

AER821

Spacecraft Attitude Dynamics and Control

Hardware Introductory Lab

MATLAB Simulink Tutorial

Fall 2025

Instructor: Dr. John Enright

C. Chan

Rev 1.1

## **0 General Safety Rules AND Regulations for Laboratories and Research Areas**

The following safety rules and regulations are to be followed in all Aerospace Engineering laboratories and research facilities. These rules and regulations are to insure that all personnel working in these laboratories and research areas are protected, and that a safe working environment is maintained.

1. "Horseplay" is hazardous and will not be tolerated.
2. No student may work alone in the laboratory at any time, except to prepare operating procedures for equipment or data write-up/reduction/simulations.
3. Required personal protective equipment (PPE) will be provided by the Department for use whenever specified by the Faculty, Engineering Support or Graduate/Teaching Assistant, .i.e., hearing protection, face shields, dust masks, gloves, etc.
4. Contact lenses will not be worn in the laboratory when vapours or fumes are present.
5. Safety glasses with side shields and plastic lenses will be required when operating targeted class experiments as outlined in the experimental procedures. Splash goggles or face shields will also be provided and worn also, for those experiments which have been identified as a requirement.
6. Each student must know where the location of the First Aid box, emergency equipment, eye wash station is, if required in the laboratories, shops, and storage areas.
7. All Faculty, Engineering Support and Graduate/Teaching Assistants must know how to use the emergency equipment and have the knowledge to take action when an accident has occurred, .i.e., emergency telephone number, location, emergency response services.
8. All Faculty, Engineering Support and Graduate/Teaching Assistants, and Research Assistants, must be familiar with all elements of fire safety: alarm, evacuation and assembly, fire containment and suppression, rescue.
9. Ungrounded wiring and two-wire extension cords are prohibited. Worn or frayed extension cords or those with broken connections or exposed wiring must not be used. All electrical devices must be grounded before they are turned on.
10. All Faculty, Engineering Support and Graduate/Teaching Assistants, and Research Assistants, must be familiar with an approved emergency shutdown procedure before initiating any experiment.
11. There will be NO deviation from approved equipment operating procedures.

12. All laboratory aisles and exits must remain clear and unblocked.
13. No student may sniff, breathe, or inhale any gas or vapour used or produced in any experiment.
14. All containers must be labeled as to the content, composition, and appropriate hazard warning: flammable, explosive, toxic, etc.
15. The instructions on all warning signs must be read and obeyed in all laboratories and research facilities.
16. All liquid and solid waste must be segregated for disposal according to Faculty, Engineering Support or Graduate/Teaching Assistant instructions. All acidic and alkaline waste should be neutralized prior to disposal. NOTE: NO organic waste material is to be poured down the sink or floor drains. These wastes should be properly placed in designed waste disposal containers, labeled and stored in the department's flammable storage cabinet which is ventilated and secured.
17. Good housekeeping must be practiced in all teaching and research laboratories, shops, and storage areas.
18. Eating, drinking, tobacco products, gum chewing or application of makeup is strictly prohibited in the laboratories and shops.
19. Only chemicals may be placed in the "Chemicals Only" refrigerator. Only food items may be placed in the Food Only refrigerator. Ice from any refrigerator is not be used for human consumption or to cool any food or drink.
20. Glassware breakage must be disposed in the cardboard boxes marked "Glass Disposal". Any glassware breakage and malfunctioning instruments or equipment must be reported to the Faculty, Engineering Support or Graduate/Teaching Assistant present.
21. All injuries, accidents, and "near misses" must be reported to the Faculty, Engineering Support or Graduate/Teaching Assistant. The Accident Report must be completed as soon as possible after the event by the Faculty, Engineering Support or Graduate/Teaching Assistant and reported to the Departmental Safety Officer immediately. Any person involved in an accident must be sent or escorted to the University Health Centre. All accidents are to be REPORTED.
22. All chemical spills are to be reported to the Faculty, Engineering Support or Graduate/Teaching Assistant, whose direction must be followed for containment and cleanup. Faculty, Engineering Support or Graduate/Teaching Assistant will follow the prescribed instructions for cleanup and decontamination of the spill area. The Departmental Safety Officer must be notified when a major spill has been reported.
23. All students and Faculty, Engineering Support or Graduate/Teaching Assistant must wash their hands before leaving targeted laboratories, research facilities or shops.
24. No tools, supplies, or any other items may be tossed from one person to another.
25. Compressed gas cylinders must be secured at all times. Proper safety procedures must be followed when moving compressed gas cylinders. Cylinders not in use must be capped.

26. Only gauges that are marked "Use no oil" are for Oxygen cylinders. Do not use an oiled gauge for any oxidizing or reactive gas.
27. Students are never to play with compressed gas hoses or lines or point their discharges at any person.
28. Do not use adapters or try to modify any gas regulator or connection.
29. There will be no open flames or heating elements used when volatile chemicals are exposed to the air.
30. Any toxic chemicals will be only be exposed to the air in a properly ventilated Fume Hood. Flammable chemicals will be exposed to the air only under a properly ventilated hood or in an area which is adequately ventilated.
31. Personal items brought into the laboratory or research facility must be limited to those things necessary for the experiment and safe operation of the equipment in the laboratories and research facilities.
32. General laboratory coats, safety footwear are not provided by the Department of Aerospace Engineering, although some targeted laboratories and research areas will be supported by a reasonable stock of protective clothing and accessories, i.e., gloves, welding aprons, dust masks, face shields, safety glasses, etc.
33. Equipment that has been deemed unsafe must be tagged and locked out of service by the Technical Officer in charge of the laboratory or research facility. The Departmental Safety Officer must be notified of the equipment lockout IMMEDIATELY!
34. In June 1987 both the Federal & Ontario Governments passed legislation to implement the workplace hazardous material information system or WHMIS across Canada. WHMIS was designed to give workers the right-to-know about hazardous material to which they are exposed to on the job. Any person who is required to handle any hazardous material covered by this act should first read the label and the product's material safety data sheet (MSDS). No student is to handle any hazardous materials unless supervised by a Faculty, Engineering Support or Graduate/Teaching Assistant. The laboratory Technical Officer, Faculty, Engineering Support or Graduate/Teaching Assistant is responsible for ensuring that any hazardous materials are stored safely using WHMIS recommended methods and storage procedures. All MSDS must be displayed and stored in a readily accessible place known to all users in the workplace and laboratory
35. All the foregoing rules and regulations are in addition to the Occupational Health and Safety Act, 1987.
36. Casual visitors to the laboratory and research areas are to be discouraged and must have permission from the Faculty, Engineering Support or Graduate/Teaching Assistant to enter. All visitors must adhere to the safety guidelines and is the responsibility of the visitor.
37. Only the Safety Officer may make changes to these policies upon confirmation of the Safety Committee and approval of the Department Chair.

## TABLE OF CONTENTS

0	General Safety Rules AND Regulations	1
1	Getting Started with MATLAB Simulink	5
1.1	Introduction to MATLAB Simulink _____	5
1.2	Constructing block diagrams with the Simulink Library _____	6
2	Position (Close-loop) Control of a DC Motor Unit	13
2.1	Introduction to model-based simulation _____	13
2.2	Create a closed-loop position control model for the DC motor unit _____	17
2.3	Create a Hardware-in-the-loop position control of the DC motor unit _____	27
3	References	28

# 1 Getting Started with MATLAB Simulink

## OBJECTIVE

Get familiar with the MATLAB Simulink programming environment

## LEARNING OUTCOME

Upon the completion of Part 1, students should be able to demonstrate the followings:

- construct block diagrams with the Simulink customized block libraries
- carry out data transfer between the main framework of MATLAB and the Simulink environment
- construct open-loop model-based system in Simulink

## 1.1 Introduction to MATLAB Simulink

Important: The lab description in Part-1 is *optional*. It's intended as a refresher/tutorial if you are unsure of your Simulink and Matlab skills. If you choose to complete this part, you should do so prior to attending the lab.

MATLAB Simulink is a graphical programming or block-diagram based environment for multi-domain (e.g. time-domain and frequency-domain) simulation and model-based design. It supports system-level design, simulation, automatic code generation, and continuous test and verification of embedded systems. Simulink provides a graphical editor, customizable block libraries, and solvers for modeling and simulating dynamic systems. It is integrated with MATLAB, enabling you to incorporate MATLAB algorithms into models and export simulation results to MATLAB for further analysis [1].

The diagram below in Figure 1 depicts an example of a feedback (close-loop) control system modeled in Simulink. The system starts off with an output signal generated from the **Source** block called *Constant* on the left hand side of the diagram (Note that a **Source** block only has one signal port (output)). The signal then travels to the next block called *Subtraction* which calculates the difference between the *Constant* and the feedback signal. The signal (the difference) then travels downstream of the path and reaches the next block on the right, and it keeps going downstream until it reaches the final destination, the **Sink** block called *To Workspace* where the value(s) of the signal will be stored in a variable (array) called *simout* (Note that a **Sink** block only has one signal port (input)).

