Lab Notebook: System Identification

Jack Center
CU Boulder ECEN 5018: Controls Lab
February 4, 2020

System Modeling

Lab one focused on system identification for a single motor:

$$V = L\dot{I} + RI + K_m\dot{\theta}$$

$$J\ddot{\theta} + b\dot{\theta} = K_T I$$

J = moment of inertia of the rotor
K_m = electromotive force constant
K_T = motor torque constant
b = motor viscous friction constant
R = resistance
L = inductance

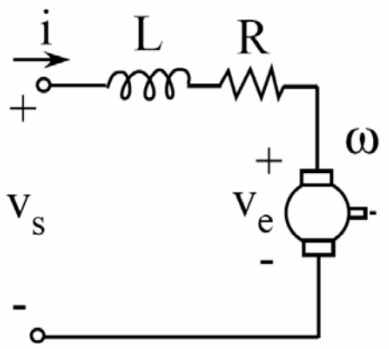

Figure 1: DC Motor Model

Transfer Function

For this analysis, dI/dt and b are considered negligible. Adjusting for this and applying the Laplace transform yields:

$$Js^2\theta(s) = K_T I(s)$$

$$V(s) = RI(s) + K_m\, s\, \theta(s)$$

$$\frac{\theta(s)}{V(s)} = \frac{1}{\frac{R\,J\,s^2}{K_T} + s\,K_m}$$

$$\theta(s) = \frac{1/K_m}{s\left(\frac{R\,J}{K_T\,K_m}s + 1\right)}\,V(s)$$

Where $\frac{R\,J}{K_T\,K_m} = \frac{1}{\omega}$, and $1/K_m = K$

$$T(s) = \frac{k}{\frac{s}{\omega} + 1} = \frac{\dot{\theta}(s)}{V(s)}$$

Under this model of the system, the unknown values are k and ω which represent a number of model parameters. It may be possible to identify these parameters individually, however, doing so is not necessary to describe transfer function of the system.

State Space

For this analysis, b will still be considered negligible, so b can be replaced by zero in the A matrix. It is included here for completeness.

$$x = \begin{bmatrix}\theta\\ \dot{\theta}\\ I\end{bmatrix}; \qquad \dot{x} = \begin{bmatrix} x_2 \\ \frac{K_T}{J}x_3 - \frac{b}{J}x_2 \\ -\frac{R}{L}x_3 - \frac{K}{L}x_2 + \frac{V}{L}\end{bmatrix}$$

$$A = \begin{bmatrix}0 & 1 & 0\\ 0 & -\frac{b}{J} & \frac{K_T}{J}\\ 0 & -\frac{K}{L} & -\frac{R}{L}\end{bmatrix}; \qquad B = \begin{bmatrix}0\\0\\\frac{1}{L}\end{bmatrix}$$

$$C = [0 \quad 1 \quad 0]$$

The input to the system, u(t), is the voltage, V, and the output is also given as a voltage. The ratio from the output voltage to radians per seconds is 1:1 which is determined in the tools section. So, for this model, the units on the output will be the angular speed of the motor in radians per second.

Tools

Experiments were run using Simulink and the data was evaluated using MATLAB. Simulations were run on a Quanser Rotary Servo Base which is a DC motor that takes torque (volts) as the input, and outputs angular position (radians) and angular velocity (volts). This servo base comes a Simulink model that allows it to be run directly from Simulink. The sampling rate for the base is 2ms and inputs must be less than 10 volts to protect the mechanical system. SHOULD I INCLUDE THE BLOCK DIAGRAM?

A quick analysis of the system was completed to determine the ratio between angular velocity in volts and radians per second by reviewing the system's steady state response to a step input. The angular rate in radians per second was determined from the change angular position over a specific time period. This was then compared to the angular velocity output in volts:

$$\frac{rads}{s} : volts = \frac{\theta(t_0) - \theta(t_f)}{t_0 - t_f} : \dot{\theta}_{volts} = 1.0061$$

This experiment revealed the ratio between the angular velocity in volts and radians per second is approximately equal to one. This fact will be used to interchange units on angular velocity between volts and radians per seconds throughout the remainder of the report.

