The McGraw-Hill Companies

PowerPoint to accompany

System Dynamics, Third Edition

William J. Palm III

Using Simscape™ for Modeling Electromechanical Systems:

Dynamics and Control of a Robot Arm

These slides are intended to be used with the author's text, *System Dynamics*, 3/e, published by McGraw-Hill©2014.

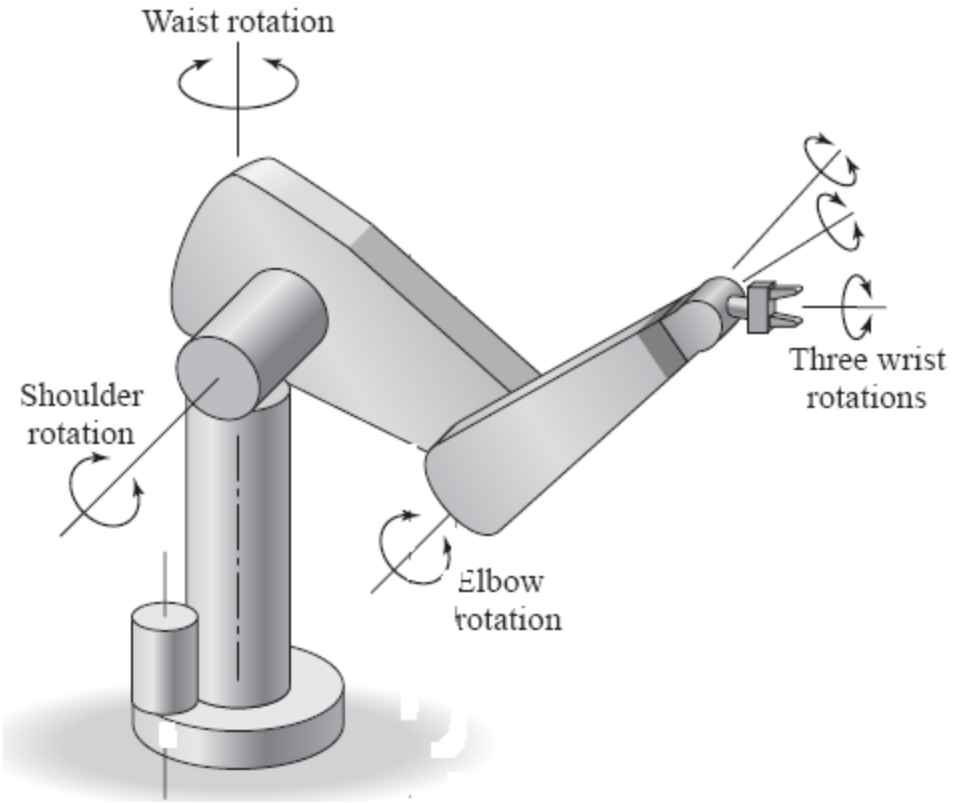
Acknowledgments

The author wishes to acknowledge the support of McGraw-Hill for hosting these slides, and The MathWorks, Inc., who supplied the software. Naomi Fernandes, Dr. Gerald Brusher, and Steve Miller of MathWorks provided much assistance. Dr. Brusher's contributions formed the basis for many of the Simscape models presented here.

