Test 1

For data from $SEA\_Free\_K334\_kp10.mat$, I use the following codes:

```
%% This is the program to apply experimental data to our linear matlab model
% estimate the transfer function
% by Yi Yang
% Date 9/26/2016
%--------------------------------------------------------------------------
% Admittance Control Devices
clear all; close all;
load('SEA_Free_K334_kp10.mat');
s = tf('s');
dataID = sysID();
Z_h = dataID.M * s + dataID.B; % the inherent impedance of haptic devices
Kve = 0; % [N/m], virtual environment spring constant
Z_ve = Kve/s; % impedance of virtual environment
C = 10; % portional control law
Z_padm = (dataID.n * C * Z_ve + Z_h)/(dataID.n * C + 1);

% Series Elastic Actuator Device
k = 334; % [N/m], Physical spring constant
kk = k/s + dataID.b;
Z_e = dataID.m * s;
Z_psea = (Z_e * (Z_h + k * dataID.n * C/s + kk) + kk * (Z_h + dataID.n * C * Z_ve))/(Z_h + ...
    k * dataID.n * C/s + kk);

% This is the free space simulation using the data acquired from SEA_Free_K334_kp10.mat
format long;
Ts=(timeStamp(end - 1)-timeStamp(1))/(length(timeStamp)-2)/(10^6);
LinTime = timeStamp(1)/(10^6):Ts:timeStamp(end - 1)/(10^6);
Resample_RevPosB = interp1(timeStamp(1:end-1)/(10^6), RevPosB(1:end-1), LinTime);
Fuser = k * (Resample_RevPosB)/1000.0; % [N] ([N/m]*[m])
Uo = lsim(1/Z_psea, Fuser, LinTime);
figure(4);
hold on;
plot(LinTime, Fuser, 'b--');
plot(LinTime, Uo, 'r');
legend('Force', 'Velocity');
title('Simulation');
grid on;
box on;
hold off;

```

```
% estimate the transfer function
Fs = 1/Ts;
NFFT = 2^nextpow2(length(LinTime));
[Z_e_sea, freq2] = tfestimate(Uo, Fuser, [], [], NFFT, Fs);

[MagTh_sea, PhaseTh_sea, FreqTh_sea] = bode(Z_psea, {2*pi*1e-1, 2*pi*1e3});
Mag_sea = zeros(length(MagTh_sea), 1);
Freq_sea = FreqTh_sea./(2*pi);
for i = 1:length(MagTh_sea)
    Mag_sea(i, 1) = MagTh_sea(1, 1, i);
    Mag_sea(i, 1) = 20 * log10(Mag_sea(i, 1));
end
figure(5);
% bodemag(Z_ve, '--', opts);
hold on;
semilogx(Freq_sea, Mag_sea, 'r');
semilogx(freq2, mag2db(abs(Z_e_sea)));
set(gca,'xscale','log');
legend('the', 'exp');
title('SEA');
grid on;
box on;
hold off;
```

The plots for simulation and impedance are shown below:

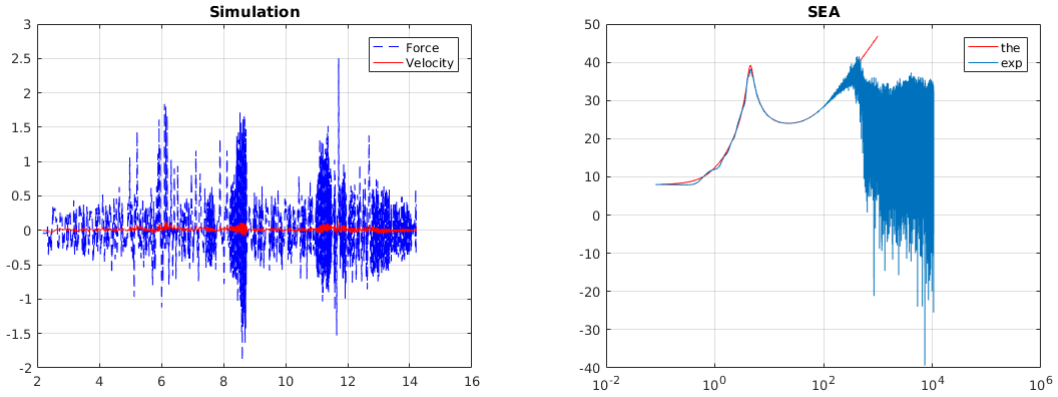

Figure 1: Simulation of Linear Matlab Model

Remark:

- the experiment data I use for the above test experiment is $RevPosB$, and use it to calculate user force $Fuser$
- I apply $Fuser$ to the linear matlab model, using $lsim$ function to get the output data U_o
- The last step is to estimate the transfer function using $Fuser$ and U_o.

