23 Identification of a Response Amplitude Operator from Data

Load the data file `homework5.dat` from the course website. The first column is time t in seconds, the second column is a measured input signal $u(t)$, in Volts, and the third column is a measured output signal $y(t)$, also in Volts. $u(t)$ has no significant frequency content above $2rad/s$.

Your main task is to estimate the Response Amplitude Operator (RAO) $H(\omega)$ and the underlying transfer function $G(j\omega)$ of the system that created $y(t)$ when driven by $u(t)$. Your deliverables are:

1. A plot of the data-based RAO $H(\omega)$ versus frequency.
2. The specific transfer function that best describes the data, e.g., $G(j\omega) = 6.5/(j\omega+2.2)$. Recall that $H(\omega) = |G(j\omega)|^2$.
3. A plot of the step response of $G(j\omega)$, making the assumption that the system is causal.

Some hints:

- The recommended overall procedure is to calculate the autocorrelation function for each signal, compute the Fourier transforms of these, and then divide according to the Wiener-Khinchine relation so as to recover an empirical $H(\omega) = |G(j\omega)|^2$.
- A detailed discussion on use of the FFT in MATLAB is given in the worked problem entitled *Dynamics Calculations Using the Time and Frequency Domains*. This includes specification of the frequency vector that goes with an FFT, which is very important for the current problem.
- The FFT of an autocorrelation function is supposed to be all real because $R(\tau)$ has only the cosine phases represented. Yet look at the signal `fft(cos(t));` and you will see that it not only has some imaginary parts, but also a lot of content NOT at frequency one. Some of this appears because numerical Fourier transforms assume that the time-domain signal is periodic - that the end reconnects with the beginning. This often gives a discontinuity, which the transform tries to take care of with the extra stuff you see. Some of this messiness also comes up because the FFT is given only at discrete frequencies. When your signal actually has other frequencies, the FFT spreads things out a bit.

  The first problem can be reduced by multiplying your time-domain signal by a cosine window such as $1/2 * (1 - cos(2 * pi * [0 : n - 1]/(n - 1)))$. When you plot it out, you can see directly that there won't be any discontinuity between the beginning and end. The second problem above doesn't affect the answers you get on this homework; I assure you that the RAO is pretty smooth.xt

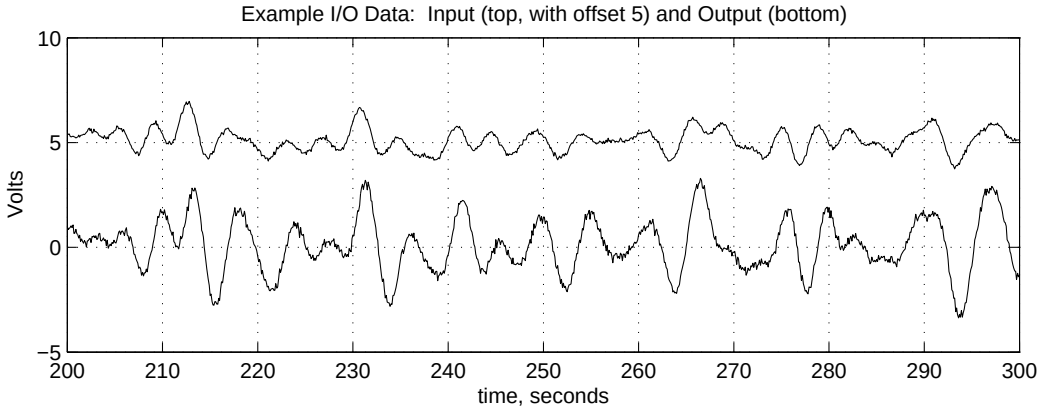
- Super-duper hint: The true transfer function has either no zeros or one zero (which could be in the right-half plane), and one or two poles, both in the left-half plane (since the system is clearly stable). It will take a little effort to get a good fit with your data, because you are searching in four parameters. But think about the damping ratio and undamped frequency and you will already be in the right ballpark. You might even like to automate the fit procedure by writing it as an optimization problem!