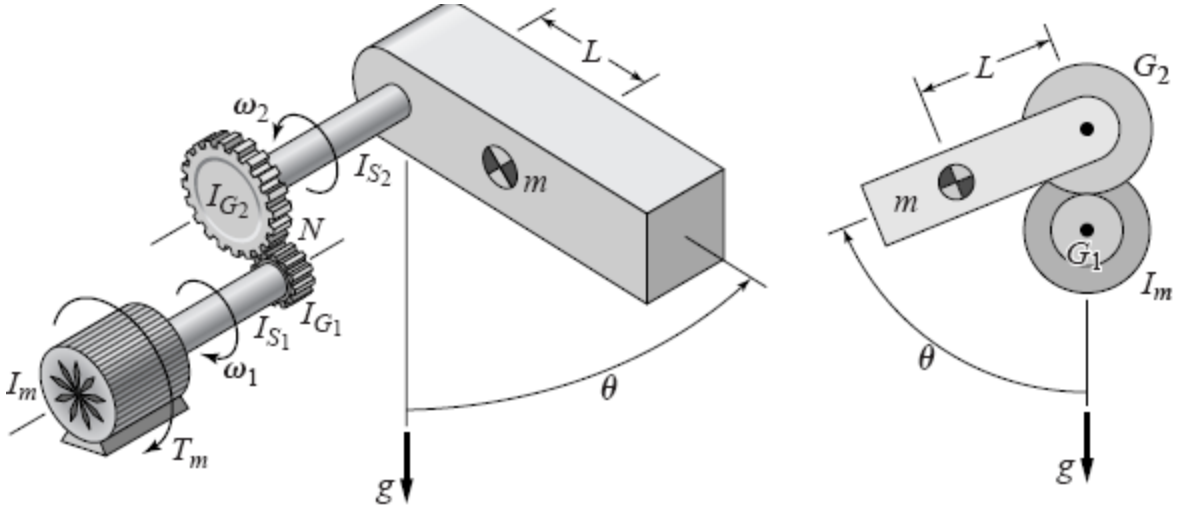
The equations and math symbols in these slides were created with the new equation editor in PowerPoint 2010, and thus material containing these elements will appear as graphics when viewed in an earlier version.

INTRODUCTION

Simscape™ extends the capabilities of Simulink® by providing tools for modeling and simulation of multi-domain physical systems, such as those with mechanical, hydraulic, and electrical components. In this presentation, we will show you how to utilize Simscape to construct models of electrical and mechanical systems. Shown below is a robot arm that has six joints. We will develop a model of one of those joints.



Our ultimate goal is to develop a model of a controller to ensure that the angle θ of the robot arm joint shown below tracks a prescribed profile. The joint is actuated by a dc motor that drives an arm of mass *m* through a gear pair. The mass center is located a distance *L* from the rotational axis of the joint. The weight *mg* exerts a torque $mgL\sin\theta$ that acts in the negative θ direction. The dynamics are treated in Example 3.5.5 of *System Dynamics*, 3/e.



Since a dc motor consists partly of a circuit having resistance and inductance, we will start with an electrical circuit example (Example 1), and then use it to build a model of such a motor (Example 2).

We will then build a model of the dynamics of a rotational mechanical system containing gears, such as the robot arm joint (Example 3).

Then we will add the gravity torque $mgL\sin\theta$ to complete the model of the arm's dynamics (Example 4).

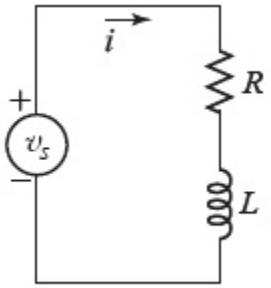
Then we will use the dc motor model to drive the mechanical system, thus obtaining the full model of an electrical-mechanical system that is widely used in engineering (Example 5).

Finally, we will design a position controller for the system (Example 6).

