

Hands-on Lab:

LabVIEW – Simulink Testing

Previously, one captured both the step and free fall response of the LEGO Damped Compound Pendulum (DCP). Using the logarithmic decrement equation and the DCP's physical measurements, one calculated the damping ratio, coefficient of friction, and moment of inertia (**Figure 1-1a**). Such system identification can now be used to determine the DCP (i.e. plant) transfer function. Equipped with this transfer function, one can use simulation tools, like Simulink, for future controller design.

Concept 1: Calculating the DCP transfer function

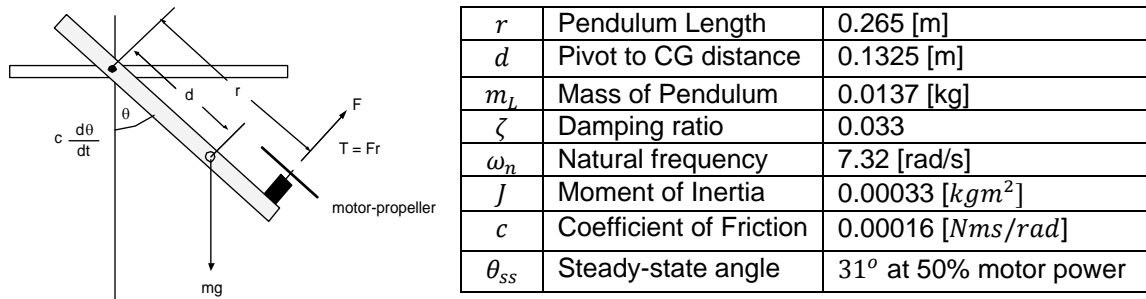


Figure 1-1a: DCP free body diagram (left) and both physically measured and calculated values (right)

A second-order system is characterized by the equation (1):

$$\ddot{\theta} + 2\zeta\omega_n\dot{\theta} + \omega_n^2\theta = 0 \quad (1)$$

From the free body diagram, the motor-prop generates a torque T . Thus dynamic equilibrium is given by (2):

$$\ddot{\theta} + \frac{c}{J}\dot{\theta} + \frac{m_Lgd}{J}\sin\theta = T \quad (2)$$

Linearizing (2) where $\sin\theta \approx \theta$ for small θ , and matching coefficients in equations (1) and (2) reveals the following relationships

$$c = 2\zeta\omega_nJ \quad (3a)$$

$$J = \frac{m_Lgd}{\omega_n^2} \quad (3b)$$

Taking the Laplace transform of (2) yields

$$s^2\theta(s) + s\frac{c}{J}\theta(s) + \frac{m_Lgd}{J}\theta(s) = T(s) \quad \text{or} \quad \left(s^2 + \frac{c}{J}s + \frac{m_Lgd}{J}\right)\theta(s) = T(s) \quad (4)$$

Hence

$$\frac{T(s)}{\theta(s)} = \frac{1/J}{Js^2 + cs + m_Lgd}$$

In block diagram form, (4) looks like **Figure 1-1b**. The torque $T(s)$ results from the thrust force applied to the DCP's lever arm r (i.e. the pendulum's length). This thrust is a result of a voltage applied to the motor-prop and prop size. The NXT Brick creates this voltage using a power command (in units of %) ranging from 0 to 100. Let $M(s)$ represent this motor power.

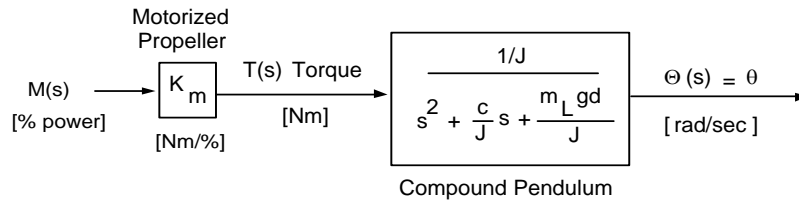


Figure 1-1b: Block diagram relating the motor power input $M(s)$, to the DCP's angle output $\theta(s)$

One could analytically use the prop's dimensions (e.g. diameter, number of blades, and pitch) to calculate lift. If can assume $M(s)$ and $T(s)$ are linearly related and use a proportionality constant (motor gain) K_m for the relationship as in (5)

$$K_m = \frac{T(s)}{M(s)} \quad (5)$$

To determine K_m one can recognize that at dynamic equilibrium, when the DCP is at its steady-state angle θ_{ss} , (2) becomes:

$$m_L g d \sin \theta_{ss} = T|_{ss} \quad (6)$$

In the previous lab, as shown in Figure 1-1a (last row of table), for a $M(s) = 50\%$ step input, yielded a $\theta_{ss} = 31^\circ$. Substituting this into (6) and (5) yields

$$K_m = \frac{m_L g d \sin 31^\circ}{50\%} = \frac{0.0137 \text{ kg} \cdot 9.81 \frac{\text{m}}{\text{s}^2} \cdot 0.1325 \text{ m} \cdot 0.515}{50\%} = 0.00917 \text{ Nm} \quad (7)$$

Using the values from (7) and the table in **Figure 1-1a**, the resulting block diagram is given in **Figure 1-1c**.