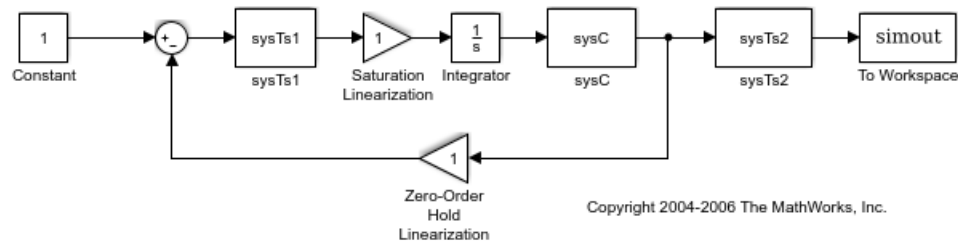


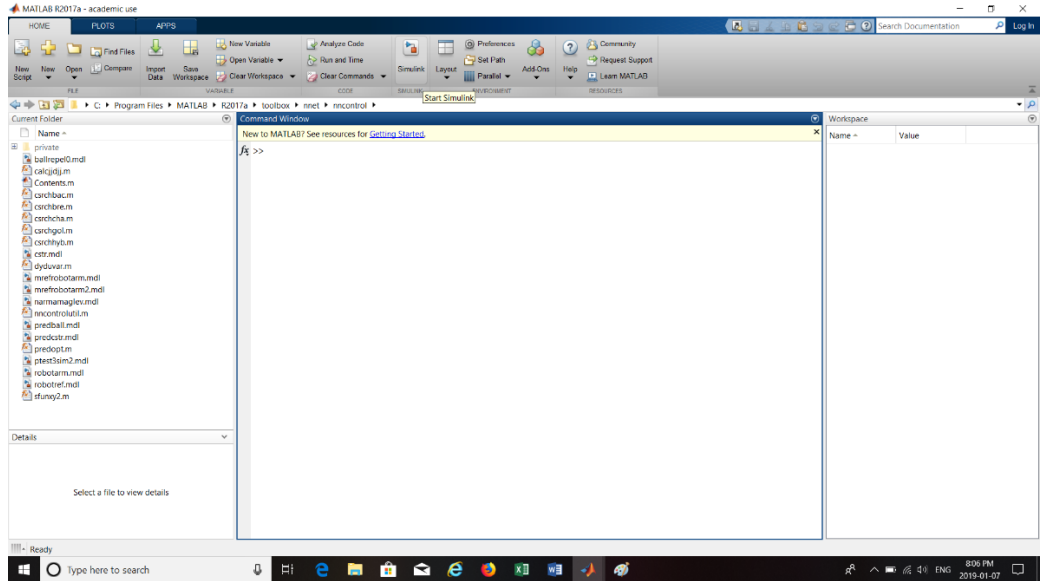
Figure 1: An example of a feedback control system modeled in Simulink [1]

## 1.2 Constructing block diagrams with the Simulink Library

This section explains how to construct a simple block diagrams using the Simulink Library

**Apparatus:** 1 X PC station with Windows 7 or later version installed

1 X MATLAB R2017b or later version

INSTRUCTION		
STEP		
✓	1	Launch the MATLAB program.
	2	<p>From the top menu bar, select the <b>Simulink</b> button as shown in Figure 2.</p>  <p>Figure 2: Launching the Simulink environment</p>
	3	<p>The <b>Simulink Start Page</b> should now appear as shown in Figure 3. Now select <b>Blank Model</b> from the top to start building your model from scratch.</p>

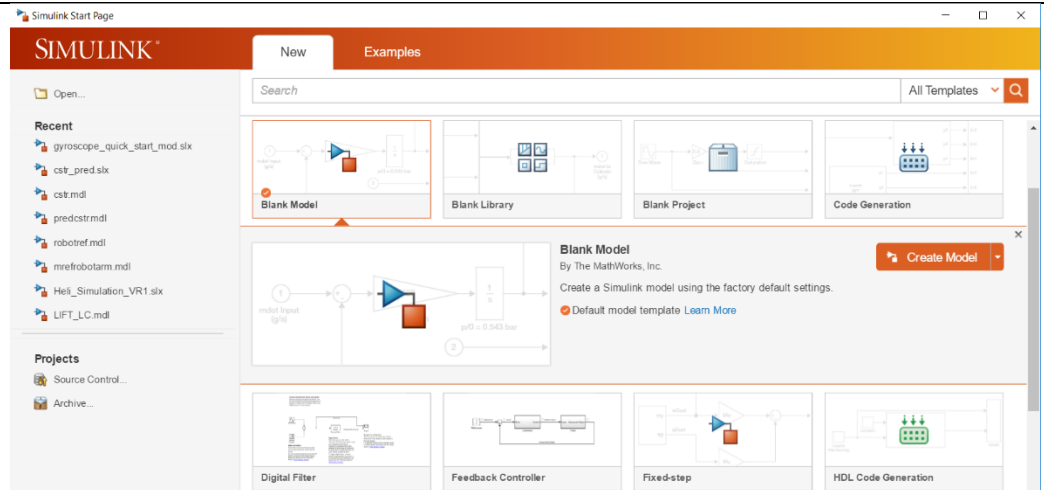


Figure 3: Simulink Start Page

Now you should see a new window pops up as shown in Figure 4. Select the **Library Browser** from the top menu bar to access the predefined block diagrams from Simulink.

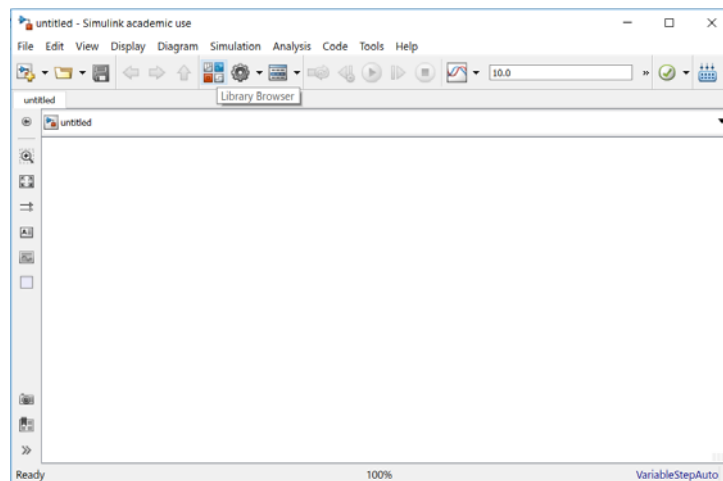
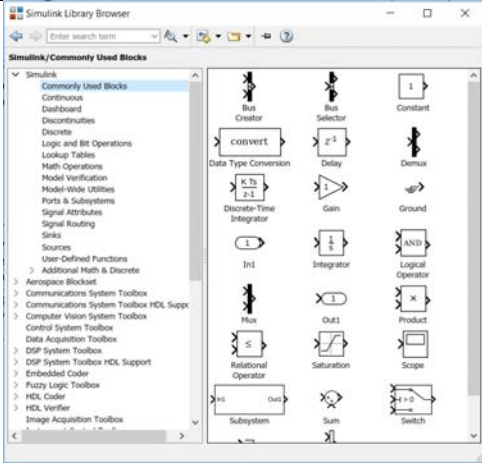
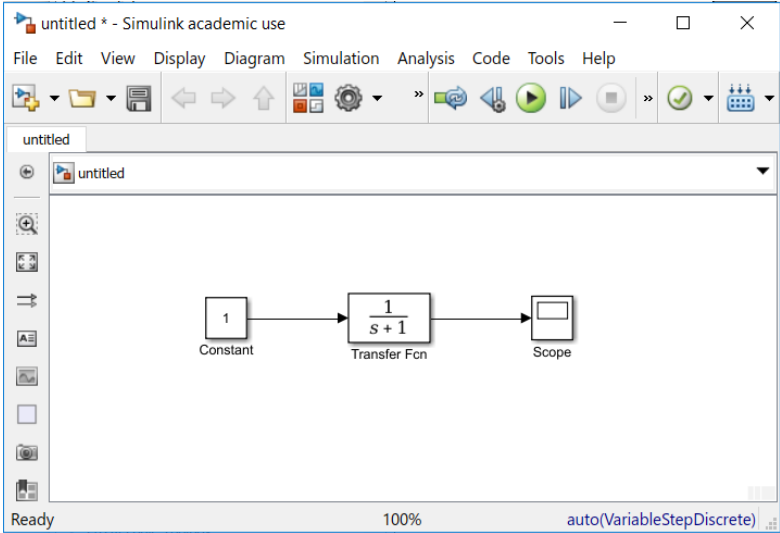


Figure 4: Launching Library Browser



5	<p>Imagine we have a RC (Resistor-Capacitor) Circuit (Figure 5) we want to model in Simulink and we want to observe the buildup of voltage across the terminals of the capacitor, <math>V_c</math> with respect to time.</p> <div data-bbox="764 327 1073 485" data-label="Diagram"> </div> <p>Figure 5: RC Circuit</p> <p>The RC Circuit can also be used to modeled a pneumatic system having a valve (analogous to (flow) resistance) and a vessel (analogous to a capacitor). In this case, we can model the change of vessel pressure with respect to time.</p>
6	<p>The relationship between the input voltage, <math>V_s</math> and the output voltage, <math>V_c</math> in Figure 5 is governed by the following frequency-domain equation:</p> $V_c(s) = \frac{1}{1+RCs} V_s(s) \quad (1)$ <p>And the corresponding transfer function that transforms the input to the output is represented by Equation (2) below:</p> $T(s) = \frac{V_c(s)}{V_s(s)} = \frac{1}{1+RCs} \quad (2)$ <p>Equation (2) represents a system that demonstrates a first-order natural response to the input signal. The product of the resistance <math>R</math> and capacitance <math>C</math> represents the time constant, <math>\tau</math> of the system. <math>\tau</math> can be defined as the time it takes for the output of the system to reach 63% of the final or steady-state value.</p>
7	<p>To construct a model governed by Equation (2), we go to <b>Commonly Used Blocks</b> from <b>Simulink Library Browser</b> to select blocks to construct your model as shown in Figure 6. For your convenient access, the <b>Commonly Used Blocks</b> panel stores blocks that you frequently use to construct your model.</p>