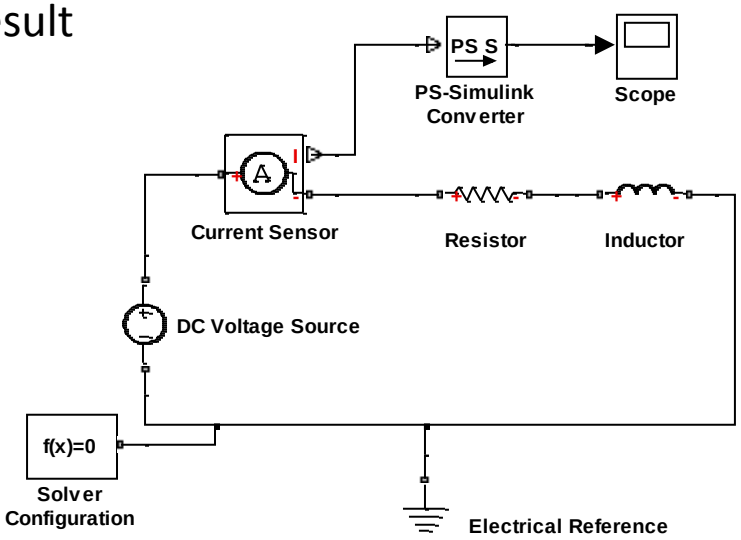
With each example, we will illustrate the model construction step by step. If possible, you should construct that model at each step as you follow the presentation.

Our example systems are simple enough such that we can obtain and solve the corresponding analytical models. We will use these solutions to check our simulation results. However, for more complicated systems this is not possible, and it is for these types of problems that simulation software is invaluable.

EXAMPLE 1: AN RL CIRCUIT:



This is the final result we will obtain:

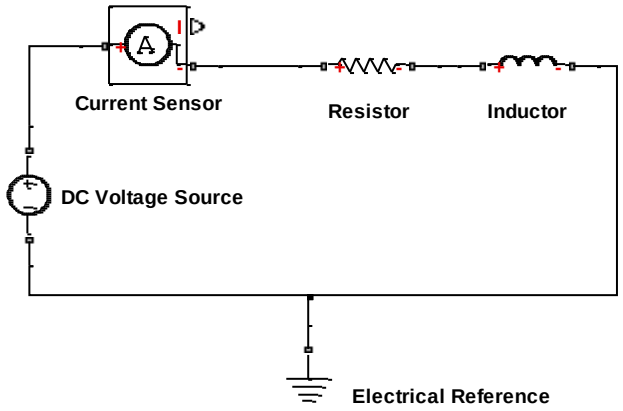


STEP 1: Select and place the Resistor, Inductor, and Electrical Reference elements from the Simscape>Foundation Library>Electrical>Electrical Elements library.

STEP 2: Select and place the DC Voltage Source element from the Simscape>Foundation Library>Electrical>Electrical Sources library.

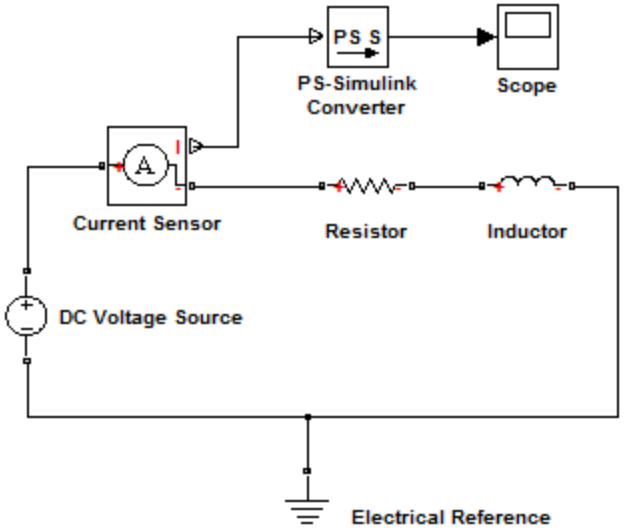
STEP 3: Select and place the Current Sensor element from the Simscape>Foundation Library>Electrical>Electrical Sensors library.