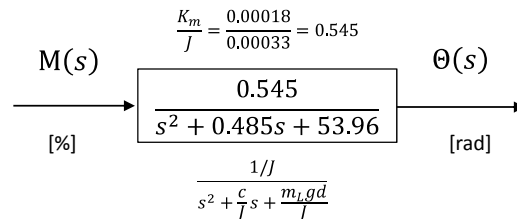


Figure 1-1c: Transfer function relating the motor input (as a percentage) and angle output (in radians)

Exercise 1:

- 1.1 Redo the table in Figure 1-1a, using the values you observed and calculated
- 1.2 Using 1.1, calculate the motor gain K_m
- 1.3 Sketch Figure 1-1c, using and answers to 1.1 and 1.2

Concept 2: Simulink Testing

Simulink is Matlab's graphical programming environment. XCOS is the equivalent, for Scilab, a free open-source version. Simulink and XCOS while not exactly the same, have very similar blocks. For this concept, Simulink¹ will be used.

Step 1: Launch Matlab and Open Simulink

In Matlab R2016, launching the program will reveal the opening screen (**Figure 2-1A left**) and clicking the Simulink icon brings up the Simulink Start Page (**Figure 2-1A right**).

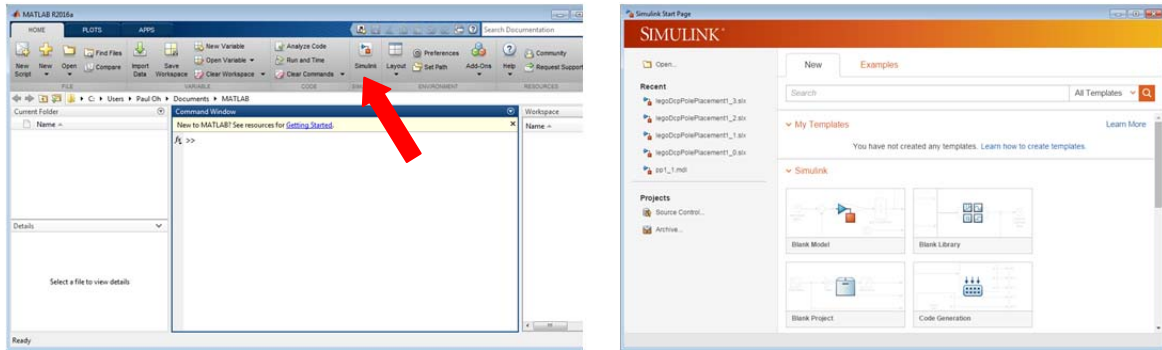


Figure 2-1A: Matlab launch allows one to click on the Simulink icon, indicated by the red arrow (left). This launches the Simulink Start Page (right)

Clicking on New Model launches a blank canvas screen (**Figure 2-1B left**). From the canvas' menu bar, click View – Library Browser. A menu of blocks is then revealed (**Figure 2-1B right**)

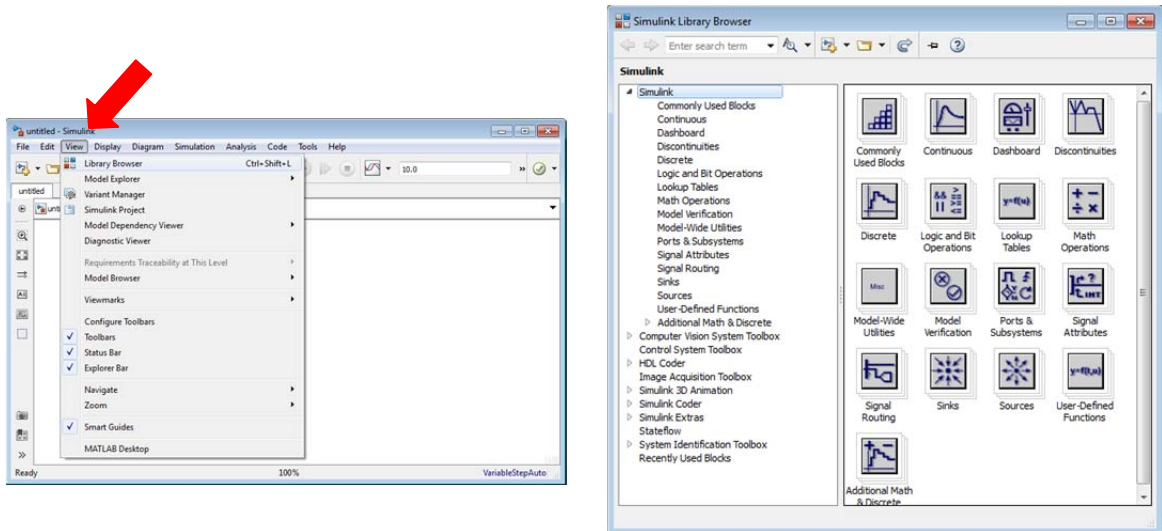


Figure 2-1B: Red arrow points to the menu bar's View selection (left). Clicking Library Browser reveals the blocks one can use in a Simulink simulation (right)

¹ Matlab R2016a version was used for these notes

Before populating the Simulink canvas, save the file as `simulinkDcpStepInput1_0.slx`.

Step 2: Populate the Simulink Canvas – Transfer Function

Using the Library Browser, click Continuous, select and drag the Transfer Fcn block into the canvas (**Figure 2-2A left**). Double-clicking any white space in the canvas, allows one to add text. Commenting your program with details is a good practice. Double-clicking the Transfer Fcn block pops-up the Block Parameter box. Referring to **Figure 1-1C** and your answers from **Exercise 1-3**, enter the Numerator and Denominator coefficients (**Figure 2-2A right**).