		 <p>Figure 6: Simulink Library Browser</p>
8		<p>Drag and drop a <b>Constant</b> block onto the newly opened window. Then, go to the <b>Continuous</b> directory from the left and do the same for a <b>Transfer Fcn</b> block. Next, do the same for a <b>Scope</b> block that you can find under the <b>Sinks</b> directory. We are going to use a <b>Scope</b> to show a time plot of the output signal <math>V_C</math>.</p>
9		<p>Lastly, put your mouse cursor on the output port of the Constant block, left click your mouse and hold the button down. Drag your mouse cursor towards the input port of the block downstream of the <b>Constant</b> block (<b>Transfer Fcn</b> block in this case) and release the left click button. Now you should send a solid line with an arrow head connecting the two blocks. Repeat the same procedure for the rest of the blocks in your model file. Your model should look like the one depicted in Figure 7 upon completion.</p>  <p>Figure 7: Simulink model of a RC circuit</p>

Before running the simulation of the model, you can go to the top menu bar and select **Simulation >> Model Configuration Parameters** to configure your simulation criterion. Here, you can define, for example, the simulation time and a wide variety of other modeling features and criterion. In most cases, when we observe the dynamic response of a system, we are interested in knowing its steady-state performance. To observe the steady-state performance, we have to ensure the duration of the simulation is long enough for the system to reach steady state. Therefore, it is important to change the simulation time according to your model dynamics. You can do so under the **Solver** section as depicted in Figure 8 and adjust the **Stop time**. Click **OK** after the changes are made. Now run the simulation and see what you get from the **Scope** (You will have to double click the Scope to see the signal plot).

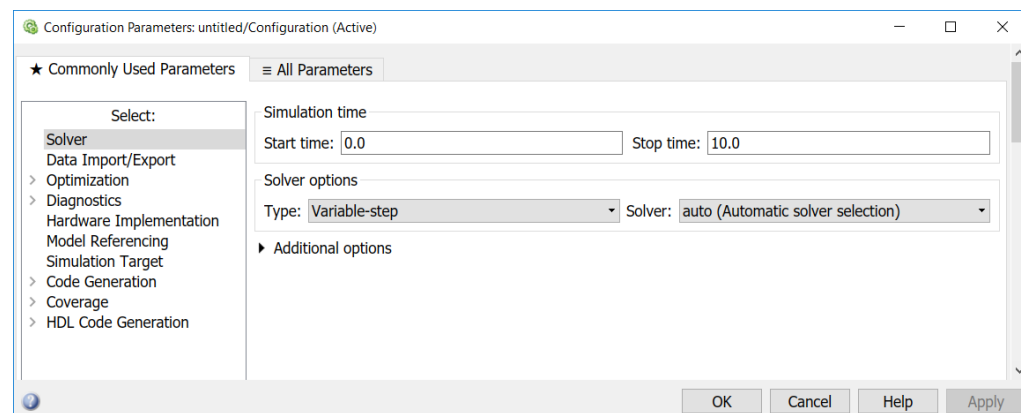


Figure 8: Configuration Parameters menu

You can modify the parameters of the **Transfer Fcn** by double clicking the block. The coefficient associated with the open-loop pole **s** here (highlighted in blue in Figure 9) defines the time constant  $\tau$ . Changing the value of this parameter will change the time it takes for the system to reach steady state.

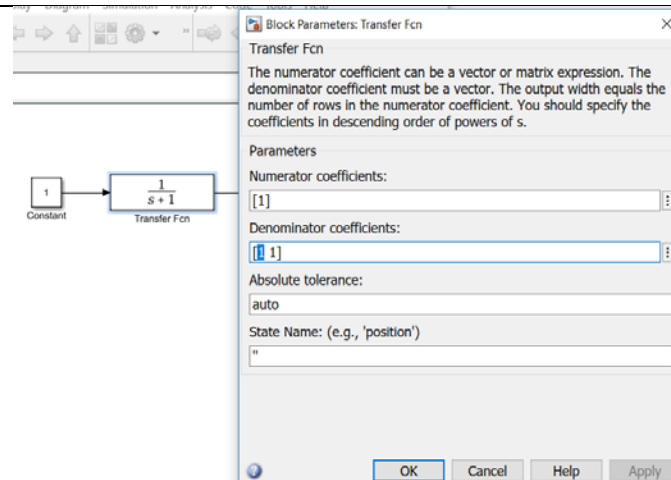


Figure 9: Block parameter configuration

After you have finished running the simulation, go back to the MATLAB main window and look for the variable called *tout* from the **Workspace** panel. Double click *tout* to see if you can view the values as shown in Figure 10.

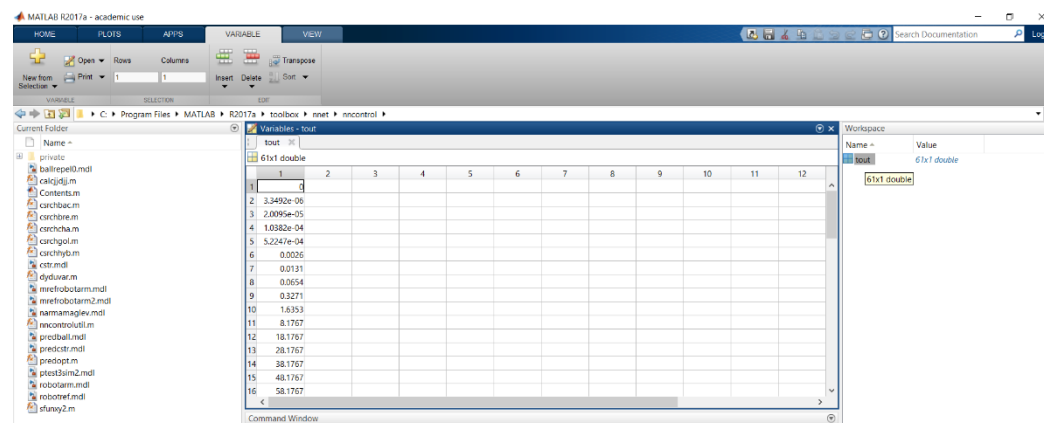


Figure 10: Variable storage in Workspace

By default, MATLAB will store the time stamps of your Simulink model to a variable (1-D array) called *tout* unless you define a variable yourself to store this data. Each value of *tout* is an element of the time series and is recorded according to the defined sampling time of your model. The default setting of MATLAB allows the solver to execute variable-step numeric iteration and the step size is set to auto-adjust mode as shown in Figure 11. Thus, it is difficult to determine the actual sampling rate that is being used. The user, however, can select the solver step type by selecting the options from the drop-down menu under **Solver options** (Figure 11). For example, if you want to define a fixed sampling time of 0.1 second for your simulation, simply change the solver step type from *Variable-step* to **Fixed-step** and change the **Fixed-step size** under the

**Additional options** from *auto* to **0.1**. So after you finished running your simulation for a duration of say, 100 seconds, you will generate a time series that consists of 1001 elements with  $t = 0$  inclusive.

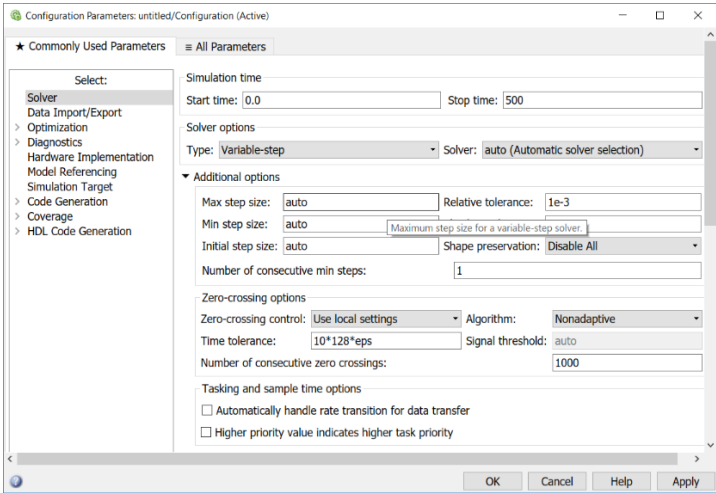


Figure 11: Solver options

To define your own user-defined variable to store the output data, you need to go back to your Simulink model file and add a block from the **Simulink Library Browser** called **To Workspace** from within the **Sink** panel. Rename the block from *simout* to **myData**. Your model should now look like the one shown in Figure 12.

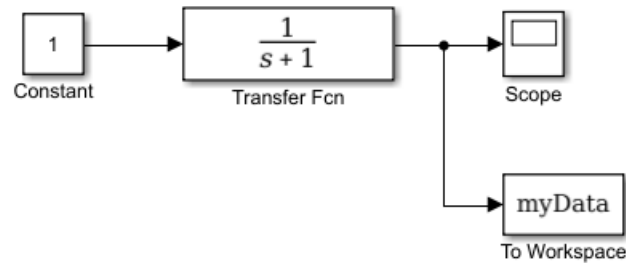


Figure 12: Simulink model with To Workspace output redirection

Now, go to the main MATLAB window and double click the variable **myData**. You should see that this variable consists of two columns of 1-D arrays, the one on the left being the time stamps (values should be identical to the values shown in *tout*) and the one on the right being the output values. If you right click on **myData** from the **Workspace** panel and select **Save As...**, you can actually save the data in a **MATLAB Data File** with the extension **.mat**. If you need to reuse these data at a later time you can simply go to the top menu bar of the MATLAB main window, click **Import Data** from the **Home** tap and locate your data file, then you can load the data to MATLAB.