Next, I will estimate transfer function from input and output data sets that are all measured from experiments (real data).

Test 2

For admittance device, we use data set with $K_p = -3$ and $K_{ve} = 400$. Matlab codes are shown below:

```
%% Experiemnt data demonstration
% by Yi Yang
% Date: 9/30/2016
% data set: adm_Kp-3_Kev100
%--------------------------------------------------------------------------
clear all;
close all;
struct = importdata('test/20160929105124_adm_Kp=-3_Kimg=400.txt');
data = struct.data(:,:);
timeStamp = data(:, 1);
LoadCellForceA = data(:, 2);
MotorOutputA = data(:, 3);
RevPosA = data(:, 4);
RevPosB = data(:, 5);
CurrentA = data(:, 6);

% Admittance theoretical model
s = tf('s');
dataID = sysID();
Z_h = dataID.M * s + dataID.B; % the inherent impedance of haptic devices
Kve = 400; % [N/m], virtual environment spring constant
Z_ve = Kve/s; % impedance of virtual environment
C = -3; % portional control law
Z_padm = (dataID.n * C * Z_ve + Z_h)/(dataID.n * C + 1);

% Model constructed by experiment data
format long;
Ts=(timeStamp(end)-timeStamp(1))/(length(timeStamp)-1)/(10^5);
LinTime = timeStamp(1)/(10^5):Ts:timeStamp(end)/(10^5);
LinTime = LinTime';
% Resample_RevPosB = interp1(timeStamp(1:end-1)/(10^6), RevPosB(1:end-1), LinTime);
% Fuser = k * (Resample_RevPosB)/1000.0; % [N] ([N/m]*[m])
% Uo_sea = lsim(1/Z_psea, Fuser, LinTime);
Resample_RevPosA = interp1(timeStamp(1:end)/(10^5), RevPosA(1:end), LinTime);
Resample_LoadCellForceA = interp1(timeStamp(1:end)/(10^5), LoadCellForceA(1:end), LinTime);
% First order Euler backward difference methods to approximate velocity,
% [mm] -> [m]
V = diff([Resample_RevPosA(1); Resample_RevPosA])./diff([LinTime(1) - Ts; LinTime])/1000;
figure(1);
plot(LinTime, Resample_LoadCellForceA);
legend('LoadCellForceA');
figure(2);
plot(LinTime, Resample_RevPosA);
legend('RevPosA');
figure(3);
plot(LinTime, V);
legend('VelocityA');
% Then I will use experiment data to estimate system impedance
Fs = 1/Ts;
NFFT = 2^nextpow2(length(LinTime));
[Z_e_adm, freq_e] = tfestimate(V, Resample_LoadCellForceA, [], [], NFFT, Fs);
% Sample experiment and theoretical transfer function
[MagTh_adm, PhaseTh_adm, FreqTh_adm] = bode(Z_padm, {2*pi*3e-1, 2*pi*2e4});
Mag_adm = zeros(length(MagTh_adm), 1);
Freq_adm = FreqTh_adm./(2*pi);
for i = 1:length(MagTh_adm)
    Mag_adm(i, 1) = MagTh_adm(1, 1, i);
    Mag_adm(i, 1) = 20 * log10(Mag_adm(i, 1));
end
figure(4);
hold on;
semilogx(Freq_adm, Mag_adm, 'r');
semilogx(freq_e, mag2db(abs(Z_e_adm)));
set(gca,'xscale','log');
```

```
65  legend('the', 'exp');
66  title('Admittance');
67  grid on;
68  box on;
69  hold off;
70
71  % Pos_des = Resample_LoadCellForceA/Kve;
72  % V_des = diff([Pos_des(1); Pos_des])./diff([LinTime(1)-Ts; LinTime]);
73  % figure(5);
74  % [V_des_sorted, sortID] = sort(V_des);
75  % V_sorted = V(sortID);
76  % plot(V_des_sorted, V_sorted, 'r-');
77  % xlabel('$\dot{x}_{cmd}$');
78  % ylabel('$\dot{x}_{act}$');
79
80  % figure(4);
81  % hold on;
82  % plot(LinTime, Fuser, 'b--');
83  % plot(LinTime, Uo, 'r');
84  % legend('Force', 'Velocity');
85  % title('Simulation');
86  % grid on;
87  % box on;
88  % hold off;
```