STEP 4: Connect the elements as shown below:

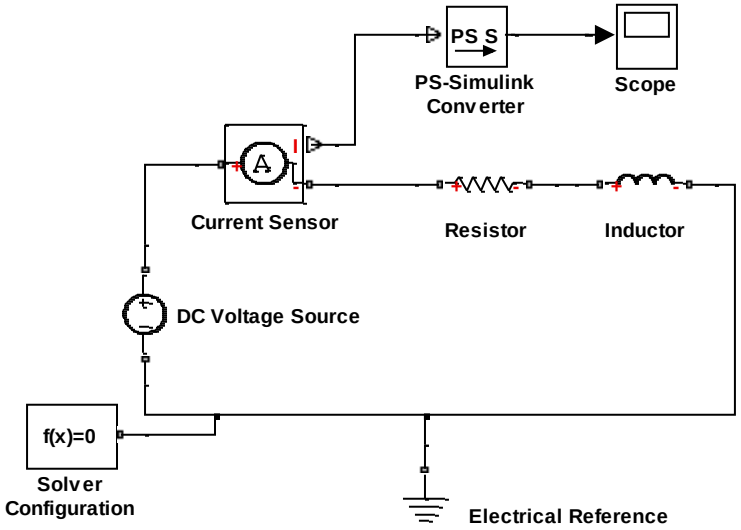


STEP 5: Select and place the PS-Simulink Converter from the Simscape>Utilities library. This block converts the physical signal (PS) to a unit-less Simulink output signal. Connect its input to the upper output port of the current sensor. This is the I port, where I stands for "current". The I port outputs the current as a physical signal which has units. The other ports ('+' and '-') are physical connections to the rest of the circuit.

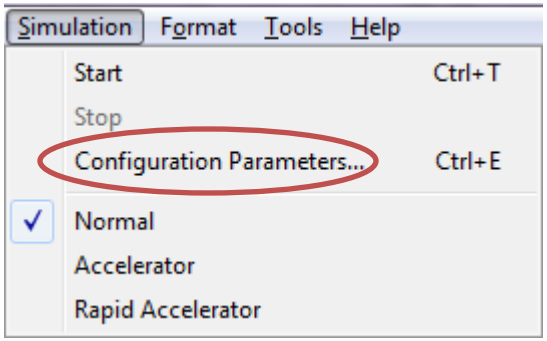
STEP 6: Select and place the Scope block from the Simulink>Sinks library. The diagram should now look like the one below.