13	<p>Now consider the following example. Imagine that we have a water system that consists of a one-inch diameter pipe and a one-inch opening valve whose flow coefficient, <math>C_v</math> is 14 [Gallons/min-psi<sup>-1/2</sup>] and whose flow resistance coefficient, <math>K_v</math> can be calculated with Equation (3) as:</p> $K_v = 890.9 \frac{d^4}{C_v^2} \quad (3)$ <p>where d is the diameter of the valve measured in inches. Downstream of the valve there is a 12-gallon tank that is used to store the water being pumped into it. To simplify the problem, we assume the pump provides a constant pumping pressure of 60 psi to feed the water into the tank. Now try to construct a simulation model for this water system. Estimate the time it takes for the system to reach steady state and its time constant, <math>\tau</math> measured in seconds.</p>
----	--

## 2 Position (Closed-loop) Control of a DC Motor Unit

### OBJECTIVE

Model a close-loop control system in Simulink and control the angular position of a DC motor unit in real time

### LEARNING OUTCOME

Upon the completion of Part 2, students should be able to demonstrate the followings:

- construct a model-based system in Simulink
- model a physical system with dynamics
- perform close-loop position control simulation with a PID controller

### 2.1 Introduction to model-based simulation

#### Apparatus:

- 1 X PC station with Windows 7 or later version installed
- 1 X MATLAB R2017b or later version

Any physical system can be modeled with Simulink as long as we have all the required governing equations defined. In the case of a DC motor, which is an electromechanical device that develops a displacement output for a voltage input. The schematic for a particular type of dc motor is shown in Figure 13 [2].

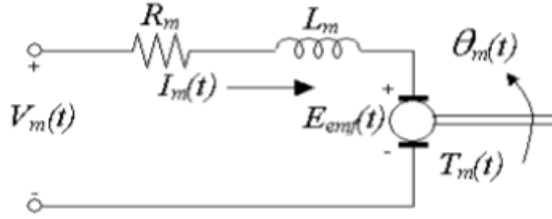


Figure 13: Schematic of a dc motor

With a fixed magnetic field created by a permanent magnet (not shown here) and a rotating circuit called the armature with current flow  $I_m$ , passes through the magnetic field at right angles, the electromagnetic force proportional to the current can be described as follows:

$$F = B l I_m(t) \quad (4)$$

where  $B$  is the magnetic field strength and  $l$  is the length of the conductor. The resulting force (torque) turns the rotor (the rotating part of the motor).

In addition, there occurs a counteracting voltage (electromotive force) that impedes the motion of the rotor, called the back e.m.f. (electromotive force),  $E_{emf}$ . A conductor moving at right angles to the magnetic field generates a voltage at the terminals of the conductor equal to:

$$e = B l v \quad (5)$$

where  $e$  is the voltage and  $v$  is the velocity of the conductor normal to the magnetic field. Since the armature is rotating in a magnetic field its voltage is proportional to the speed, hence:

$$E_{emf} = K_m \omega = K_m \dot{\theta}_m \quad (6)$$

And we can apply Kirchhoff's Voltage Law (KVL) to obtain the following equation:

$$V_m = R_m I_m + L_m \frac{dI_m}{dt} + E_{emf} \quad (7)$$

Since the motor inductance is much smaller than the resistance ( $L_m \ll R_m$ ), we can omit the  $L_m$  (typical for a DC motor) leaving us with:

$$I_m = \frac{V_m - E_{emf}}{R_m} = \frac{V_m - K_m \dot{\theta}_m}{R_m} \quad (8)$$

When applying Newton's 2<sup>nd</sup> law of motion to the driving shaft of the motor, we get:

$$J_m \ddot{\theta}_m = T_m - \frac{T_l}{\eta_g K_g} \quad (9)$$

where  $\frac{T_l}{\eta_g K_g}$  is the load torque reflected through the gears to the motor and  $\eta_g$  is the efficiency of the gearbox.

Now applying Newton's 2nd law of motion at the load of the motor:

$$J_l \ddot{\theta}_l = T_l - B_{eq} \dot{\theta}_l \quad (10)$$

where  $B_{eq}$  is the viscous damping coefficient as seen at the output. Substituting Equation (9) into Equation (10), we are left with:

$$J_l \ddot{\theta}_l + \eta_g K_g^2 J_m \ddot{\theta}_l + B_{eq} \dot{\theta}_l = \eta_g \eta_m K_g K_t I_m \quad (11)$$

Finally, we can combine the electrical and mechanical equations by substituting Equation (8) into Equation (11) and transforming the equation to a frequency-domain form gives:

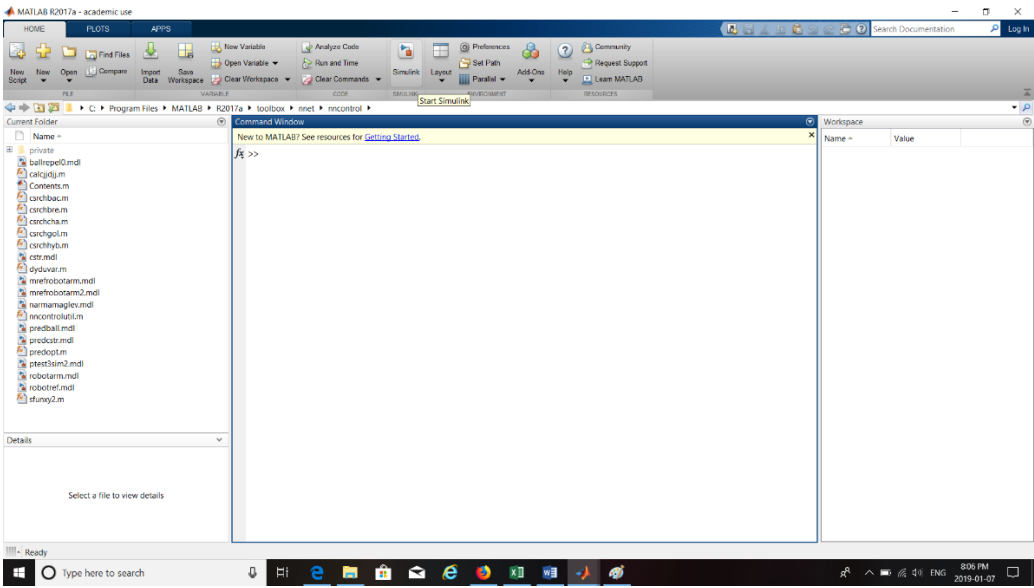
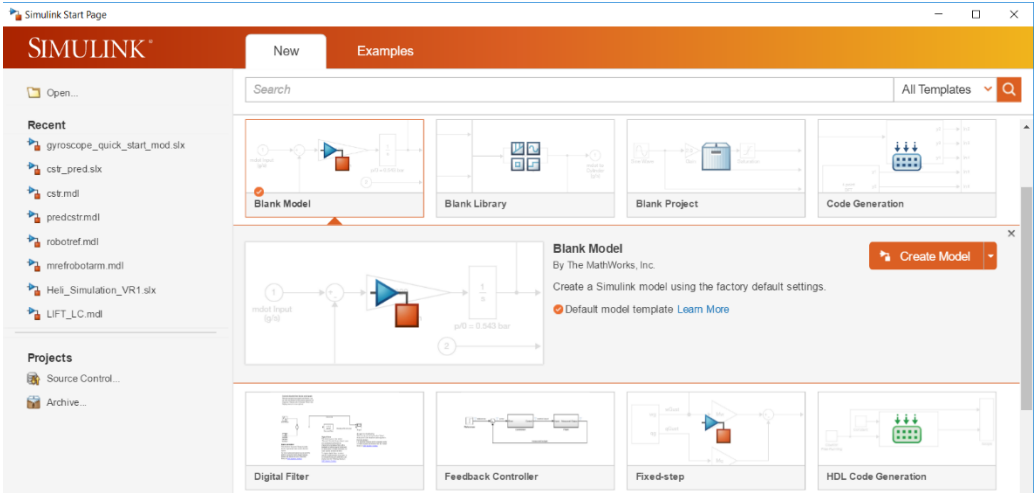
$$G(s) = \frac{\theta_l(s)}{V_m(s)} = \frac{\eta_g \eta_m K_g K_t}{J_{eq} R_m s^2 + (B_{eq} R_m + \eta_g \eta_m K_g^2 K_t^2) s} \quad (12)$$