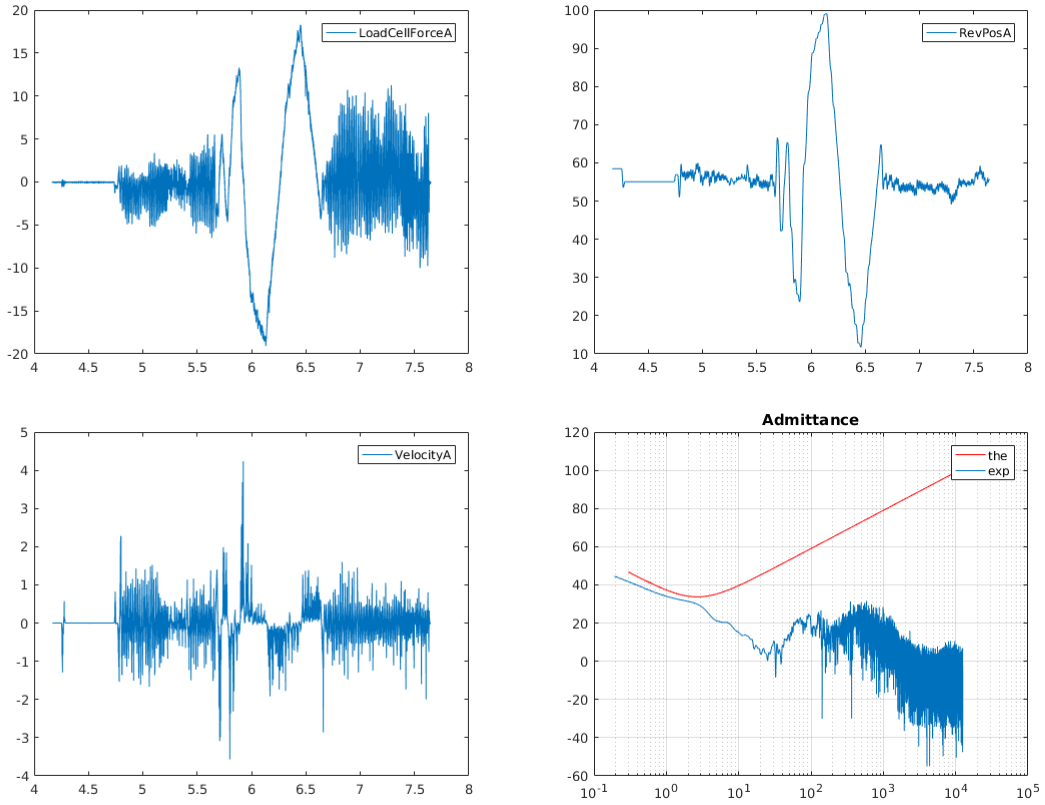

Figure 2: LoadCellForceA, RevPosA, Actual VelocityA, Driving Point Impedance

Test 3

For SEA device, we use data sets with $K_p = -3$, $K_{ve} = 400$ and $K_{spring} = 1200$. Matlab codes shown below:

```
%% Experiment data demonstration for SEA
% by Yi Yang
% Date: 10/01/2016
% data set: sea_Kp=-3_Kimg=400
%--------------------------------------------------------------------------
clear all;
close all;
struct = importdata('test/20160929114713_sea_Kp=-3_Kimg=400.txt');
data = struct.data(:,:);
timeStamp = data(2:end, 1);
LoadCellForceA = data(2:end, 2);
MotorOutputA = data(2:end, 3);
RevPosA = data(2:end, 4);
RevPosB = data(2:end, 5);
CurrentA = data(2:end, 6);

% Construct SEA and ADM theoretical model
s = tf('s');
dataID = sysID();
Z_h = dataID.M * s + dataID.B; % the inherent impedance of haptic devices
Kve = 400; % [N/m], virtual environment spring constant
Z_ve = Kve/s; % impedance of virtual environment
C = -3; % portional control law
Z_padm = (dataID.n * C * Z_ve + Z_h)/(dataID.n * C + 1);

% Series Elastic Actuator Device
k = dataID.k; % [N/m], Physical spring constant
kk = k/s + dataID.b;
Z_e = dataID.m * s;
Z_psea = (Z_e * (Z_h + k * dataID.n * C/s + kk) + kk * (Z_h + dataID.n * C * Z_ve))/(Z_h + ...
    k * dataID.n * C/s + kk);

% Model constructed from experiment data
format long;
Ts=(timeStamp(end)-timeStamp(1))/(length(timeStamp)-1)/(10^5);
LinTime = timeStamp(1)/(10^5):Ts:timeStamp(end)/(10^5);
LinTime = LinTime';
Resample_RevPosA = interp1(timeStamp(1:end)/(10^5), RevPosA(1:end), LinTime);