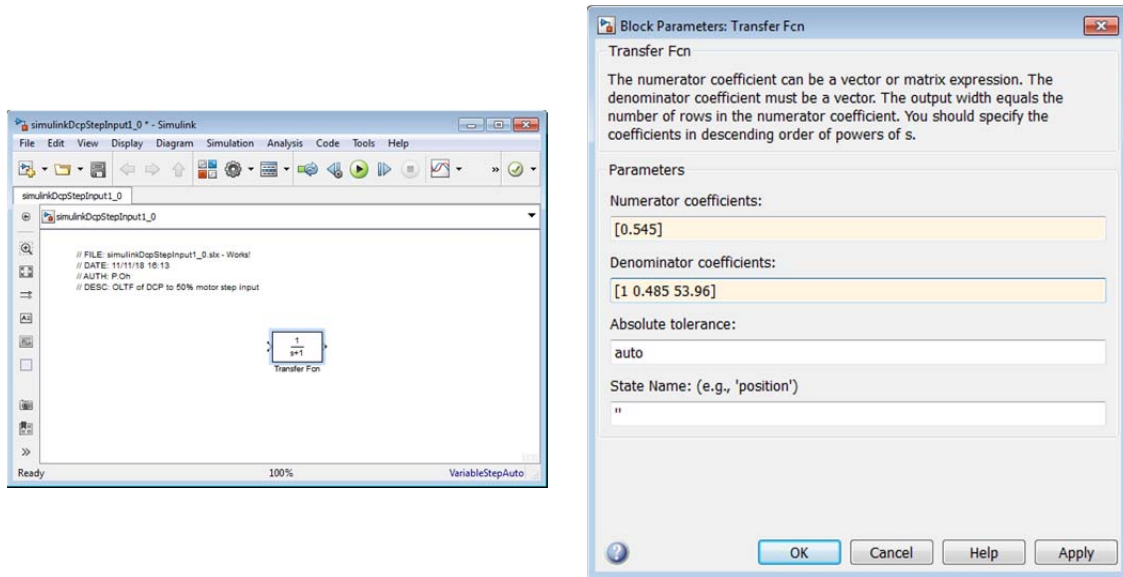


Figure 2-2: Transfer Fcn block and text comments in the Simulink canvas (left). Double-clicking the Transfer Fcn block reveals the Block Parameters pop up box (right). The numerator and denominator coefficients were entered using derived values shown in **Figure 1-1C**.

Make the Transfer Fcn block slightly bigger, by clicking and dragging a corner. One will then be able to see the transfer function values in the block.

Step 3: Populate the Simulink Canvas – Gains

Next, under Commonly Used Blocks, click and drag 2 gains. Double-click each gain to specify their values. Also connect the gain and Transfer Fcn blocks by pointing the mouse to an output and dragging the resulting wire into an input (**Figure 2-3A**).

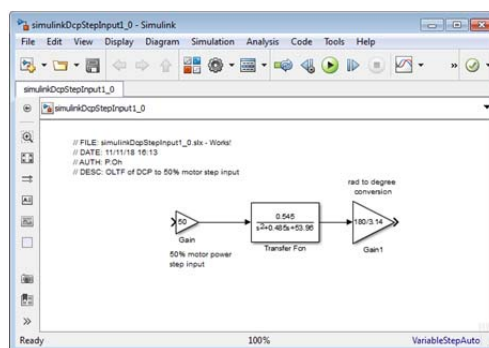


Figure 2-3A: Gain blocks before and after the Transfer Fcn block, with input and outputs wired

Step 4: Add Step Input and Output Scope

In the Library Browser, click Sources and click-and-drag the Step block into the Simulink canvas and wire it to the gain. Double-click the Step block and set Step time to 0. Click Sinks and similarly bring a Scope into the Simulink canvas and wire it to the gain. Save the resulting program (**Figure 2-4A**) as `simulinkDcpStepInput1_0.slx`.