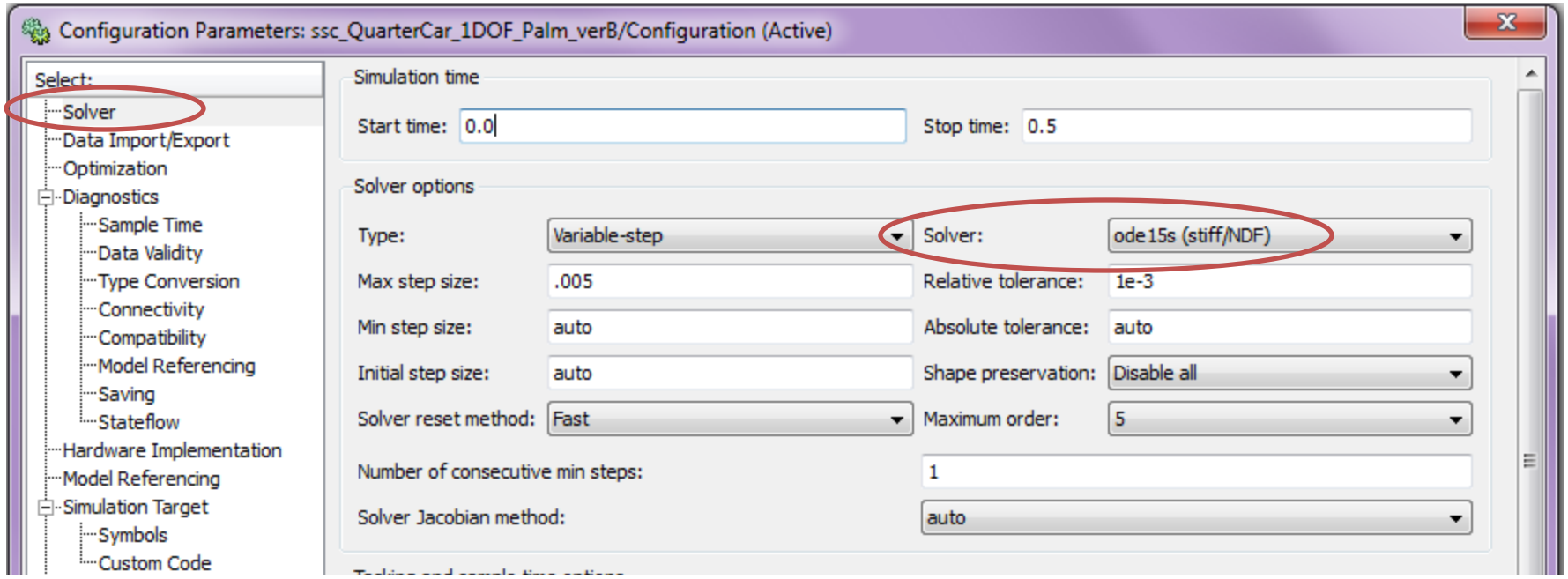


STEP 7: Select and place the Solver Configuration block from the Simscape>Utilities library. The Solver Configuration block defines the solver settings for this Simscape physical network. The Simulink solver for the entire model must be set separately. For this example, do not change any of the parameters in this block (all three boxes should be unchecked). Connect it as shown in the figure below. The model is now complete except for the parameter values.



A Note About Solvers: The default solver is ode 45. It is strongly recommended that you change the solver to a stiff solver (ode15s, ode23t, or ode14x). Do this by selecting "Configuration Parameters" from the Simulation menu, selecting the solver pane from the list on the left, and changing the "Solver" parameter to ode15s. Then click OK.



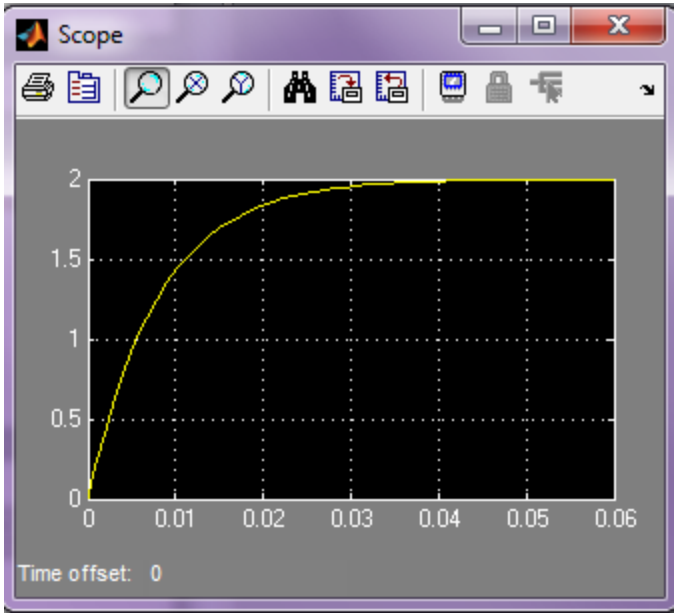


Now let's simulate the model. By double-clicking on the appropriate block, enter the following values for the resistance, inductance, and voltage: R = 5, $L = 0.004$, and v = 10 (the units are ohms, henrys, and volts, respectively, which are the default units used in these property boxes). The theoretical model for this circuit is