Theory: System Identification

Step Response

This model is a first order system with respect to the angular velocity. As such, the parameters of the transfer function can be identified directly from the time domain response to a step input from the general first order transfer function.

$$y(t) = k\left(1 - e^{-t/\tau}\right) u(t)$$

where, $\tau = 1/\omega$

As time approaches infinity, y(t) approaches k. Therefore, k is the value of the steady state response to a unit input. τ is described by the time it would take the system to reach its steady state value, k, if it continued to increase at its initial rate. This value can be calculated using the initial slope and determining at what time it intersects k. Another method is to determine the time when the output has reached 63% of the steady state value, which will be also be equivalent to 0.63 * k (Figure 2).

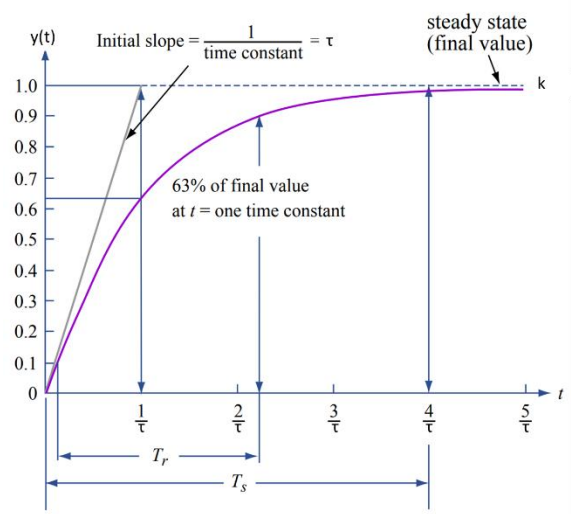

Figure 2: step response of first order systems (MIT OpenCourseWare, 2007).

The step response approach to system identification is straight forward, however, beyond a first order system it does not work. A more robust solution is to use a multi-sine for system identification.

Multi-sine

A multi-sine is the sum of excitation signals used to for system identification. Instead of running a test on each

sinusoidal input individually and compiling the results, a multi-sine allows a large number of sinusoids to be evaluated simultaneously. Restrictions on the frequencies to test are that it must be a multiple of 2π/T (where T is the test duration) and must be less than the Nyquist frequency, π/dt. This means the frequencies tested is the list 2π/T: 2π/T: π/dt. To find the input, each frequency is evaluated and the results are summed into a single vector:

$$U = \sum_{i=1}^{m} a * \sin(\omega\, i\, t\,)$$

$$where\ \omega = \frac{2\pi}{T}\ and\ m \leq \frac{T}{2dt}$$

Time is a vector from 0 to T incremented by the sampling time of the encoder. The vector can then be converted into the frequency domain through a Fourier transform. This input, U, will then be simulated to determine the output, Y, first in the time domain, then transformed into the frequency domain to obtain the transfer function:

$$T(s) = \frac{Y(s)}{U(s)}$$

To identify the system, the Bode plot of the transfer function is created by plotting T(s) in decibels (where dB = 20log(x) against the frequency list used on a logarithmic scale. The Bode plot is used to determine the values for k and ω by taking the initial value for k, and obtaining ω from the point where the magnitude has dropped by 3 dB (Figure 3).

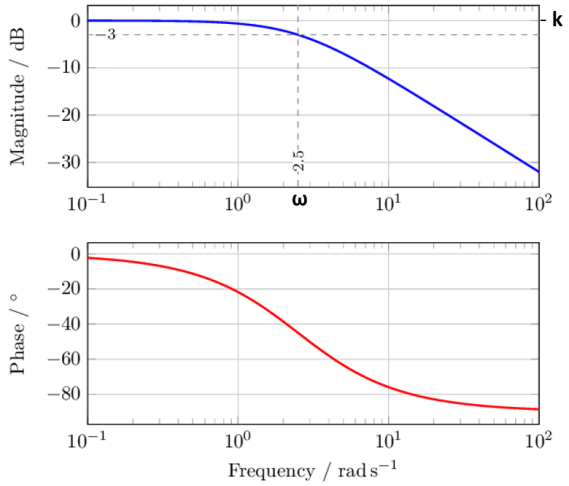

Figure 3: Bode diagram of a first order system (Octave, 2019).

