Dynamic Modeling and Control

Project 1: Time Domain Identification of a Spring-Cart System

Introduction

The goal of this experiment is to expose students to system identification methods for a mechanical system. A mass-spring cart system can be modeled as having two system poles for each cart. As a result, a single cart is a second order system, and two linked carts is a fourth order system. Several simple methods exist to find the transfer function relating the input position signal to the cart position output, but these methods fail for more complicated systems. As a result, an algorithmic MATLAB function based solution will be applied, which is more accurate and can model unknown system parameters.

Theory

Input Creation in MATLAB

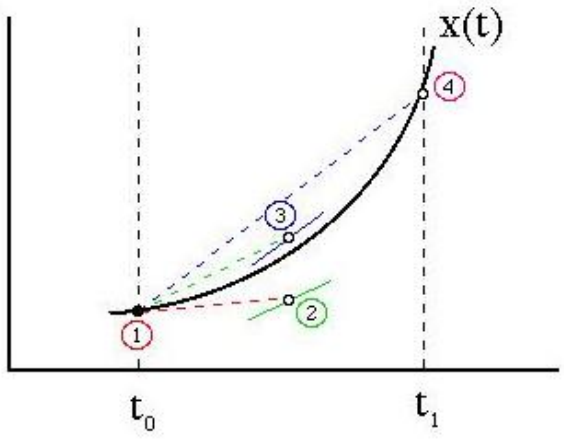
Before using any of these identification methods in MATLAB, a variable vector needs to be defined to represent the actual system input. The input is a step of position, where the cart is held at a certain place until a step time (call it 0 for ease), and then released. The input step can be defined in MATLAB using the following:

```
% define the position input
amp1 = 0.15;
input_pos1 = zeros(1,length(t));
ind = find(t>0.45); % finding time >0.45 to replace by amp1
%replace 0.45 by the actual step time you have!!!!
input_pos1(ind)=amp1;
input_pos1 = input_pos1'; % transpose input to match data from exp
```

Simple Identification Methods

Runge-Kutta Identification Method

The Runge Kutta simulation is an iterative method of approximating solutions to ordinary differential equations. This process takes the weighted average of a set of steps, and advances the particle according to the weighted average at the point being analyzed. The plot below gives a visual representation of the method using 4 point to take the weighted average, and the defining equations are defined to the right of the plot.



$$
\begin{aligned}
dx1 &= \Delta t\, v_{x,n} \\
dv_x1 &= \Delta t\, a_x(x_n, y_n, t) \\
dx2 &= \Delta t\, (v_{x,n} + \frac{dv_x1}{2}) \\
dv_x2 &= \Delta t\, a_x(x_n + \frac{dx1}{2}, y_n + \frac{dy1}{2}, t + \frac{\Delta t}{2}) \\
dx3 &= \Delta t\, (v_{x,n} + \frac{dv_x2}{2}) \\
dv_x3 &= \Delta t\, a_x(x_n + \frac{dx2}{2}, y_n + \frac{dy2}{2}, t + \frac{\Delta t}{2}) \\
dx4 &= \Delta t\, (v_{x,n} + dv_x3) \\
dv_x4 &= \Delta t\, a_x(x_n + dx3, y_n + dy3, t + \Delta t) \\
x_{n+1} &= y_n + \frac{dx1}{6} + \frac{dx2}{3} + \frac{dx3}{3} + \frac{dx4}{6} \\
v_{x,n+1} &= v_{x,n} + \frac{dv_x1}{6} + \frac{dv_x2}{3} + \frac{dv_x3}{3} + \frac{dv_x4}{4}
\end{aligned}
$$

This method can accurately approximate a solution to an exponentially decaying sine wave, as it is only a second order system that is being analyzed. The definitive second order equation that is being analyzed is shown below in Eq. 1. For this particular approximation, the initial force is being modeled as a step force. (*Galaxy*)

$$y'' = -2\xi\omega_n y' - {\omega_n}^2 y + F_0/M \qquad \text{(Eq. 1)}$$

Where:

$\xi = {C}/{2\sqrt{KM}}$ $\omega_n = \sqrt{{K}/{M}}$

K = spring constant M = Mass

C = damping constant F0 = Initial step force

This approximation can be readily used in MATLAB. It uses the ode45 solver function to perform the RK4 simulation. The ode45 solver takes an input function, a time vector initial conditions. The use of the ode45 solver in MATLAB is shown below.

```
options = odeset('RelTol',1e-4,'AbsTol',[1e-4 1e-4],'MaxStep',.001);
   [TL,YL] = ode45(@stepresponsedot,[0 tfinal], [0 0],options);

    function dy = stepresponsedot(t,x)
    global omegan zeta f0 t0 M
    dy = zeros(2,1);
    dy(1) = x(2);
    dy(2) = -2*zeta*omegan*x(2) - omegan^2*x(1) + ...
        f0/M*Step(t,t0);
    end
```