$$L\frac{di}{dt} + Ri = v$$

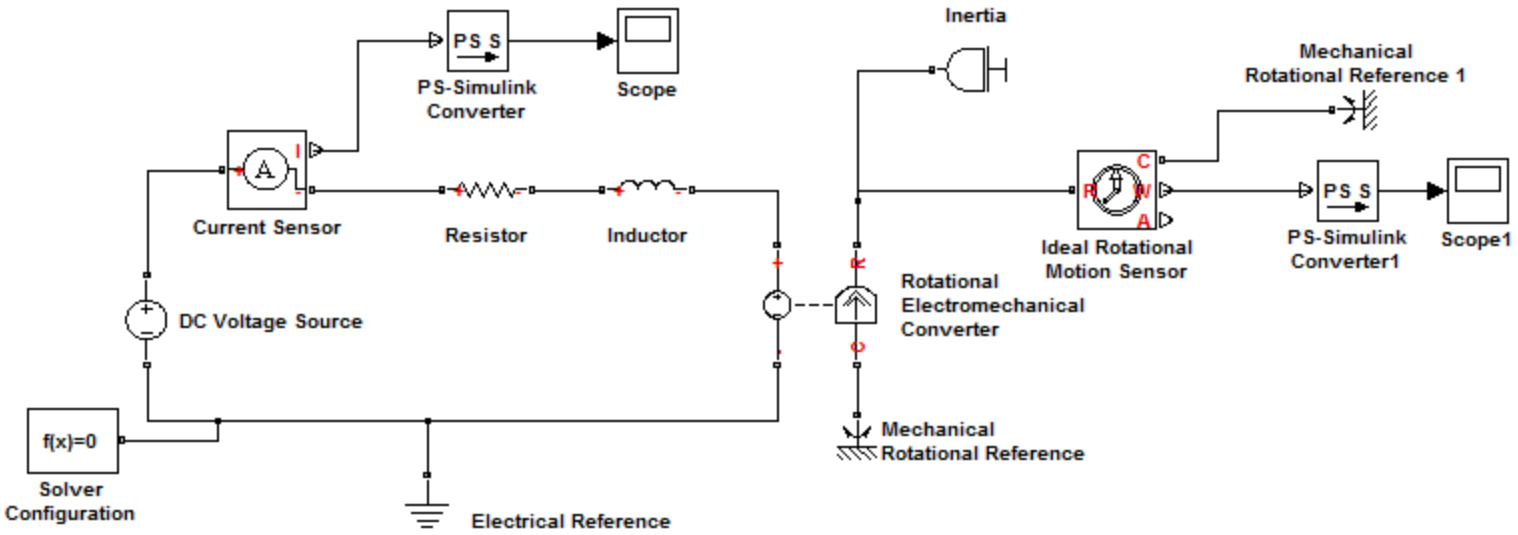
The time constant is therefore L/R. If v is constant, the steady state current will be v/R. For our values this gives L/R = 0.008 seconds and a steady state current of 2 amps, which is reached to within 2% after 4(0.008)=0.032 seconds.

Set the simulation time to 0.06 and run the model. You should see a plot like that shown in the scope to the right. The results agree with our analytical model.