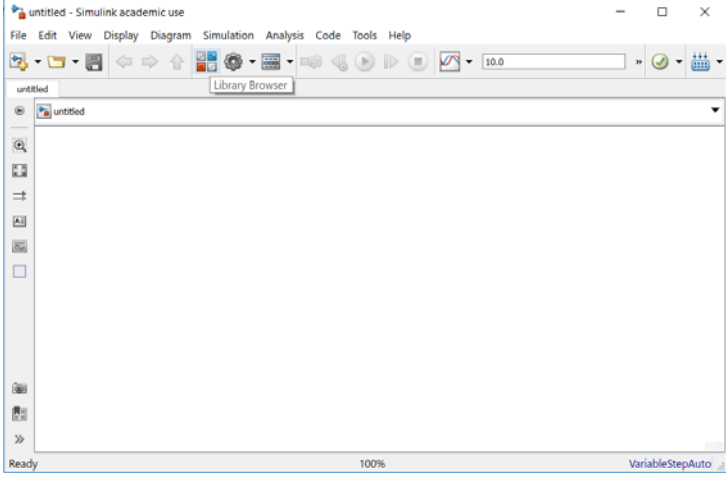
where  $J_{eq} = J_l + \eta_g J_m K_g^2$  is the equivalent moment of inertia of the motor system as seen at the output. Also, the magnitudes of  $K_m$  and  $K_t$  are equivalent when using consistent units (SI or imperial). Table 1 below shows the specifications of the DC motor unit that are required when you create your model-based Simulink file:

Symbol	Description	Matlab Variable	Nominal Value	SI Units	Variation [%]
$V_m$	Armature circuit input voltage			V	
$I_m$	Armature circuit current			A	
$R_m$	Armature resistance	<b>Rm</b>	<b>2.6</b>	$\Omega$	12
$L_m$	Armature inductance			H	
$E_{emf}$	Motor back-emf voltage			V	
$\theta_m$	Motor shaft position			rad	
$\omega_m$	Motor shaft angular velocity			rad/s	
$\theta_l$	Load shaft position			rad	
$\omega_l$	Load shaft angular velocity			rad/s	
$T_m$	Torque generated by the motor			N-m	
$T_l$	Torque applied at the load			N-m	
$K_m$	Back-emf constant	<b>Km</b>	<b><math>7.67 \times 10^{-3}</math></b>	V/rad/s	12
$K_t$	Motor-torque constant	<b>Kt</b>	<b><math>7.67 \times 10^{-3}</math></b>	N-m/A	12
$J_m$	Motor moment of inertia	<b>Jmotor</b>	<b><math>3.87 \times 10^{-7}</math></b>	$K_g \cdot m^2$	
$J_{eq}$	Equivalent moment of inertia at the load	<b>Jeq</b>	<b><math>2.1 \times 10^{-3}</math></b>	$K_g \cdot m^2$	10
$B_{eq}$	Equivalent viscous damping coefficient	<b>Beq</b>	<b><math>4.0 \times 10^{-3}</math></b>	N-m/rad/s	20
$K_g$	High Gear Ratio (motor->load)	<b>Kg</b>	<b>70</b>	--	
$\eta_g$	Gearbox efficiency	<b>Eff_G</b>	<b>0.9</b>	--	10
$\eta_m$	Motor efficiency	<b>Eff_M</b>	<b>0.69</b>	--	
$\omega_n$	Undamped natural frequency	<b>Wn</b>		rad/s	
$\zeta$	Damping ratio	<b>zeta</b>		--	
$K_p$	Proportional gain	<b>Kp</b>		V/rad	
$K_v$	Velocity gain	<b>Kv</b>		V/rad/s	
$T_p$	Time to peak	<b>Tp</b>		s	

Table 1: DC motor technical specifications [3]



✓	STEP	INSTRUCTION
	1	Launch the MATLAB program.
2		<p>From the top menu bar, select the <b>Simulink</b> button as shown in Figure 14.</p>  <p>Figure 14: Launching the Simulink environment</p>
3		<p>The <b>Simulink Start Page</b> should now appear as shown in Figure 15. Now select <b>Blank Model</b> from the top to start building your model from scratch.</p>  <p>Figure 15: Simulink Start Page</p>

4		<p>Now you should see a new window pops up as shown in Figure 16. Select the <b>Library Browser</b> from the top menu bar to access the predefined block diagrams from Simulink.</p>  <p>Figure 16: Launching Library Browser</p>
5		<p>Construct a DC motor model in Simulink using the transfer function described in Equation (12). Use the <b>MATLAB Variable</b> nomenclatures defined in <b>Table 1</b> to construct your model. Use an input voltage, <math>V_m</math> starting from 0 V and gradually step up the voltage with an increment of 0.5 V until 10 V is reached. Each time you step up the voltage, record the steady-state value of the angular rate, <math>\dot{\theta}_l</math>. Plot the graph of <math>\dot{\theta}_l</math> vs <math>V_m</math>, is this a linear graph? Why or why not?</p>

## 2.2 Create a closed-loop position control model for the DC motor unit

### Apparatus:

- 1 X PC station with Windows 7 or later version installed
- 1 X MATLAB R2017b or later version

Up to this point here, we have only investigated the system response of open-loop systems, which means we have not utilized the feedback signal such as the pressure measurement in the case of the water tank system in Section 1.2 or the angular position measurement in the case of the model-based DC motor simulation in Section 2.1. To compensate for the disturbance that may cause the system to behavior erroneously, we have to create a system that incorporates feedback in the calculations so the input or command signal can be modulated constantly during the course of the system operation to minimize the difference between the actual system response and the desired response the user expects.

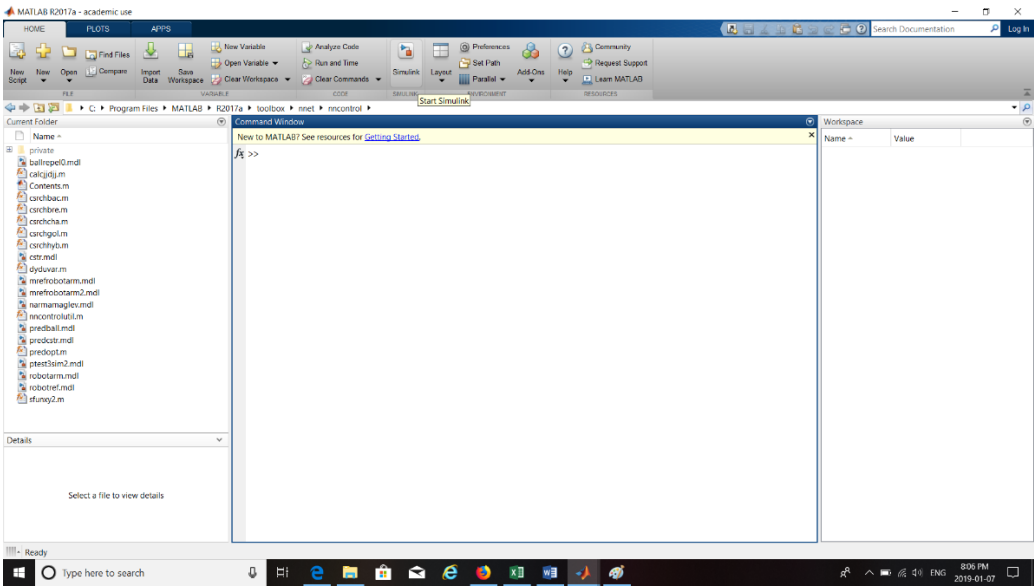
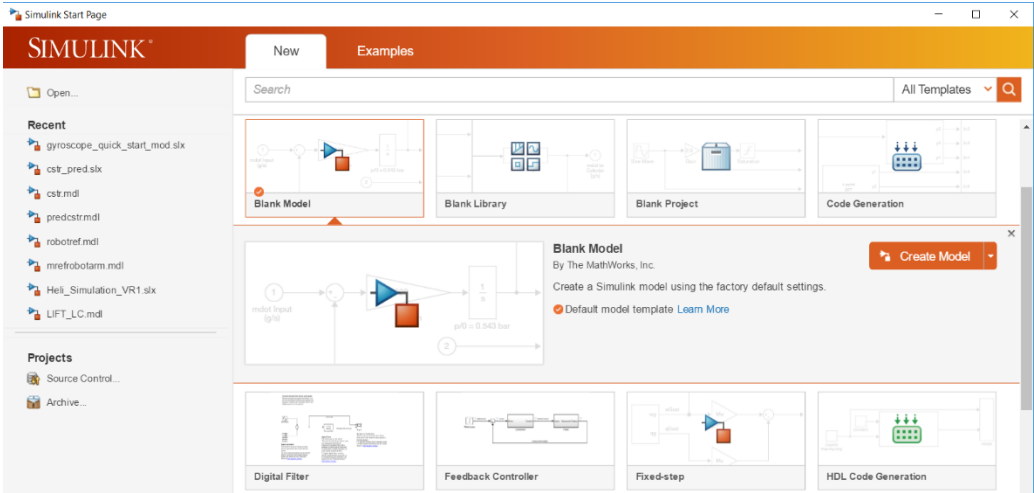
One way to compensate disturbance is to employ a compensator such as a PID controller, which is one of the most popular controllers you will find these days in a wide variety of applications. A PID controller is governed by Equation (13) as follows:

$$u(t) = K_p e(t) + K_i \int_0^t e(t) dt + K_d \frac{d}{dt} e(t) \quad (13)$$

where  $u$  is the input,  $e$  is the error (difference between the set-point and the actual response),  $K_p$ ,  $K_i$ , and  $K_d$  are the proportional gain, integral gain, and derivative gain respectively.

A typical PID controller has three parameters that governed its behavior, they are  $K_p$ ,  $K_i$ , and  $K_d$  respectively.  $K_p$  is called the proportional gain which controls how fast or how slow the system response reaches the steady state. The higher the value of  $K_p$  the faster the system reaches steady state but it also induces undesired oscillation and overshoot as the value increases.  $K_d$  is called the derivative gain which controls the amount of oscillation or overshoot of a system. The higher the value of  $K_d$  the more the oscillation or overshoot will be damped. However, if the value of  $K_d$  is set too high the steady-state value will be overdamped and the system will not be able to reach the user's defined set-point value.  $K_i$  is called the integral gain which controls the difference between the actual response and the set-point in the long run (steady-state error). We normally fine-tune this parameter at last especially for a position compensator since this parameter is usually not as dominant as the former two in a position control application.  $K_i$  is typically used to minimize the steady-state error.