This section of a MATLAB script will give an approximation of a given step function. The step function simulates the force acted upon the studied system, returning a

solution similar to the theoretical definition of the response. However, this method is simply studying a second order system, and does not account for any irregularities or changing of natural frequencies as seen in the actual response of the system. Therefore, this approximation can only model a perfect second order system.

Euler Identification Method

The Euler method is a simple iterative numerical analysis that performs integrates ordinary second order differential equations, and is essentially a simple version of the Runge Kutta method. The basic form of the equation is shown below in Eq. 2. The equation that is being analyzed is shown in Eq. 3 and 4.

$$y_{n+1} = y_n + hf(t_n, y_n) \quad \text{(Eq. 2)}$$

$$\omega_{n+1} = \omega_n - \frac{M_f}{M}\omega_n h - \frac{K}{M}\theta_n + \omega_n h \quad \text{(Eq. 3)}$$

$$\theta_{n+1} = \theta_n + \omega_n h \quad \text{(Eq. 4)}$$

Where

- h = step size
- ω = frequency
- M_f = dry friction coefficient
- C = damping coefficient
- K = spring constant
- M = mass
- θ = position

The step size is what determines the accuracy of the approximation. The higher the step size, the more accurate the approximation will be. In terms of MATLAB code, the program was created to run a for-loop through the analyzed equation for the number of steps. The section of the script that performs the iteration is shown below. This specific iteration uses a very similar transform compared to the equation seen in equation 1, however this equation uses an Mf value to help compensate for dry friction (*Inman 2008*).

```
deltat = tfinal/Nsteps;
theta(i)=theta0;
omega(i)=omega0;
for i=1:(Nsteps)
omega(i+1)=omega(i)-Mf/J*sign(omega(i))*deltat...
      -K/J*theta(i)*deltat-C/J*omega(i)*deltat;
theta(i+1)=theta(i)+omega(i)*deltat;
end
```

MATLAB Algorithmic Identification Method

Prediction-Error Method

The task of finding a transfer function that maps the input step $R(t)$ to the output $X(t)$ is posed as a system identification problem. In this particular application, the system identification is performed by batch processing using N points. In this scheme, the step input $R(t)$ is applied to the unknown system composed of the mass-spring system with the position encoder sensor, and the resulting output $X(t)$ is recorded. The system identification algorithm is provided both the input $R(t)$ and the output $X(t)$ to identify a transfer function that predicts $\hat{X}(t)$. The output of the predicted transfer function $\hat{X}(t)$ is then subtracted from the output of the position

sensor $X(t)$to generate an error signal. This error signal is used as a part of an identification algorithm to update the prediction function to generate a better prediction of the unknown system for the batch of N data points. This update process is repeated until the error either reaches some convergence criterion or the maximum number of iterations is reached. The performance metric used in this project is the mean square error cost function, which is defined as

$$J_{MSE}(\Theta) = E[e^2(n;\Theta)] \approx \frac{1}{N}\sum_{n=1}^{N} e^2(n;\Theta) \quad \text{(Eq. 6)}$$

where Θ is a parameter vector describing the identified filter and the error signal is $e(n;\Theta) = X(n) - \hat{X}(n,\Theta)$ (*Panda 2011*).

The goal of any optimization algorithm is to minimize a given error metric. For simplicity, consider the replacement of the argument *t* by the argument *n* to describe the sampled-data system, where n represents a sample instance. The output $X(n)$ can be represented as

$$X(n) = \frac{B(q)}{A(q)} R(n) \quad \text{(Eq. 7)}$$

where $A(q)$ and $B(q)$ are polynomials, whose ratio $B(q)/A(q)$ is the true transfer function of the unknown system. The forward shift operator is represented by q, defined such that $q^{-i}X(n) = X(n-i)$.

The output of the prediction filter, denoted as $\hat{X}(t)$, is

$$\hat{X}(n) = \frac{\hat{B}(q)}{\hat{A}(q)} R(n) \quad \text{(Eq. 8)}$$

In order to identify the prediction filter $\hat{X}(n)$, the polynomials $\hat{B}(n)$, $\hat{A}(n)$and their degrees (or orders) n_B and n_A are optimized as a part of a parameter vector Θ. The parameter vector Θ is used as the argument in the MSE cost function defined in Equation 6, and the parameters are optimized to produce a minimal MSE.