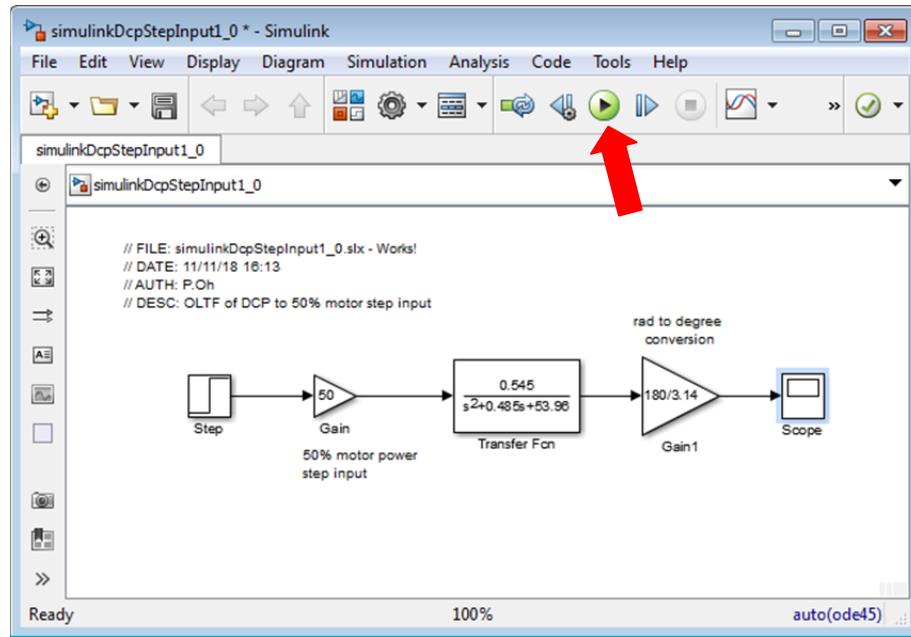


Figure 2-4A: Finished Simulink program. Setting the Step time to 0, sends a step input value of 1, which is multiplied by the gain 50. This represents a 50% motor power input into the DCP transfer function. Since the output of this transfer function is in radians, a rad-to-degree gain is used. The result is then displayed on a scope.

Step 5: Execute Simulink program

Hitting the play button (red arrow in Figure 2-4A) will execute the program and when completed, a chime will sound. Double-clicking the scope will display the output (**Figure 2-5A**).

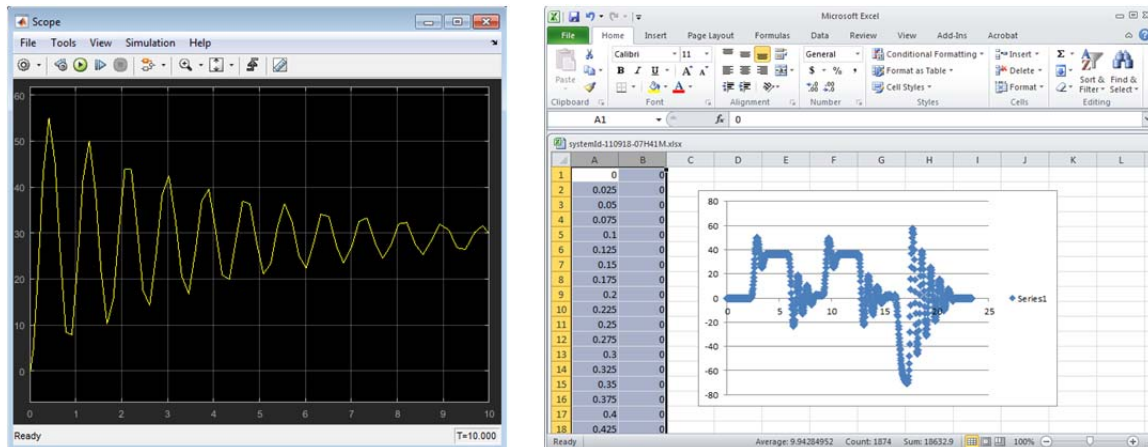


Figure 2-5A: Simulink output (left) shows steady-state $\theta_{ss} \approx 31^\circ$. In the previous lab, an experimental step response was captured and plotted (right). The two show similarities, given confidence in the transfer function derived in **Figure 1-1C**.

Concept 3: Comparing Simulation and Experimental Data

Simulink can import CSV data. This is important because one can overlay experimental and simulation data, for comparison.

Step 1: In Excel, open your CSV file and select data

In a previous lab, the experimental step response of the LEGO DCP to a 50% motor command. **Figure 3-1A left** shows sample data (e.g. `dcp081218-1445.csv`) opened in Excel. When this experiment was performed, the first few seconds, nothing happened; the step command was given a few seconds later. Thus, select the range of time where the DCP's step response was observed. For this sample, this range is approximately 2.7 to 5.7 seconds (**Figure 3-1A right**). Save this set of data as a new CSV file e.g. `dcp081218-1445-Time2.7secTo5.7sec.csv`