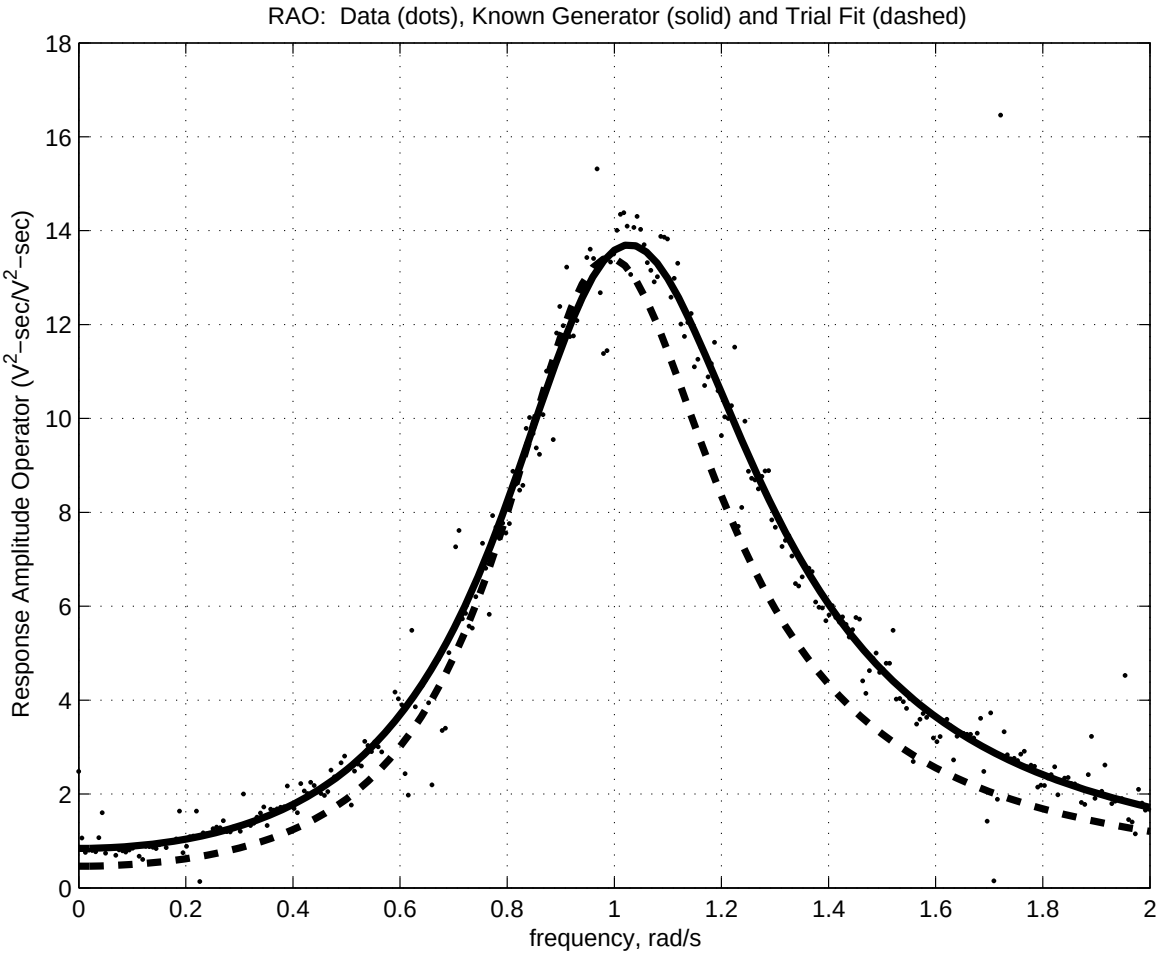
Solutions

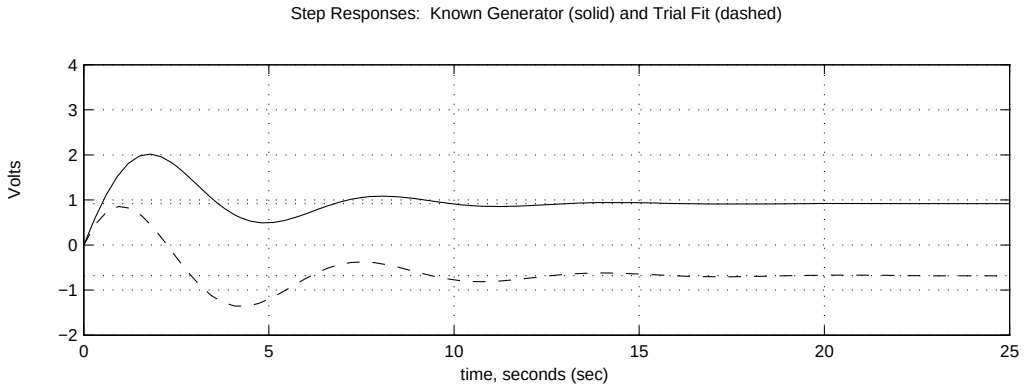
1. See the attached code and figures. The input is quite broad-band: in fact, the frequency content is uniform from zero to two radians per second, and zero above (except for the added noise). There is a resonance effect going on at around a six-second period, with fairly light damping. The empirical RAO is the *dots* in the second figure; the lines are the known and trial fits of specific transfer functions. The low-frequency RAO is about one, and at high frequencies it goes to zero (although the plot given doesn't go to high enough frequencies to see this). I note that because the input has no frequency content above $2 rad/s$, the sensor noise will muddy everything up above this frequency, and the RAO is basically undefined in this range.

2. The generating transfer function is $G(j\omega) = (2j\omega + 1)/(-\omega^2 + 0.6j\omega + 1.09)$, and it is causal so you can replace the $j\omega$'s with s's if you like. This has a left-half-plane zero at $s = -0.5$, and (of course) two lightly-damped poles near $s = \pm j$. The magnitudes of this transfer function fit the RAO from data quite well. A second fit is plotted also, $G_2(j\omega) = (1.7j\omega - 0.68)/(-\omega^2 + 0.5j\omega + 1)$, which is based on the trying to match the natural frequency and damping ratio first, and then looking at the magnitude and a possible zero.