Second Order System

Finally, the second order system, for the position of θ, was evaluated to further validate the parameters identified for the system. To do this, data for the angular position, θ, from a closed loop simulation is obtained and compared to identified model using the closed loop transfer function:

$$T(s) = \frac{\theta(s)}{V(s)} = \frac{k}{s\,(\frac{s}{\omega} + 1)}$$

$$H(s) = \frac{T(s)}{1 + T(s)}$$

The experiment output of θ can then be compared to the identified model of the system.

Results

Parameter Identification

Step Response

k = 1.3831

τ = 0.0280

Multi-sine

k = 1.4623

τ = 0.0224 (calculated)

τ = 0.0205 (adjusted)

The methods discussed in the theory section were implemented using the rotary base. First, the step response was evaluated by running a simulation for two seconds, with a unit step input at one second. This amount of time was sufficient for the output to reach steady state so the parameters could be identified (Figure 4). The value for k was determined by taking the average output over the last 0.6 seconds, yielding k = 1.0060. The value for τ was determined by finding the time associated with the data point immediately preceding the one that exceeded 63.2% of the stead state value, or k. In other words, this was the last data point less than 63.2% of k.

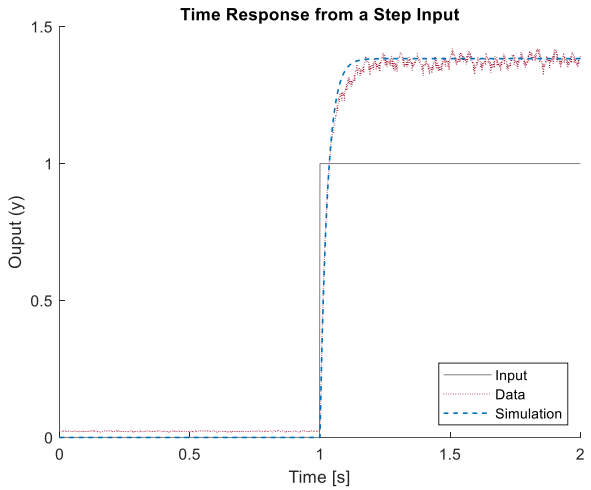

Figure 4: System response to a step input. Data plotted was used to obtain the parameters for step response identification.

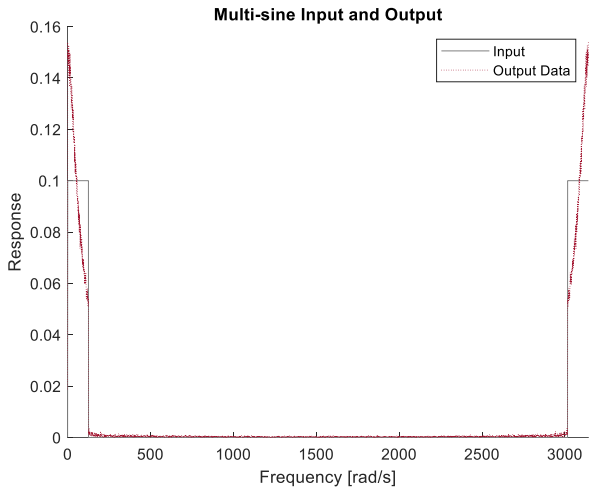

Figure 5: Input and output plotted in the frequency domain for the multi-sine used for system identification.

The multi-sine was constructed from a small sample of possible frequencies according to multi-sine theory section. The input and output for the multi-sine in the frequency domain were plotted to show they match the intended multi-sine input (Figure 5). Output values that were less than 9e-2 were considered zero and adjusted in the data set. The experimental transfer function was then constructed using the output and input data and plotted in Bode form for system identification. The value for k was determined by taking the average of the initial steady data points and yielded k = 1.4623. The value for τ was determined by finding the time associated with the first data point that was 3db less than k. This yielded τ = 0.0224.

After superimposing a Bode plot generated with the identified parameters, there was a clear phase misalignment between the data and the simulation. This was corrected for by adjusting the value of τ to 0.0205 which resulted in a much closer alignment to the data, as seen in Figure 6.

