Table of Contents

an example of sine sweep identification of frequency response of a KNOWN plant, so that the answer can be verified

```
Prabir Barooah
I have coded the algorithm and verified that it works, meaning, it estimat
the frequency response of the plant quite accurately even with large amoun
sensor noise, as long as the parameters that are up to the designer (trans
number of periods to average over etc.) are chosen appropriately. You only
appropriate values of those parametrs and hit the run button.
1. The default sine sweep parameters  I have included below
ARE DELIBERATELY POORLY CHOSEN.  YOU WILL HAVE TO DO A FEW SINGLE-SINE EXP
(USE THE FIRST SEGMENT OF THE SCRIPT MARKED "PRELIMINARY INVESTIGATION")
-- AND THINK -- TO FIGURE OUT APPROPRIATE VALUES
2. The script estimates the frequency response of the plant P(jw) at one
specific frequency. You will need to modify the script a tiny bit to
repeat the estimation at a number of frequency values.
```

```
clear all

conductSingleExp = 1;
conductSineSweepExp = 1; %specify the parameters(see below) before you
 set this to 1, otherwise
% the script will not run

%%------ specify the plant ----
% You can of course try with a different plant, but I'd suggest that
 at
% first you do not touch this part.
P_TF = tf([10, 2],[1 1 4.25]);
[num,den]  = tfdata(P_TF,'v');
[A,B,C,D] = tf2ss(num,den);
Plant = ss(A,B,C,D);
n = size(A,1);
```

```
A_u = 1; %Amplitude of input sinusoid
%%------
```

user inputs (common):

you can vary these parameters: start by setting them to 0, which will make the system identification task easier. Then you should increase them and check

```
%how large you can make them and still identify the plant accurately
 (you
%will have to decide for yourself how accurate is accurate enough)
std_dev_noise  = 0.1; %standard deviation of the sensor noise
X0factor = 10;
```

more user inputs

```
if conductSingleExp==1
    Ts = 1/550;
    freqSingleExp = 100; %rad/time unit
end
```

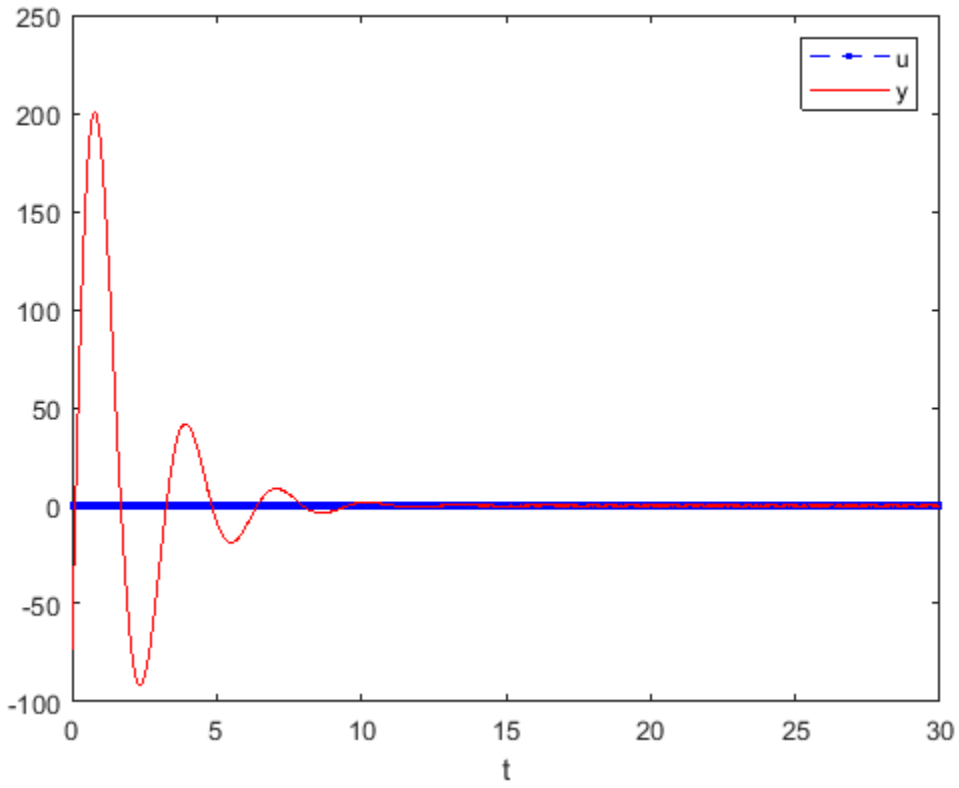
user inputs (for sine sweep only)

```
if conductSineSweepExp==1
    %--------------------------
    omega_array =
 [100,10,0.08,0.7,0.4,0.9,0.6,0.02,15,40,65,89,1,450,3,260,3.20]; %it
 needs to be a row or column vecor of frequencies in
    % "radians/time unit"
    %--------------------------

    %--------------------------
    decay_time = [4]; %in "time units"
    %--------------------------

    %--------------------------
    num_cycles2average_nom = [100]; %needs to be a positive integer
    %--------------------------

    Fs = [300]; %sampling period in "samples/time unit"
    % choose it large enough to give you enough samples in one period
 of the sinusoid
    % (depends on the highest frequency you use, of course), but not
 so
    % high that the simulation takes foreever.
    Ts = 1/Fs;

end
```