3. The step response of the system is a trick question. The RAO only tells us the magnitude information of the system, and nothing about the phase. Notice the sign change in the numerator of G_2; it fits the RAO just fine, but is a somewhat different system when you look at the step response!


Example I/O Data: Input (top, with offset 5) and Output (bottom)
Volts
time, seconds
10
5
0
−5
200
210
220
230
240
250
260
270
280
290
300


RAO: Data (dots), Known Generator (solid) and Trial Fit (dashed)
Response Amplitude Operator (V2−sec/V2−sec)
frequency, rad/s
0
2
4
6
8
10
12
14
16
18
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2


Step Responses: Known Generator (solid) and Trial Fit (dashed)
Volts
time, seconds (sec)
4
3
2
1
0
−1
−2
0
5
10
15
20
25

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Extract an RAO from I/O data
% MIT 2.017 FSH October 2009

clear all ;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This first part of code generates the noisy i/o data
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

dt = .1 ; % time step, sec
t = 0:dt:10000*dt; % time vector

%dw = .02 ; % frequency increment, rad/s
dw = .04 ; % frequency increment, rad/s

w = dw:dw:100*dw ; % frequency vector
w = w + randn(size(w))*dw/5 ; % randomize a little to prevent cycling

a = .08*ones(size(w)) ; % amplitudes (uniform)

phi = 2*pi*rand(1,length(w)); % random phase angles

%uNoise = .1 ; % input noise level
%yNoise = .1 ; % output noise level
uNoise = .001 ; % input noise level
yNoise = .001 ; % output noise level