To provide further validation of the parameters identified, another simulation was run, this time for the closed loop second order system for angular position. Bode for this system was plotted against the data for the angular position to check how well the model matched reality (Figure 7).

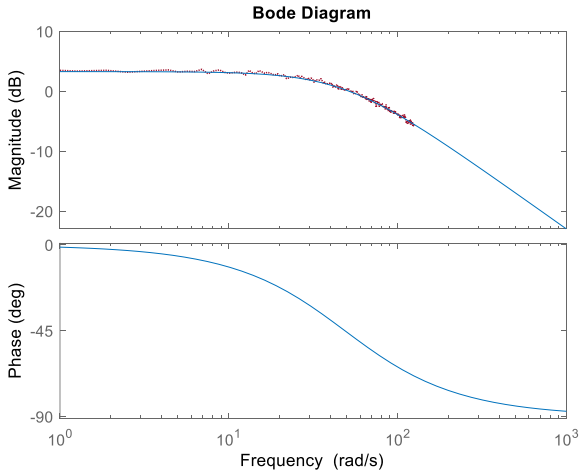

Figure 6: Bode diagram of first order system frequency response with the identified transfer function graphed as a blue line and the experimental data plotted as red dots.

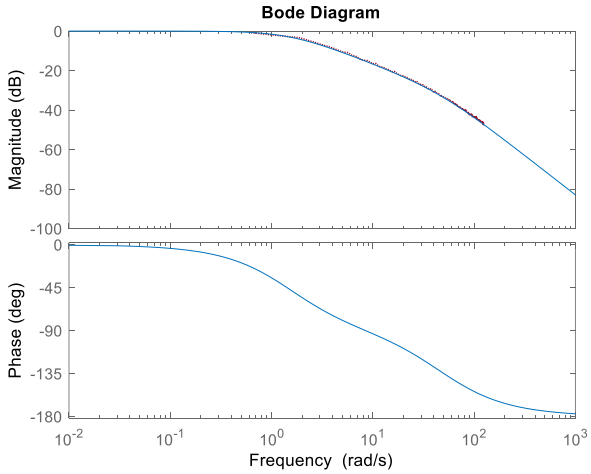

Figure 7: Bode diagram of second order system frequency response with the identified transfer function graphed as a blue line and the experimental data plotted as red dots.

Discussion

Reviewing the results from the two methods of system identification showed slight differences in the system identified. Plotting the parameters identified using the multi-sine approach agains the step input data shows that these parameters do not match that data particularly well either, however, it is also reasonably close (Figure 8). Furthermore, plotting a Bode diagram for the parameters identified using the step response method shows that they do not match the data from the multi-sine experiment particularlly well, although they are reasonably close (Figure 9).

Because both systems identified appear to match the system reasonably well, either model could be used to implement a feedback controller with good results. The best parameters to use will depend on the specific problem and the types of input expected, whether they be sinusoids or step functions. It is worth noting that although the step input method provided a better model for the system under a step input, this method can only be used for a first order system and, therefore, is not usable for any non-trivial system identification.

The parameters identified using the multi-sine were plotted against the second order system which closely matched the experimental data. This further validates the system identified using the multi-sine method and gives improved confidence in using these parameters to create control laws moving forward.

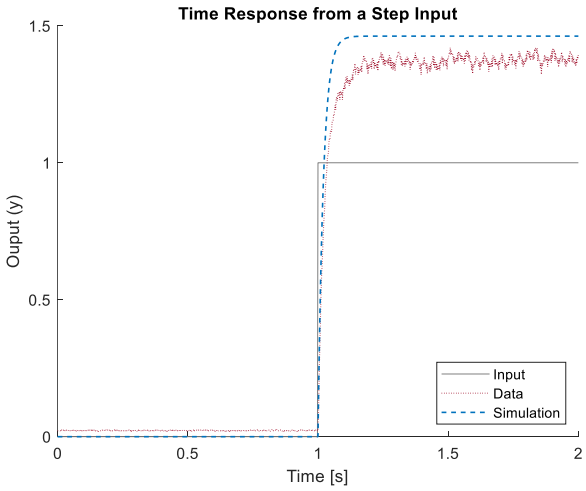

Figure 8: Parameters identified by step response plotted against the experimental multi-sine data.