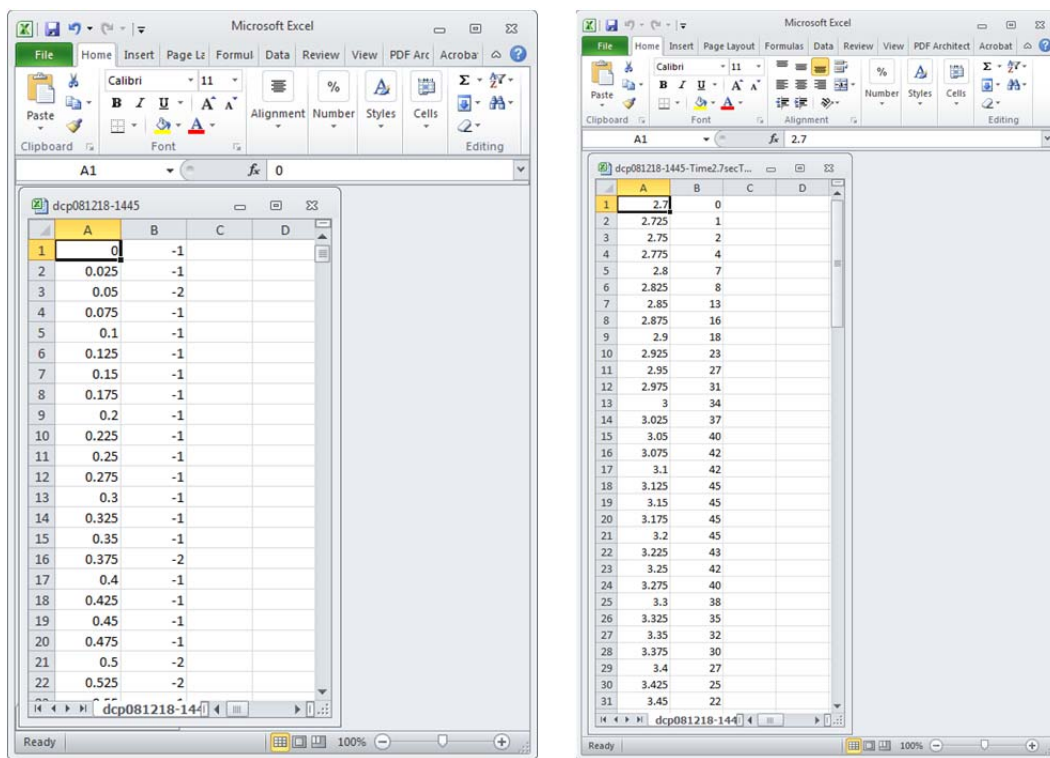


Figure 3-1A: Open the experimental step response data (a CSV file) in Excel (left). Identify the data range when the 50% step command was issued; in this particular data set, 2.7 seconds (cell A1) is when the step was issued. Thus rows of data before 2.7 seconds were deleted (right). Not seen is data after 5.7 seconds that was also deleted. For plotting purposes, only data between 2.7 and 5.7 seconds will be used.

Step 2: Rerun Simulink and plot the scope's simulation

Recall and perform the steps that created the Scope plot (i.e. **Figure 2-5A left**). Recall that experimentally (in the example above), the step command was given at 2.7 seconds. Thus, in Simulink, double-click the Step block. In the Block Parameters pop-up box, enter Step time as 2.7 (**Figure 2-2A left**). Run the simulation to get the step response (**Figure 2-2A right**). One notes that the step begins at 2.7 seconds.

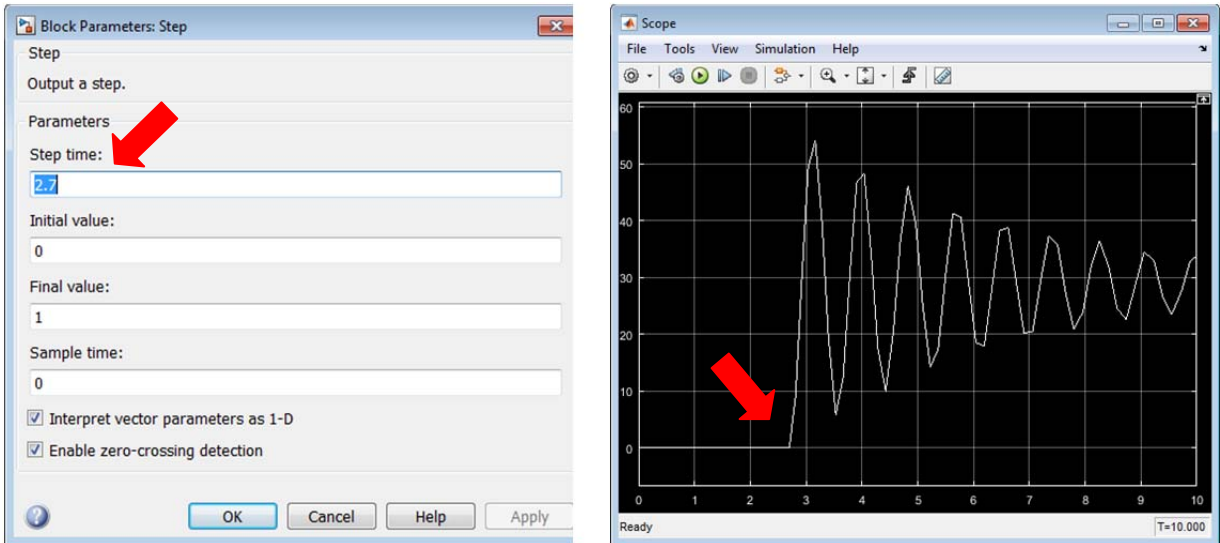


Figure 3-2A: Red arrow shows that the step command in Simulink set to 2.7 seconds (left). Running the simulation, the red arrow shows that the step command happens at 2.7 seconds (right)

Step 3: Log Simulink Scope Data

From the menu bar of the Scope figure, click View – Configuration Properties (**Figure 3-3A left**). Next click on the Logging data tab and check the boxes Limit data points to last and Log data to workspace and click OK (Figure 3-3A right). By default, Simulink sets the Variable name as ScopeData. One could change this e.g. legoDcpSimulation.