The identification method used in this project is the prediction-error method (PEM). PEM uses numerical optimization to minimize the prediction error. The prediction error $\bar{e}(n)$is the error between the predicted output of the system and the measured output. First, consider the SISO model described as

$$X(n) = \frac{B(q)}{A(q)} R(n) + H(q)\chi(n) \quad \text{(Eq. 9)}$$

where $\chi(n)$ is noise present in the system output, and $H(q)$ models noise dynamics. The prediction error between the measured output and predicted output can be determined to be

$$\bar{e}(n) = \hat{H}^{-1}(q)\left[X(n) - \frac{\hat{B}(q)}{\hat{A}(q)} R(n)\right] \quad \text{(Eq. 10)}$$

The goal of PEM is to minimize the MSE cost function using the error defined in Equation 10. Thus, the parameter vector for the MSE cost function is $\Theta = (\hat{A}, \hat{B}, n_A, n_B, \hat{H})$. When the cost function is minimized, the resulting model is the system estimate. PEM identification is a batch estimation method, where a minimization algorithm must be applied to determine the optimal model. This identification method is further described by (*Ljung*).

Before using the PEM algorithm, you must do some small data processing to set your input and output.

The PEM algorithm is implemented in MATLAB using the pem command. PEM chooses the optimal parameters of Θ, including $\hat{A}, \hat{B}, n_A, n_B$, and $\hat{H}$ (*McPheron 2014*). The steps to implement this method in MATLAB are shown (with comments in the code below. Start by naming the input position $R(n)$ input_pos1 and the output position $X(n)$ pos1. Define the time information for the system *t*.

```
% Get time period for system ID
Ts      = t(3) - t(2);

% build system model for system ID
data = iddata(pos1, input_pos1, Ts);
% Set time units to seconds
data.TimeUnit = 'sec';
% Set names of input channels
data.InputName = {'Input: Position Step'};
% Set units for input variables
data.InputUnit = {'m'};
% Set name of output channel
data.OutputName = {'Output: Cart 1 Position'};
% Set unit of output channel
data.OutputUnit = {'m'};

% estimate the model
est_model_pem = pem(data);

% Build a state space (time-domain model from the ID data
[Apem,Bpem,Cpem,Dpem,Kpem,ts] = idssdata(est_model_pem);
% Get the continuous time transfer function from the state space model
[num1d,den1d] = ss2tf(Apem,Bpem,Cpem,Dpem);
[num1c,den1c] = d2cm(num1d,den1d,Ts);
tf_cart1_pem = tf(num1c,den1c)
```

The resulting transfer function tf_cart1_pem is the identified system model.

You may then simulate the response of this identified transfer function to the input signal and compare this with the measured signal.

MATLAB Data Visualization

To visualize the recorded data and the data resulting from the identified system (particularly from PEM method), you'll need to simulate the system using:

```
% Simulate the identified system with
[out_pem,state_pem] = dlsim(Apem,Bpem,Cpem,Dpem,input_pos1);
```

Then you can plot the data as shown below, where pos1 is the measured cart position from the experiment.

```
figure(5)
plot(t,pos1,'r',t,out_pem,'k','Linewidth',2)
set(gca,'FontName',FontName,'FontSize',FontSize)
xlabel('Time [s]','FontName',FontName,'FontSize',FontSize)
ylabel('Position [m]','FontName',FontName,'FontSize',FontSize)
legend('Cart Position','Estimated Cart Position')
title('Training Data: 0.165 m
Step','FontName',FontName,'FontSize',FontSize)
```

Tasks

Part 1: Hand Derivation

Derive, by hand, the general expression for a **single** cart and a **two** cart system that defines the transfer function from a position input to the position output for each system. Keep in mind that friction constitutes damping. Write the transfer functions

Part 1: Data Collection

- Verify the setup of the two cart system. The setup is shown in:
  https://www.youtube.com/watch?v=zKWSyb4i_dw
- Collect data for the following cases:

1. Single cart (right, cart 1), with cart 2 pinned, ***0.15 m input step.*** At rest, zero the cart positions. Then, pin down cart 2, pull cart 1 until the position is -0.15 m, ZERO the positions, start experiment, and release.

2. Single cart (left, cart 2), with cart 2 pinned, ***0.15 m input step.*** At rest, zero the cart positions. Then, pin down cart 1, pull cart 2 until the position is -0.15 m, ZERO the positions, start experiment, and release.