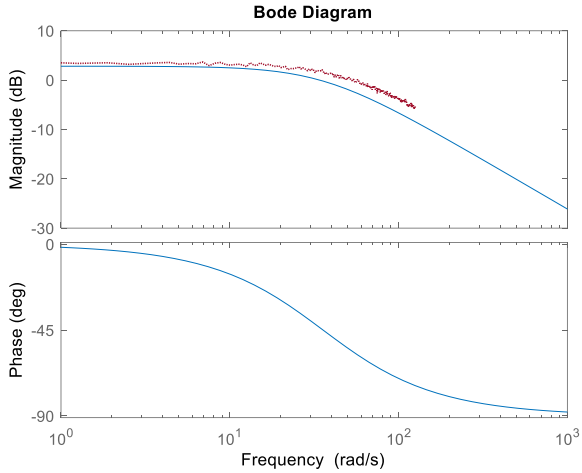

Figure 9: Parameters identified by multi-sine plotted against the experimental step response data.

References

MIT OpenCourseWare. (2007, September 19). *Review: step response of 1st order systems.* Retrieved from MIT OpenCourseWare: https://ocw.mit.edu/courses/mechanical-engineering/2-004-systems-modeling-and-control-ii-fall-2007/lecture-notes/lecture07.pdf

Octave. (2019, June 14). *Control Package.* Retrieved from Octave Wiki: https://wiki.octave.org/Control_package

Appendix A: Code

Quarc_Initial.m

```
clear; clc; close all;

% Initializing QuarcBlock

%Input Signal
    a = 1/5;                        %Amplitude
    T = 10;                         %Experiment Time
    dT = 0.002;                     %Sampling Time
    f = 2*pi/T;                     %Frequency
    F = 0:2*pi/T:2*pi/dT-2*pi/T;        %Frequency Vector
    Max = 200 * T/(2*pi);

    save('Motor_initialization_data', 'a', 'T', 'dT', 'F')
```

Quarc_PP.m (step input)

```
clear; clc; close all;

blue = [0, 0.4470, 0.7410];
orange = [0.8500, 0.3250, 0.0980];
yellow = [0.9290, 0.6940, 0.1250];
green = [0.4660, 0.6740, 0.1880];
grey = [0.5, 0.5, 0.5];
purple = [0.4940, 0.1840, 0.5560];
light_blue = [0.3010, 0.7450, 0.9330];
burgandy = [0.6350, 0.0780, 0.1840];

%% Quarc Post-Processing Data: Step Response
load('Motor_initialization_data')
load('step_response_data.mat')


    u = Input;                                          %Input Signal
    pos = Theta;                                        %Position [rad]
    vel = dTheta;                                       %Velocity [V]
    T = 10;
    t = 0:dT:T-dT;                                      %Time Vector
    f =0:2*pi/T:2*pi/dT-2*pi/T;                             %Frequency Vector [rad/s]

    % Identifying the System
    n = 300;
    k = (Theta(end) - Theta(end-n)) / ( Time(end) - Time(end-n) );          %"Slope" [rad/s]

    volt_avg = mean(vel(end-n:end));                            %Transient Voltage
    volt_avg_60 = volt_avg*0.632;                               %60% of Transient Voltage
    index_tau = length(vel(1:find(vel > volt_avg_60)));
    tau = Time(index_tau) - 1;

    K = k/volt_avg;         %[(rad/s)/V]

    %% Creating Transfer Function
    s = tf('s');
    % TF = K / (tau*s +1);              %Tranfer Function with vel thing
    TF = 1.4623 / (0.0205*s + 1);  % from multi-sine
    % TF = k / (tau*s + 1);               % from step id

    output_TF = lsim(TF, u, Time);

    %% Time Response of System
    figure
    title("Time Response from a Step Input")
    hold on
    plot(Time, u, "color", grey, "LineStyle", "-")
    plot(Time, vel, ':', "color", burgandy, "LineWidth", .7)
    alpha(.5)
    plot(Time, output_TF, "color", blue, "LineStyle", "--", "LineWidth", 1)
    hold off
    xlabel('Time [s]')
    ylabel('Ouput (y)')
    legend("Input", "Data", "Simulation", "Location", "southeast")
```