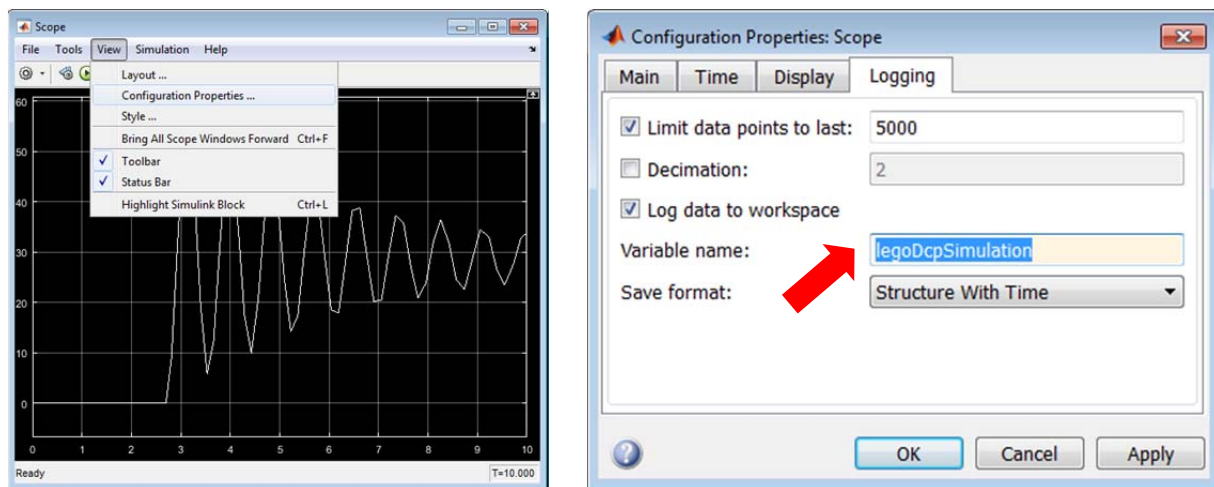


Figure 3-3A: Log the scope data file by selecting View – Configuration Properties (left) and clicking the Logging tab. Make sure to check the boxes (right) and give a suitable variable name e.g. legoDcpSimulation. Red arrow and red circles serve as reference

Step 4: Create Matlab Figure of Simulink Scope Display

The aforementioned step created a new variable in Matlab, named `legoDcpSimulation`. Rerun your simulation – this will now store that data into that variable.

In the Matlab console type (see **Figure 3-4A left**):

```
tSimulation = legoDcpSimulation.time;
angleSimulation=legoDcpSimulation.signals.values;
hold on;
title('LEGO DCP Step Response to 50% Motor Command');
xlabel('Time [sec]'); ylabel('Angle [deg]');
plot(tSimulation, angleSimulation)
```

This will result in a new Figure (see **Figure 3-4A right**) that was previously seen on the Scope. The above Matlab statements assign the variables named `tSimulation` and `angleSimulation` to the time and angle data logged from simulation (i.e. `legoDcpSimulation`). Since we'll be overlaying this figure, the Matlab statement `hold on`, sets the displayed figure as the one that will be manipulated. Lastly, titles and axes labels are given and the data is plotted

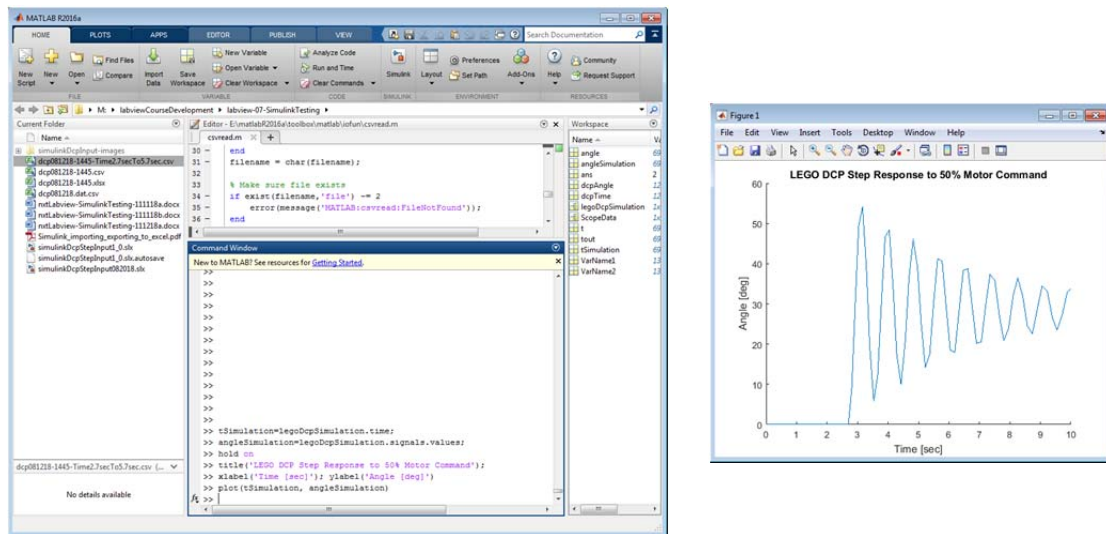


Figure 3-4A: Matlab console allows one to type in statements (left) and results in a Figure (right).

Step 5: Read CSV file and plot onto Matlab Figure

The goal now is to overlay the CSV file from Step 1 (e.g. `dcp081218-1445-Time2.7secTo5.7sec.csv`) on the Matlab Figure from Step 4. First, set the Matlab path to point the folder containing this CSV file (**Figure 3-5A**).