----- PRELIMINARY INVESTIGATION - ---

RESPONSE TO SINE INPUT AT SOME ARBITRARY FREQUENCY

```
if conductSingleExp ==1
    x0 = X0factor*randn(n,1); %somewhat random initial state
    time = [0:Ts:30]';
    noise = randn(length(time),1)*std_dev_noise;
    u = A_u*sin(freqSingleExp*time);
    y_at_omega = lsim(Plant,u,time,x0)+noise;

    figure
    plot(time,u,'b.--',time,y_at_omega,'r');
    xlabel('t'); legend('u','y');
end
```



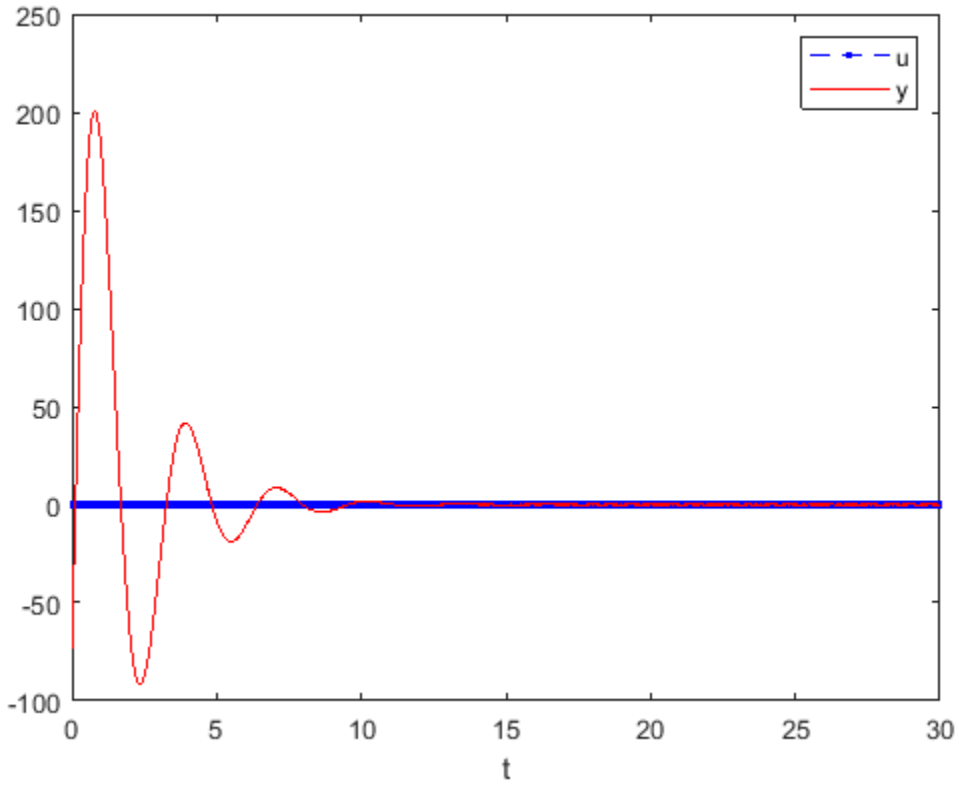
---- SINE SWEEP ---

```
if conductSineSweepExp==1

    g_hat_array = nan*ones(length(omega_array),1);
    theta_hat_array = nan*ones(length(omega_array),1);
```

```
    for omega_index = 1:length(omega_array)

        omega = omega_array(omega_index);

        num_cycles2average = num_cycles2average_nom
+5*ceil(omega);%this increases
        % the value of N for higher frequencies.

        t_final = decay_time+(2*pi/omega)*num_cycles2average;
        time = [0:Ts:t_final]';
        u = A_u*sin(omega*time);

        x0 = X0factor*randn(n,1); %random initial state
        noise = randn(length(time),1)*std_dev_noise;
```

do experiment to colect data

```
        y_at_omega = lsim(Plant,u,time,x0)+noise;
```




```
        %cut the transient period out, so that the data is consistent
 with the
        %theory of sine in sine out
        inds2average = [ceil(decay_time/Ts):1:length(time)]';
        N = length(inds2average); %number of samples to average over
```

```
        cosine_vector = cos(omega*time);
        sine_vector = sin(omega*time);
        ZcN = y_at_omega(inds2average)'*cosine_vector(inds2average);
        ZsN = y_at_omega(inds2average)'*sine_vector(inds2average);
        g_hat_omega = 2/A_u/N*sqrt(ZcN^2+ZsN^2); %gain est
        theta_hat_omega = atan2(ZcN,ZsN); %phase est, in rad

        %save estimates
        g_hat_array(omega_index) = g_hat_omega;
        theta_hat_array(omega_index) = theta_hat_omega;

        disp(['done with freq = ',num2str(omega),' rad/sec']);
```

```
done with freq = 100 rad/sec

done with freq = 10 rad/sec

done with freq = 0.08 rad/sec

done with freq = 0.7 rad/sec

done with freq = 0.4 rad/sec

done with freq = 0.9 rad/sec

done with freq = 0.6 rad/sec

done with freq = 0.02 rad/sec

done with freq = 15 rad/sec

done with freq = 40 rad/sec

done with freq = 65 rad/sec

done with freq = 89 rad/sec

done with freq = 1 rad/sec

done with freq = 450 rad/sec

done with freq = 3 rad/sec

done with freq = 260 rad/sec

done with freq = 3.2 rad/sec
```

```
    end
```

compute the true magnitude and phase at a large number of frequencies:

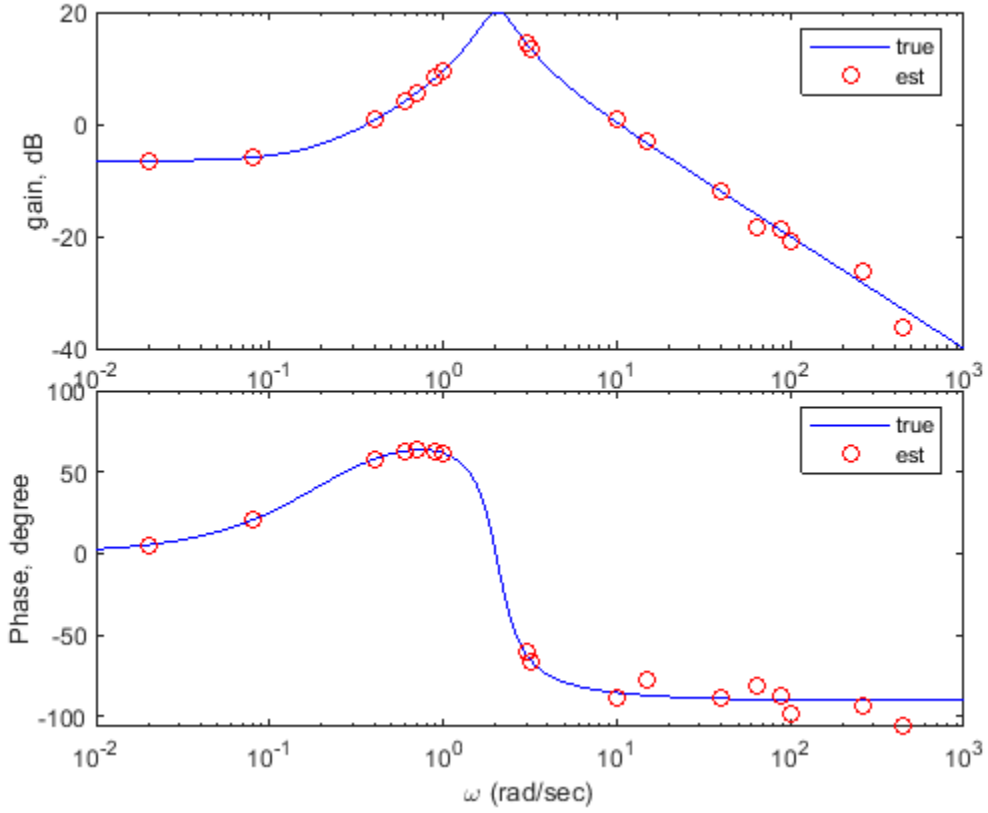
```
    w = logspace(-2,3,1000);
    [Gjw] = freqresp(Plant,w);
    Gjw = Gjw(:);
```

superimpose the estimate and the true frequency response on the same Bode plot

```
bodefig = figure
ax1 = axes('position',[0.1300 0.55 0.7750 0.4])
semilogx(w,20*log10(abs(Gjw)),'b-');
hold on;
semilogx(omega_array,20*log10(g_hat_array),'ro');
ylabel('gain, dB');
legend('true','est');

ax2 = axes('position',[0.1300 0.1 0.7750 0.4])
semilogx(w,angle(Gjw)*180/pi,'b-');
hold on;
semilogx(omega_array,theta_hat_array*180/pi,'ro');
xlabel('\omega (rad/sec)');
ylabel('Phase, degree');
legend('true','est');
```

```
bodefig =

  Figure (2) with properties:

      Number: 2
        Name: ''
       Color: [0.9400 0.9400 0.9400]
    Position: [440 298 560 420]
       Units: 'pixels'

  Use GET to show all properties


ax1 =

  Axes with properties:

              XLim: [0 1]
              YLim: [0 1]
            XScale: 'linear'
            YScale: 'linear'
    GridLineStyle: '-'
         Position: [0.1300 0.5500 0.7750 0.4000]
            Units: 'normalized'

  Use GET to show all properties


ax2 =

  Axes with properties:
```

```
            XLim: [0 1]
            YLim: [0 1]
          XScale: 'linear'
          YScale: 'linear'
   GridLineStyle: '-'
        Position: [0.1300 0.1000 0.7750 0.4000]
           Units: 'normalized'

  Use GET to show all properties
```




```
end
```

Published with MATLAB® R2020a