Resample_RevPosB = interp1(timeStamp(1:end)/(10^5), RevPosB(1:end), LinTime);
Fuser = k * (Resample_RevPosB)/1000.0; % [N] ([N/m]*[m])
% Uo = lsim(1/Z_psea, Fuser, LinTime);
% First order Euler backward difference methods to approximate velocity
V = diff([Resample_RevPosA(1)+Resample_RevPosB(1); ...
    Resample_RevPosA+Resample_RevPosB])./diff([LinTime(1) - Ts; LinTime])/1000;
figure(1);
plot(LinTime, Fuser);
legend('Fuser');
figure(2);
plot(LinTime, Resample_RevPosA+Resample_RevPosB);
legend('RevPosA+RevPosB');
figure(3);
plot(LinTime, V);
legend('Velocity');

% estimate the transfer function
Fs = 1/Ts;
NFFT = 2^nextpow2(length(LinTime));
[Z_e_sea, freq_e] = tfestimate(V, Fuser, [], [], NFFT, Fs);

```

```
[MagTh_sea, PhaseTh_sea, FreqTh_sea] = bode(Z_psea, {2*pi*3e-1, 2*pi*2e4});
Mag_sea = zeros(length(MagTh_sea), 1);
Freq_sea = FreqTh_sea./(2*pi);
for i = 1:length(MagTh_sea)
    Mag_sea(i, 1) = MagTh_sea(1, 1, i);
    Mag_sea(i, 1) = 20 * log10(Mag_sea(i, 1));
end
figure(4);
% bodemag(Z_ve, '--', opts);
hold on;
semilogx(Freq_sea, Mag_sea, 'r');
semilogx(freq_e, mag2db(abs(Z_e_sea)));
set(gca,'xscale','log');
legend('the', 'exp');
title('SEA');
grid on;
box on;
hold off;

Pos_des = Fuser/Kve;
V_des = diff([Pos_des(1); Pos_des])./diff([LinTime(1)-Ts; LinTime]);
figure(5);
% plot(V_des, V);
[V_des_sorted, sortID] = sort(V_des);
V_sorted = V(sortID);
plot(V_des_sorted, V_sorted, 'r-');
xlabel('$\dot{x}_{cmd}$');
ylabel('$\dot{x}_{act}$');

% figure(4);
% hold on;
% plot(LinTime, Fuser, 'b--');
% plot(LinTime, Uo, 'r');
% legend('Force', 'Velocity');
% title('Simulation');
% grid on;
% box on;
% hold off;
```