In this section, you will be using the **sisotool** from MATLAB to help you to determine the appropriate gain values for your PID controller from the Root Locus diagram such that the error between the actual system response and the set point is kept to a minimum and the settling time of the system is achieved at the shortest duration possible.

✓	STEP	INSTRUCTION
	1	Launch the MATLAB program.
	2	<p>From the top menu bar, select the <b>Simulink</b> button as shown in Figure 17.</p>  <p>Figure 17: Launching the Simulink environment</p>
	3	<p>The <b>Simulink Start Page</b> should now appear as shown in Figure 18. Now select <b>Blank Model</b> from the top to start building your model from scratch.</p>  <p>Figure 18: Simulink Start Page</p>

Now you should see a new window pops up as shown in Figure 19. Select the **Library Browser** from the top menu bar to access the predefined block diagrams from Simulink.

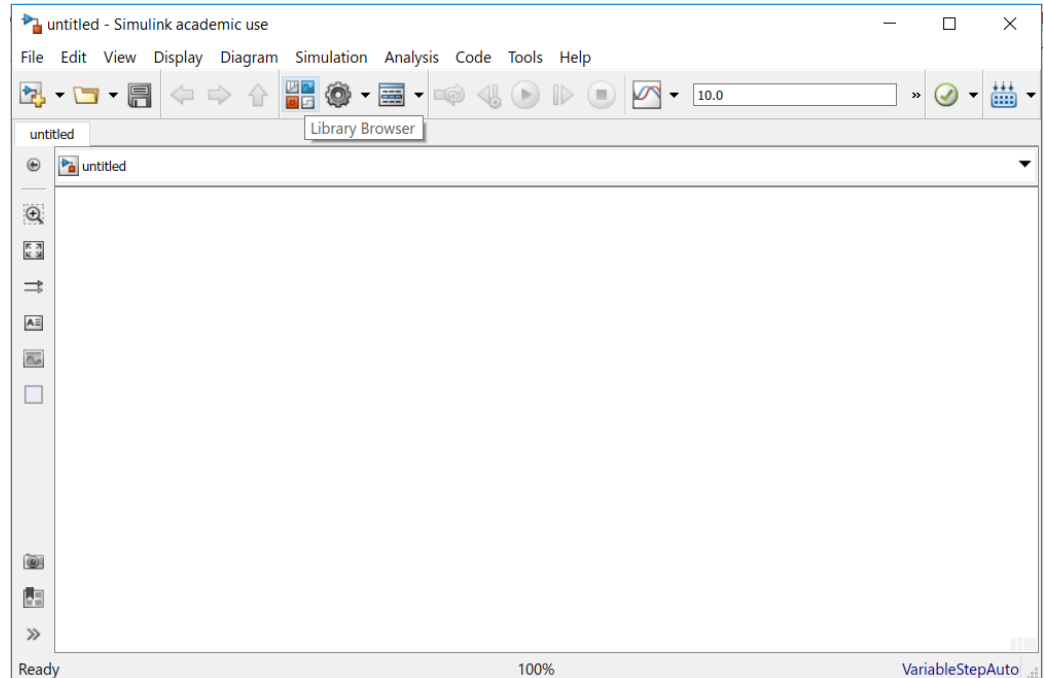


Figure 19: Launching Library Browser

Construct a closed-loop position control model of a DC motor in Simulink using according to Equations (12) and (13). Your Simulink model should look like Figure 20:

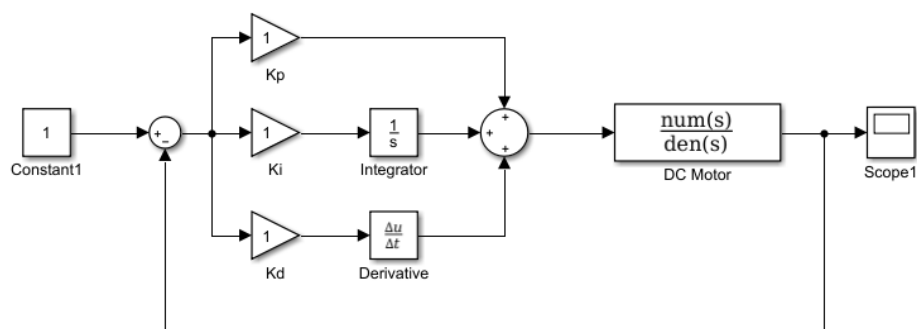
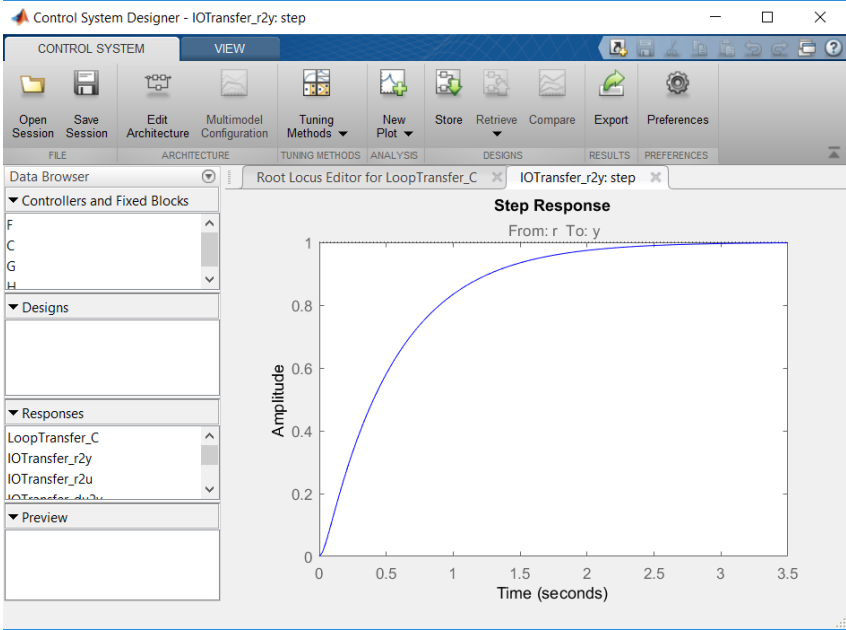


Figure 20: Closed-loop control system with a PID compensator

6		<p>To use the <b>sisotool</b> to determine your PID controller parameters, go back to the <b>MATLAB Command Window</b> and insert the transfer function of the DC Motor model with the following command and then press <b>Enter</b>:</p> $H = \text{tf}(\text{Eff\_G}*\text{Eff\_M}*K_g*K_t, [\text{Jeq}*R_m, \text{Beq}*R_m + \text{Eff\_G}*\text{Eff\_M}*K_g^2*K_t^2, 0])$
7		<p>Insert the following <b>sisotool</b> command and then press <b>Enter</b> to launch the Root Locus design feature of the tool:</p> <p><code>sisotool('rlocus', H)</code></p> <p>You should see the following window in Figure 21 being launched in the foreground:</p>  <p>The screenshot shows the 'Control System Designer' window. The 'VIEW' tab is selected, displaying a 'Step Response' plot. The plot shows 'Amplitude' on the y-axis (ranging from 0 to 1) versus 'Time (seconds)' on the x-axis (ranging from 0 to 3.5). The response curve starts at (0,0) and rises to approach an amplitude of 1. The left sidebar contains a 'Data Browser' with sections for 'Controllers and Fixed Blocks' (listing F, C, G, H), 'Designs', 'Responses' (listing LoopTransfer_C, IOTransfer_r2y, IOTransfer_r2u, IOTransfer_r2u), and 'Preview'.</p> <p>Figure 21: Control System Designer</p>
8		<p>Click on the <b>Root Locus Editor for Loop Transfer_C</b> tap as shown in Figure 22. Right click on the blank space inside the plot to bring up <b>Design Requirements</b> and select <b>New...</b> Here you will define your design criterion of your system (e.g. you can define a <b>%OS</b> of 4% and a settling time, <b>T<sub>s</sub></b> of 1.0 seconds as the starting point).</p>

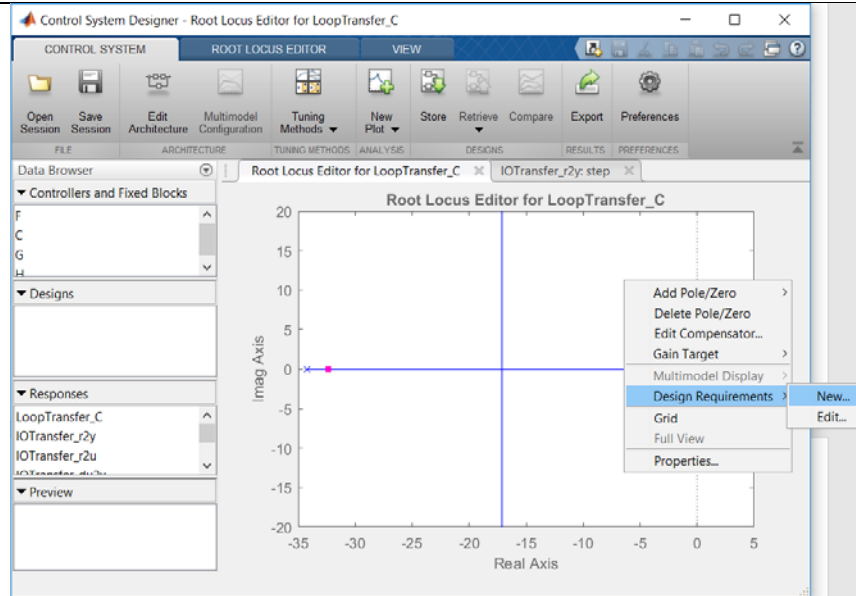


Figure 22: System design requirements

Now right click on the blank white space as seen in Figure 23 to bring up **Edit Compensator...** This is where you will configure the compensator type of your system.