% build up the input signal
n = length(t) ;
u = zeros(1,n) ;
for i = 1:length(w),
    u = u + a(i)*sin(w(i)*t + phi(i)) ;
end;
u = u + randn(1,n).*std(u)*uNoise ; % add noise to the input

% to make the output, define an LTI system, and run input through it
%sys = zpk(-0.5,[-.3+sqrt(-1), -.3-sqrt(-1)],2);
sys = zpk([],[-1,0],2);
y = lsim(sys,u,t) ;
y = y' ;
y = y + randn(1,n).*std(y)*yNoise ; % add noise to output

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% Here is where the student's work begins
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

figure(1);clf;hold off;subplot(211);
plot(t,u+5,'k',t,y,'k') ;
a = axis ; axis([200 300 a(3:4)]); % show just a part
title('Example I/O Data:  Input (top, with offset 5) and Output (bottom)');
xlabel('time, seconds');
ylabel('Volts');
grid;

% calculate the autocorrelation functions
Ru = zeros(n,1);
Ry = zeros(n,1);
for i = 1:n,
    Ru(i) = 1/(n-i+1)*sum( u(1:(n+1-i)).*u(i:n)) ;
    Ry(i) = 1/(n-i+1)*sum( y(1:(n+1-i)).*y(i:n)) ;
end;

% multiply each R by a full cosine window to clean up the spectra
Ru = Ru * 1/2 .*(1-cos([0:1:n-1]/(n-1)*2*pi))';
Ry = Ry * 1/2 .*(1-cos([0:1:n-1]/(n-1)*2*pi))';

% here are the spectra
fRu = real(fft(Ru)) ; % (throw out imag part because Ru and
fRy = real(fft(Ry)) ; % Ry are known to only have cosine phase)

fw = [0:1:n-1]/n*2*pi/dt ; % frequency vector to go with fRu and fRy

% A key point here is that the scaling of the fft is the same for both
% fRy and fRu, so their division will give the correct results for the
% RAO

figure(2);clf;hold off;
%plot(fw,abs(fRy./fRu),'k.');
semilogy(fw,abs(fRy./fRu),'k.');
ylabel('Response Amplitude Operator (V^2-sec/V^2-sec)');
xlabel('frequency, rad/s');

% put the known |tf|^2 on top of the RAO data, for reference
%wb = [0:.02:2];
wb = 0:dw:max(w);
[magbdum,phaseb] = bode(sys,wb);
magb(1,1:length(wb)) = magbdum(1,1,:);
```

```
hold on;
%plot(wb,magb.^2,'k','LineWidth',3);
semilogy(wb,magb.^2,'k','LineWidth',3);
grid;

% put also a trial fit based mainly on damping ratio and natural frequency;
% also, we have to add a zero and get the gain right to make things work out
%zeta = .25 ;
%wn = 1 ;
zeta = 5 ;
wn = .1 ;
sys2 = zpk([],roots([1 2*zeta*wn wn^2]),2.2);
[magbdum2,phaseb2] = bode(sys2,wb);
magb2(1,1:length(wb)) = magbdum2(1,1,:);  % (transform a 3d to a 2d array)
%plot(wb,magb2.^2,'k--','LineWidth',3);
semilogy(wb,magb2.^2,'k--','LineWidth',3);
%a = axis ; axis([min(wb) max(wb) 0 18]); % don't show anything above wb
a = axis ; axis([min(wb) max(wb) 0 1e6]); % don't show anything above wb

title('RAO:  Data (dots), Known Generator (solid) and Trial Fit (dashed)');

figure(3);clf;hold off;subplot(211);
%step(sys,'k',30);
step(sys,'k',5);
hold on;
%step(sys2,'k--',30);
step(sys2,'k--',5);
grid;
xlabel('time, seconds');
ylabel('Volts');
title('Step Responses:  Known Generator (solid) and Trial Fit (dashed)');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

2.017J Design of Electromechanical Robotic Systems
Fall 2009