LEGO NXT LabVIEW

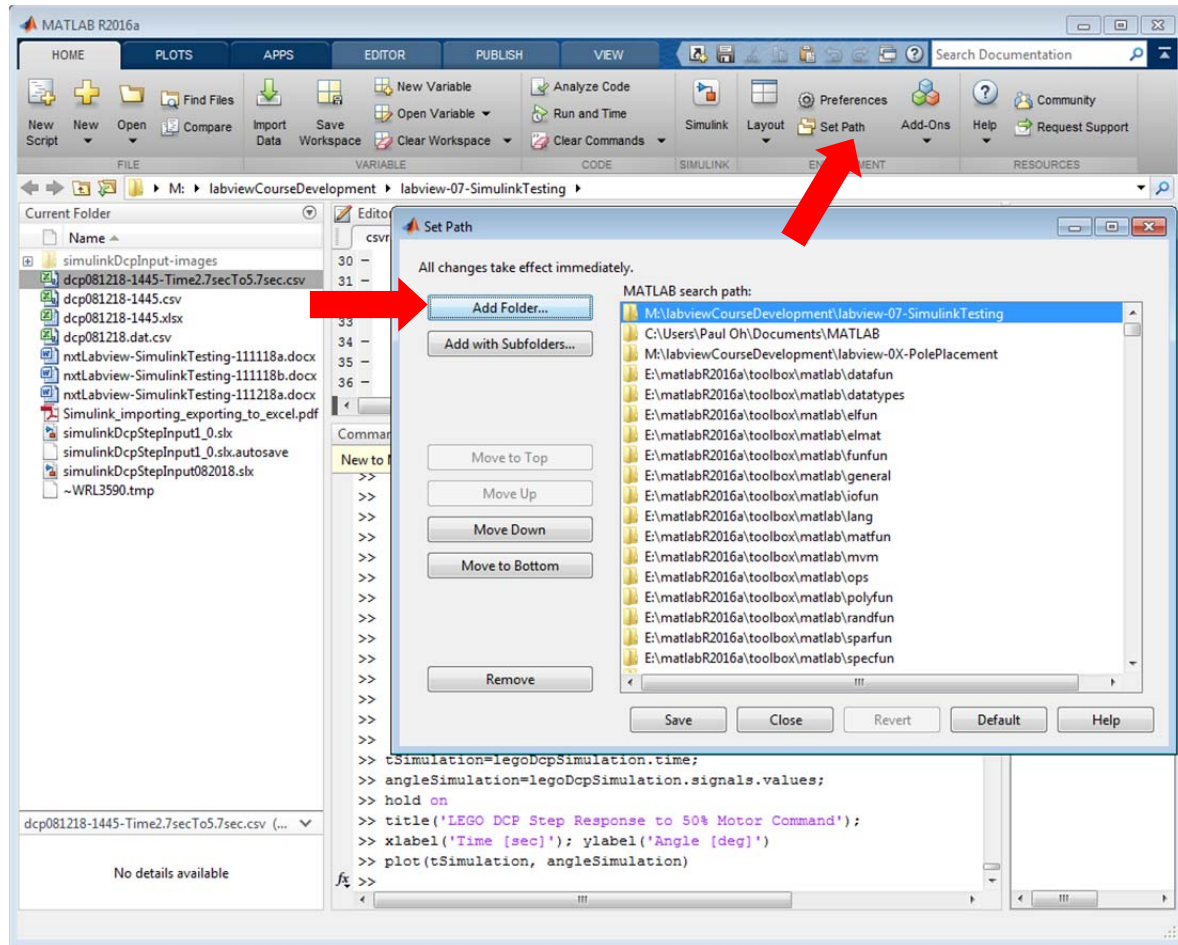


Figure 3-5A: In the Matlab console, the red arrow above points to the Set Path button. Clicking this opens a dialog box and clicking the Add Folder button, allows one to browse for the folder where the CSV file is located. Click Save and Close will allow Matlab to find your CSV file for future reference.

In the Matlab console's menu, click **Import Data** and select the desired CSV file (**Figure 3-5B left**). This will pop up a box, showing your CSV data. Double-clicking the Column heading, will allow you to rename the variables. Rename columns A and B as `tExperimental` and `angleExperimental` respectively (**Figure 3-5B right**). Make sure Column vectors is highlighted. Under the pull-down menu item **Import Selection**, select **Import Data**.

One can now close the Data Import box.

Next, in the Matlab console type: `plot(tExperimental, angleExperimental)`. This will overlay the CSV data (experimental) over the Matlab Figure (see **Figure 3-6A**).

Figure 3-6A allows one to contrast experimental results with a simulation of the calculated system identification. One observes that the period appears quite similar. However, the amplitudes differ a bit. Moreover, if a longer duration of data was plotted, it appears that both simulation and experimental data suggest a steady-state angle will eventually converge to about $\theta_{ss} \approx 31^\circ$. This provides some confidence in the transfer function that was analytically derived in a previous lab.