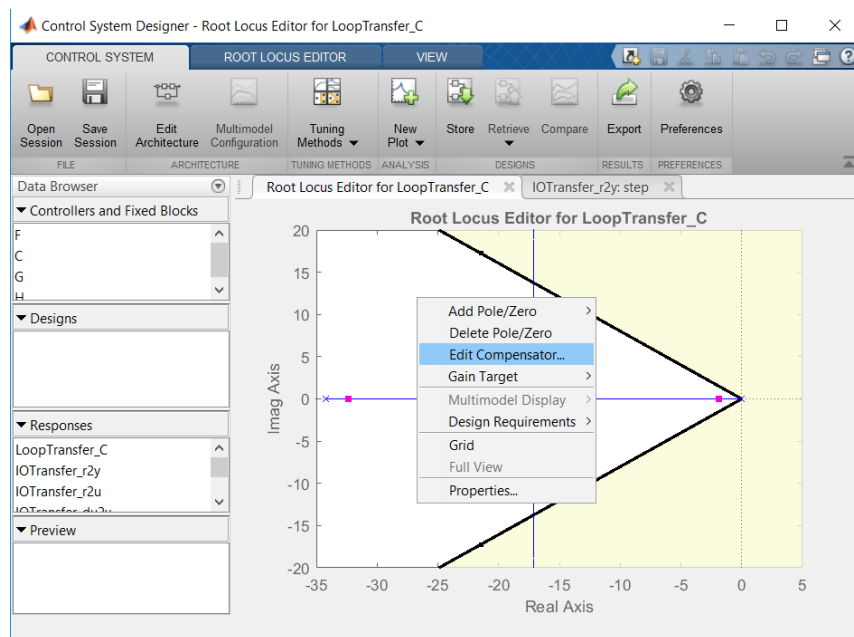


Figure 23: Define compensator type

Since the PID compensator consists of two zeroes (real or complex) and one pole at the origin as described in the following equation where  $z_{lag}$  and  $z_{lead}$  are the lag zero and lead zero respectively:

$$G(s) = \frac{K(s+z_{lag})(s+z_{lead})}{s} = \frac{K(z_{lag} \cdot z_{lead} + (z_{lag}+z_{lead})s+s^2)}{s} \tag{14}$$

In the **Compensator Editor** menu shown in Figure 24, let us add the lone pole at the origin (**integrator**) and a **Complex Zero** ( $z_{lag}$  and  $z_{lead}$ ) to construct the PID compensator. Right click in the **Dynamics** frame and add an **Integrator**. Repeat and add a **Complex Zero**.

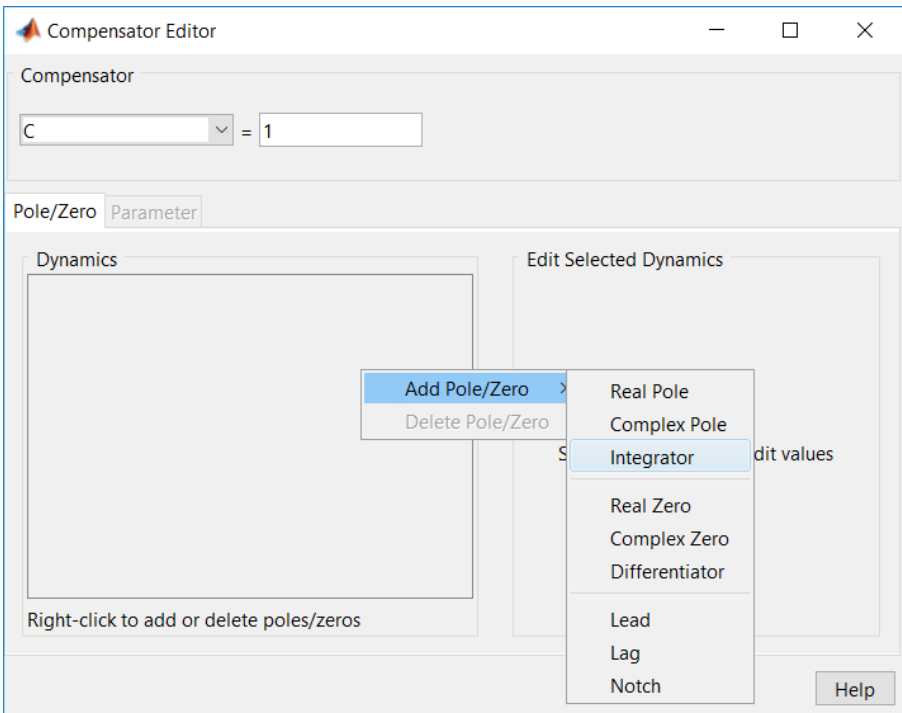


Figure 24: Define compensator dynamics

Your **Compensator Editor** should look like the one shown in Figure 25:



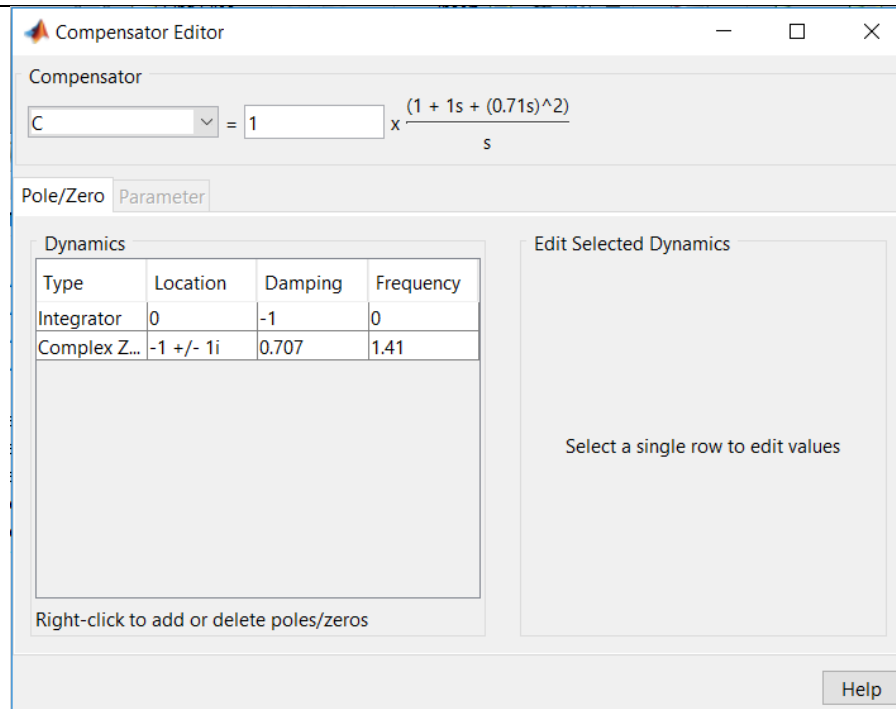


Figure 25: Defined dynamics for the compensator

Now adjust the Limits of your Root Locus plot as shown in Figure 26. You can do this on the go as you may need to adjust the limits to zoom in/out to help you to select the zeros and poles on the plot.