Quarc_MS.m (multi-sine)

```
clc; close all;

blue = [0, 0.4470, 0.7410];
orange = [0.8500, 0.3250, 0.0980];
yellow = [0.9290, 0.6940, 0.1250];
green = [0.4660, 0.6740, 0.1880];
grey = [0.5, 0.5, 0.5];
purple = [0.4940, 0.1840, 0.5560];
light_blue = [0.3010, 0.7450, 0.9330];
burgandy = [0.6350, 0.0780, 0.1840];

load('Motor_initialization_data', 'a', 'T', 'dT', 'F')
load('multisin_response_data.mat')

%% Quarc Post Processing Data: Muti-Sine

u_MS = Input_MS;        %Input Response
vel_MS = dTheta_MS*K;   %Output Response [rad/s]

%% Time Response of System
   figure()
   subplot(2,1,1)
   plot(Time_MS, u_MS)
   xlabel('Time [s]')
   ylabel('Step Input')
   subplot(2,1,2)
   plot(Time_MS, vel_MS)
   xlabel('Time [s]')
   ylabel('Ouput Response')

%% FFT of Input and Ouput

   U_MS = fft(u_MS);
   VEL_MS = fft(vel_MS);


   figure()
   hold on
   plot(F,abs(U_MS/length(U_MS)), "color", grey, "LineStyle", "-")
   plot(F,abs(VEL_MS/length(U_MS)), "color", burgandy, "LineStyle", ":")
   hold off
   title("Multi-sine Input and Output")
   legend("Input", "Output Data")
   xlabel('Frequency [rad/s]')
   ylabel('Response')
   xlim([0, 3141])

   index0 = find(abs(U_MS/length(U_MS)) >= 9e-2);
   B = 0*U_MS;
   B(index0) = VEL_MS(index0)./U_MS(index0);        %System Response
   B(index0) = abs(B(index0));                %Magnitude of System Response
   B_log = 20*log10(B);

   K_MS = mean(B_log(2:25));
   K_3dB = K_MS - 3;
   Index_tau = find(B_log(2:end) < K_3dB);
   Index_tau = Index_tau(1)+1;
   omega_MS = F(Index_tau);
   tau_MS = 1/omega_MS;

   %% Creating Transfer Function
   s = tf('s');
%     K_MS = db2mag(K_MS);
%     H = K_MS / (.0205*s +1);

   K_MS = 1.3831;
   H = K_MS / (.0280*s +1);

   figure
   hold on
   semilogx(F,B_log,"color", burgandy, "LineStyle", ":", "LineWidth", 1)
   bode(H)
   hold off
```

Quarc_ClosedLoop.m (second order system)

```
clc; close all;

load('Motor_initialization_data')
load('Closed_Loop_data.mat')

%% Quarc Post Processing Data: Closed Loop

r = Reference;        %Input Response
pos = Theta;   %Output Response [rad/s]


%% Time Response of System
   figure()
   subplot(2,1,1)
   plot(Time, r)
   xlabel('Time [s]')
   ylabel('Step Input')
   subplot(2,1,2)
   plot(Time, pos)
   xlabel('Time [s]')
   ylabel('Ouput Response')

%% FFT of Input and Ouput

   R = fft(r);
   POS = fft(pos);

%     figure()
%     subplot(2,1,1)
%     plot(F,abs(U_MS/length(U_MS)))
%     xlabel('Frequency')
%     ylabel('FFT Input Response')
%     subplot(2,1,2)
%     plot(F,abs(VEL_MS/length(U_MS)))
%     xlabel('Frequency')
%     ylabel('FFT Ouput Response')

   index0 = find(abs(R/length(R)) >= 9e-2);
   B = 0*R;
   B(index0) = POS(index0)./R(index0);        %System Response
   B(index0) = abs(B(index0));                %Magnitude of System Response
   B_log = 20*log10(B);

   %% Creating Transfer Function
   s = tf('s');
   H = db2mag(K_MS) / (s*(.0205*s +1));
   Hcl = H/(1+H);

   figure
   hold on
   semilogx(F,B_log,"color", burgandy, "LineStyle", ":", "LineWidth", 1)
   bode(Hcl)
   hold off
```