3. Two carts, with input applied to right, cart 1, ***0.15 m input step.*** At rest, zero the cart positions. Then, hold down cart 2 (at position 0), pull cart 1 until the position is -0.15 m, ZERO the positions, start experiment, and release.

4.

Data collection with Logger Pro is demonstrated in:
https://www.youtube.com/watch?v=v08xirikUSA

- Convert data to MATLAB variables for processing. This is demonstrated in:
  https://www.youtube.com/watch?v=Vhd4RZgs6rA

Part 2: Simple Identification Methods for Second Order System
Choose one of the following simple methods for system identification, using the information from the theory section. Complete this for both single cart systems. (If you completed this project early for research, you must use a different method than you have already done)

1. Runge-Kutta Method
2. Euler Identification

Part 3: Time Domain Identification Using Prediction Error Method for Second Order System
Use the Prediction Error Method to identify the transfer function from position input to the position output for each system for each of the single cart systems.

Part 4: Time Domain Identification Using Prediction Error Method for Fourth Order System
Use the Prediction Error Method to identify the transfer function from position input to the position output for each system for the multi cart system.

Submissions
Your submission report must be well organized and labeled and include the following information:

1. Hand derivation
2. Code for simple identification for both cart 1 and cart 2 second order systems.
3. Code for the PEM identification for cart 1 and cart 2 second order systems.
4. Code for the PEM identification for cart 1 and cart 2 fourth order system.
5. Plot for simple identification for both cart 1 and cart 2 second order systems showing measured output and estimated output.
6. Plot for the PEM identification for cart 1 and cart 2 second order systems. showing measured output and estimated output.
7. Plot for the PEM identification for cart 1 and cart 2 fourth order system showing measured output and estimated output.
8. A two-three paragraph reflection on the results of the experiment and the success of this experiment.

Hints For Plotting
I require the following about your plots (due to personal pet peeves):

- Figure font formatting MUST match the report formatting
- Line width must be big enough to see. That means it must be at least width of 2.
- Ylabels and xlabels are required.
- Your figures cannot have both a title and a caption.

Here are some useful hints for plotting:

1. Set the FontName and FontSize as variable at the top of your file.
2. Use figure (1), figure(2) etc. to open new plot windows for each figure.
3. The next line after a plot command will set the FontName and FontSize so that way you don't have to use these repeatedly. This is done using:
   `set(gca,'FontName',FontName,'FontSize', FontSize)`
4. Use different line styles for multiple traces on the same plot for easier reading.

References

"Documentation." *Fast Fourier Transform*. MathWorks, n.d. Web. 03 Dec. 2015. <http://www.mathworks.com/help/matlab/ref/fft.html>.

"Galaxy Simulator Parameters Defined." *Galaxy Simulator Parameters*. University of Illinois, n.d. Web. 03 Dec. 2015. <http://rainman.astro.illinois.edu/ddr/ddr-galaxy/parameters.html>.

Inman, Daniel J. "Chapter 2: Basics of Vibration Dynamics." *Engineering Vibration*. Upper Saddle River: Pearson Education, 2008. N. pag. Print.

L. Ljung. System Identification Theory for the User. 2nd ed. Prentice Hall (1999).

B. McPheron. Flux Regulation in Powered Magnets: Enabling Magnetic Resonance Experiments with Pulsed Field Gradients. A Dissertation in Electrical Engineering. The Pennsylvania State University, The Graduate School (2014).

G. Panda, P.M. Pradhan, B. Majhi. IIR system identification using cat swarm optimization. Expert Systems with Applications, Volume 38 (2011) 12671-12683.

Rubric

The rubric by which this project will be graded is seen in this section.

Requirement	**Point Value**	**Points Awarded**
System Identification [40 pts]		
Hand derivation	5	
Code for simple Identification	10	
Code for PEM for Cart 1	10	
Code for PEM for Cart 2	5	
Code for PEM for 4^{th} order system	10	
Data Presentation [20 pts]		
Plot for original and identified signal given the same input for simple ID	5	
Plot for original and identified signal given the same input for PEM ID (Cart 1)	5	
Plot for original and identified signal given the same input for PEM ID (Cart 2)	5	
Plot for original and identified signal given the same input for PEM ID (4^{th} order)	5	
Report [40 pts]		
Clear organization and labeling	10	
Figures have axis labels	10	
Figures have relevant captions and descriptions	10	
Two paragraph reflection on results	10	
TOTAL	100	