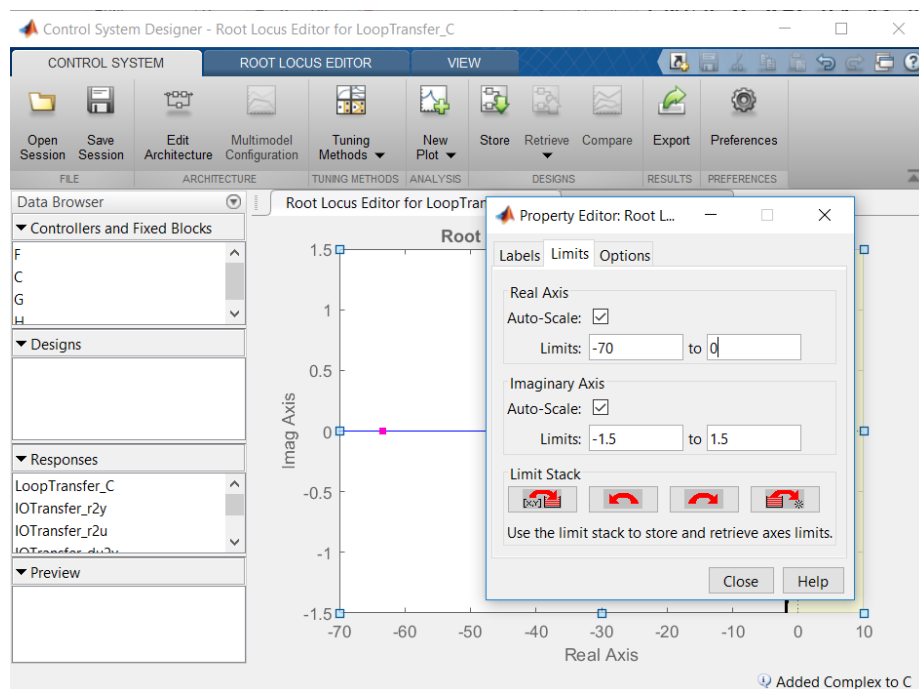
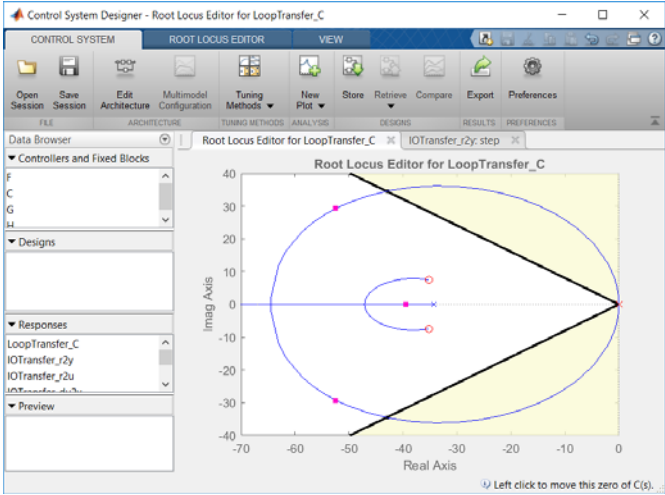
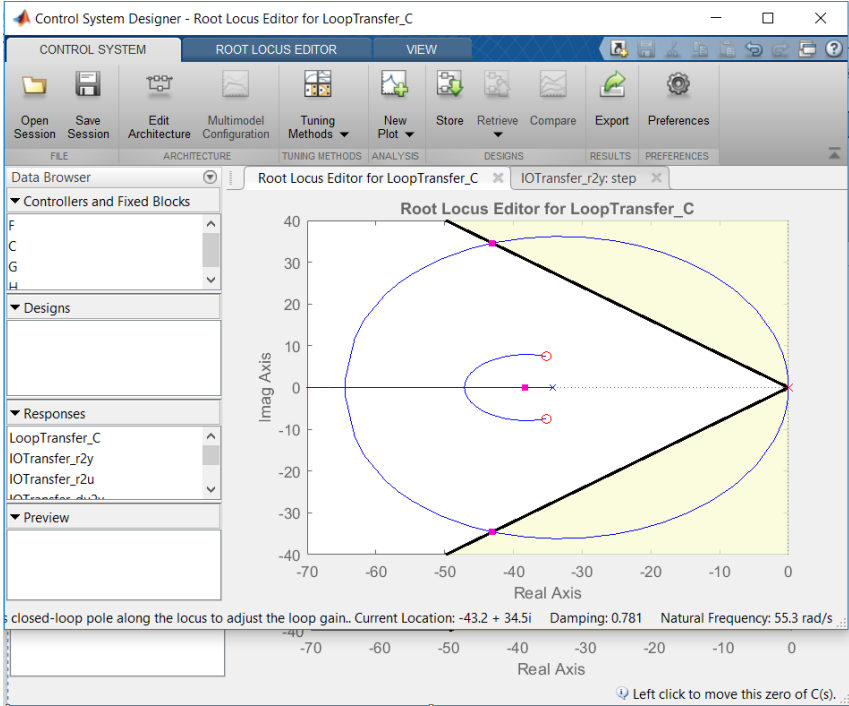
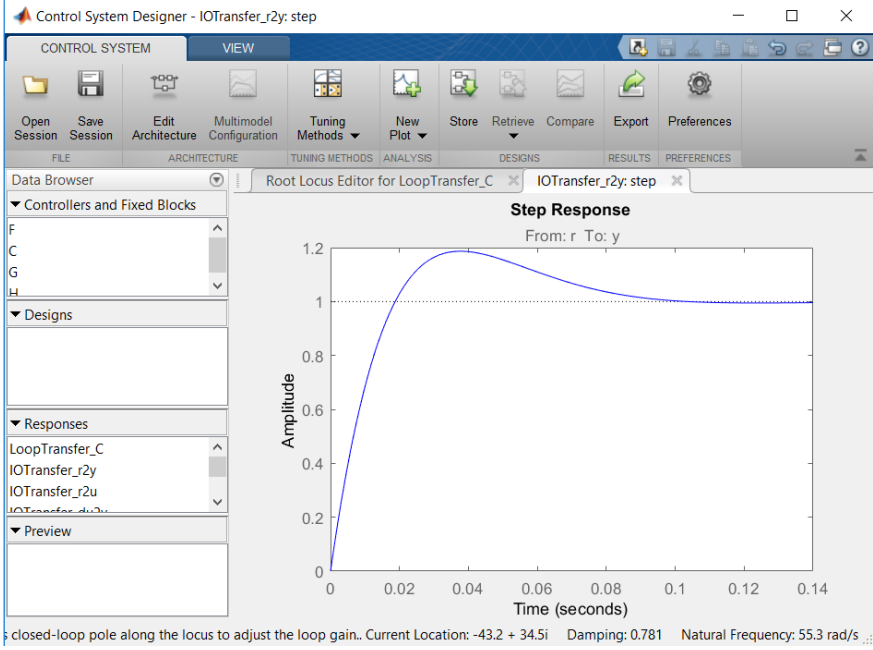


Figure 26: Set the Root Locus plot limits

12	<p>Drag the zeroes (pink circles) around until you see the paths of the root locus plot (blue lines) pass through the design intersection points (black dots on the black inclines) as seen in Figure 27.</p>  <p>Figure 27: Move the zeroes to meet with the design criterion</p>
13	<p>Once the blue lines pass through, grab the closed-loop poles (pink squares) and drag them to the intersection points as seen in Figure 28.</p>  <p>Figure 28: Move the closed-loop poles to meet with the design criterion</p>

	14	<p>Click the <b>IOTransfer_r2y: step</b> tap to view the step response of your system with the PID compensator as shown in Figure 29. You should see that your compensator has not completely achieved what you have defined earlier for the design criterion. This should really come as no surprise since the root locus design criteria applies only to systems that resemble closely to second order systems. In order to achieve our desired system response, we will have to further fine-tune the zeroes of our PID compensator.</p>  <p>Figure 29: Step response of the system with your PID compensator</p> <p>To determine the gain values of your PID compensator, we first need to expand Equation 14 and convert it to the parallel format that we are used to:</p> $G(s) = \frac{K(z_{lag} \cdot z_{lead} + (z_{lag} + z_{lead})s + s^2)}{s} = \frac{K_d \left( \frac{K_i}{K_d} + \frac{K_p}{K_d s + s^2} \right)}{s} \quad (15)$ <p>where <math>K_p</math>, <math>K_i</math>, and <math>K_d</math> are the gains of the PID compensator. Now go back to the <b>Compensator Editor</b> and find the terms correspond to <math>K_p</math>, <math>K_i</math>, and <math>K_d</math> as described in Equation (15) to determine their values.</p>
	15	<p>Now fine-tune the PID controller parameters (start with <math>K_p</math> first, then <math>K_d</math> and lastly <math>K_i</math>) until you can achieve a fast settling time with minimal oscillation and steady-state error. What are your steady-state error and</p>

		the values of your $K_p$ , $K_d$ and $K_i$ ? Discuss any challenges encountered in tuning the gains and explain any trade-offs you adopted.
--	--	---

## 2.3 Create a Hardware-in-the-loop position control of the DC motor unit

### Apparatus:

- 1 X PC station with Windows 7 or later version installed
- 1 X MATLAB R2015b
- 1 X Quanser SRV02 DC Motor Unit
- 1 X Quanser QUARC interfacing software

Up to this point here, we have a theoretical model of the PID compensator that can be used to control the DC motor. Now to validate the performance of your compensator we will need to conduct a hardware-in-the-loop demonstration using the Quanser SRV02 DC Motor Unit and its interfacing software QUARC.

✓	STEP	INSTRUCTION
	1	Launch the MATLAB program in <b>R2015b</b> version.
	2	Open and run the setup file <b>Setup_SRV02_Aero_position.m</b> to load all the required hardware configurations.
	3	Open the file <b>AER821_Lab1_Part2.slx</b> . The real plant (actual DC motor) has been constructed for you (You may need to configure the hardware type (e.g. Q2_USB, Q4, Q8) in the <b>HIL Initialize</b> block on the top right hand corner of the file to match with the controller card your PC is currently using). You now have to insert your own closed-loop DC motor model with the PID compensator from <b>Section 2.2</b> .
	4	Pass the same input signal to both the real plant and the model and run a test for at least 60 seconds to see if there is any discrepancy in the system response between the two. If there is, fine-tune your PID compensator further to see if you can improve the performance of the actual motor and record its gain values.

## 3 Lab Report Deliverables

When preparing your written report consider the following:

- You do not have to include materials from Section 1.
- You can quickly summarize the results of Section 2.1 and 2.2. You should concentrate on highlighting your control design criteria, the calculated gain values, and the predicted performance

- The bulk of this report should address the hardware-in-the-loop content of Section 2.3. Please comment on the differences you see between the theoretical predictions and the real system performance. You do not need to describe every minor tuning step, but try to illustrate the process showing increasing fidelity and performance.

## **4 References**

- 1) MATLAB HELP/Documentation Home/Simulink.
- 2) AER 509 Lab 3: Position Control
- 3) Quanser SRV02 DC Motor Unit Reference Manual