LEGO NXT LabVIEW

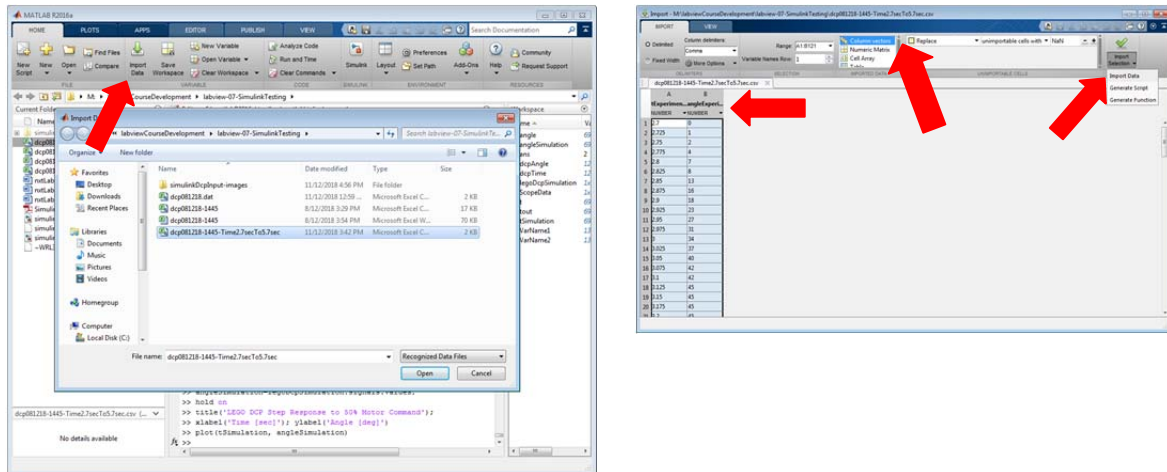


Figure 3-5B: One imports their CSV file by clicking the Import Data button in the menu bar as seen by the red arrow (left). One can now see some of their CSV data in columns. These columns were renamed as `tExperimental` and `angleExperimental` (left red arrow). The Column vectors choice is highlighted (middle red arrow) and then imported (right red arrow 3) (right)

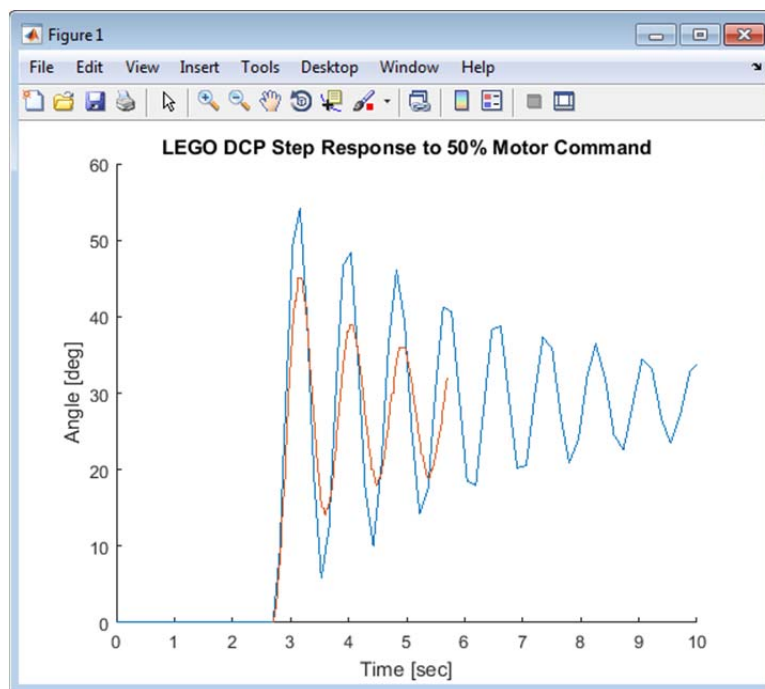


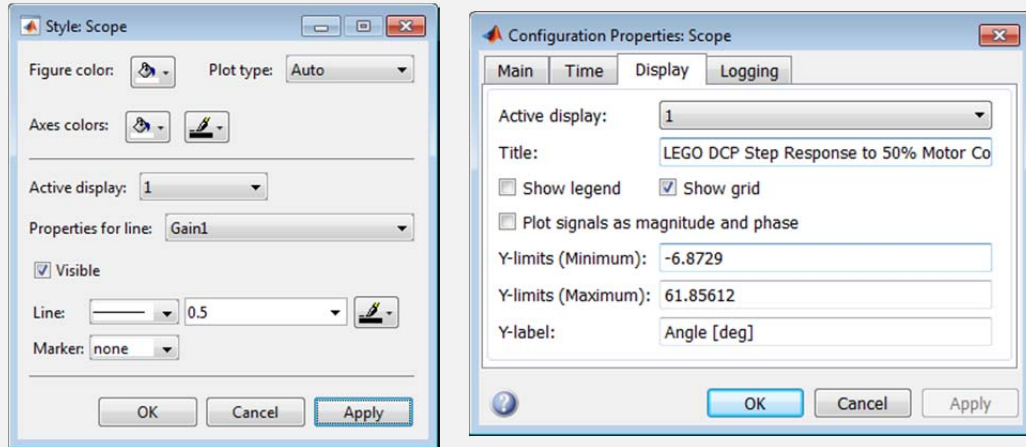
Figure 3-6A: One sees the experimental data (from the CSV) file colored in red plotted over the Simulink simulation data, colored in blue.

Congratulations: LEGO Damped Compound Pendulum Simulation Completed

Exercise 2:

2.1 What are some of the reasons that may explain the differences in Figures 2-5A and 3-6A?

2.2 Study Matlab/Simulink to change the scope's black background to white and add labels to the axes. Hint 1: Right-click on the figure, and select *Style*. Hint 2:



2.3 Plot a longer duration of data to re-create Figure 3-6A with a longer duration e.g. 10 seconds. What's the steady-state angle θ_{ss} for simulation and for experimental results?