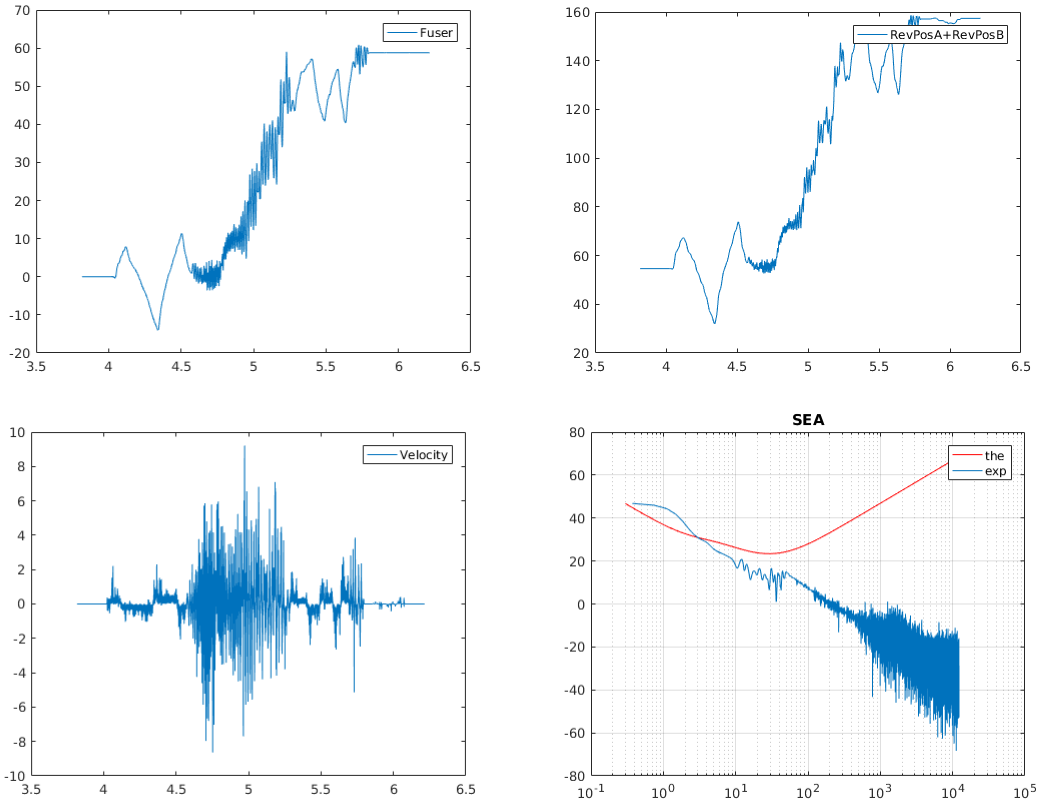

Figure 3: Fuser, Actual Position, Actual Velocity, Driving Point Impedance

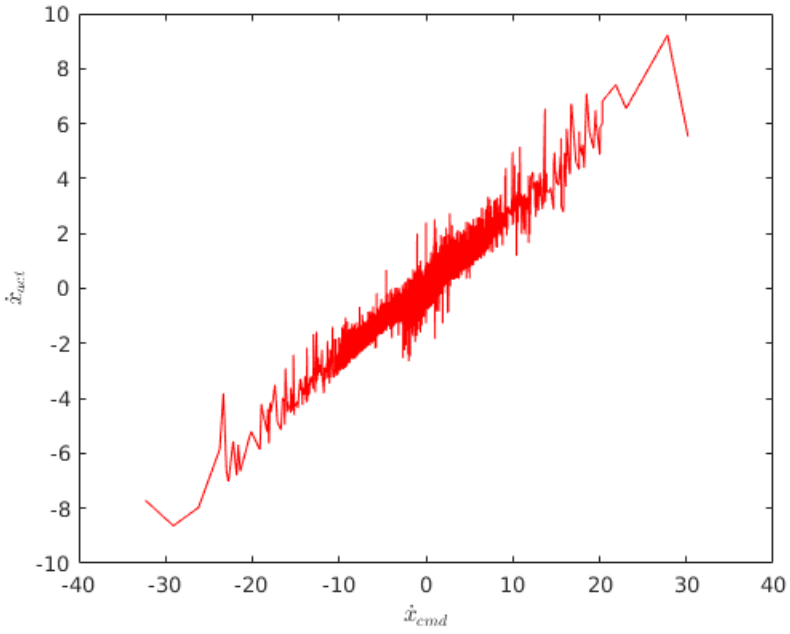

Figure 4: Actual Velocity Vs Command Velocity