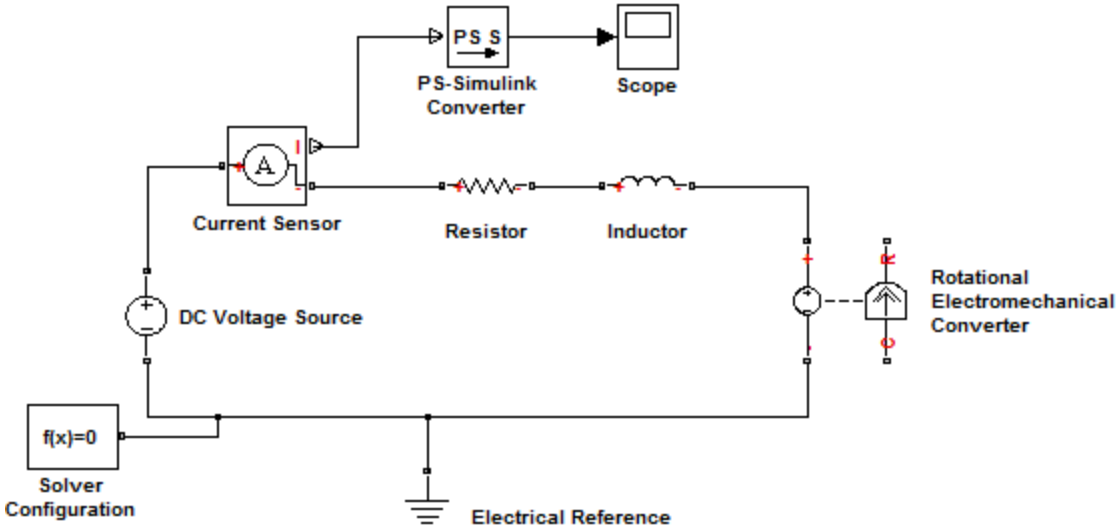


EXAMPLE 2: A DC MOTOR MODEL

We will now modify our RL circuit model to create a model of an armature-controlled dc motor. The final model will look like this:



To create the model, first delete the connection between the Inductor and the Electrical Reference. Then insert the Rotational Electromechanical Converter block from the Simscape>Foundation Library>Electrical>Electrical Elements library, and connect it with the Inductor and Electrical Reference as shown.



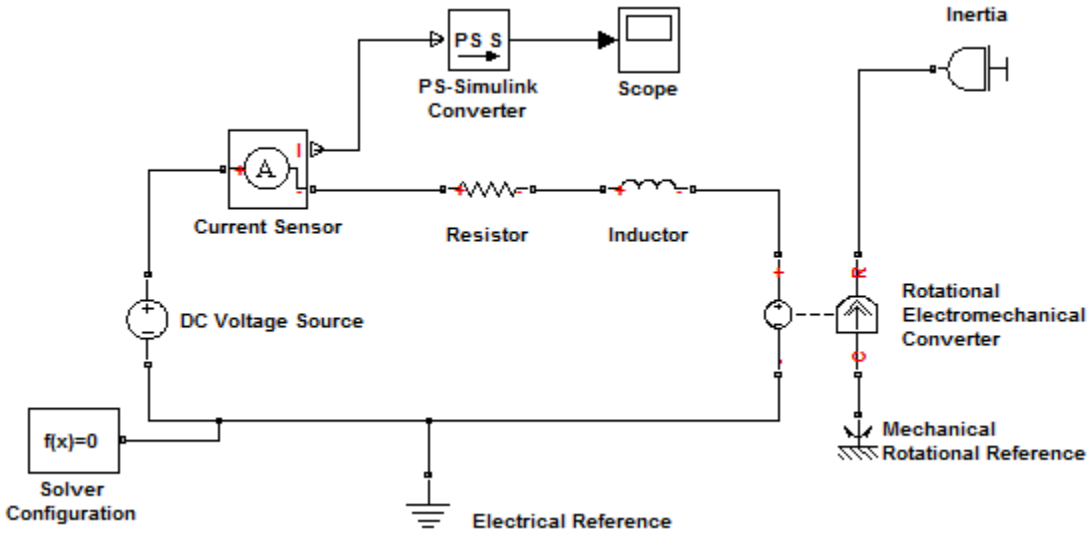
The Rotational Electromechanical Converter block provides an interface between the electrical and mechanical rotational domains. If the current and voltage through and across the electrical ports are *i* and *v*, and the torque and angular speed through and across the mechanical ports are *T* and ω, then $T = Ki$ and $v = K\omega$, where *K* is the *torque constant* with equivalent units of N·m/A or V/(rad/s). Since both the torque and back emf equations have the same value of K, this element represents a lossless electromechanical energy conversion. See Section 6.5 of *System Dynamics*, 3/e for a discussion of these and other dc motor principles.

If the current I from the electrical + to - ports is positive, then the resulting torque is positive acting from the mechanical C to R ports. This direction can be altered by using a negative value for *K*.

Open the Block Parameters dialog box of the converter and enter 0.1 N·m/A for *K*.

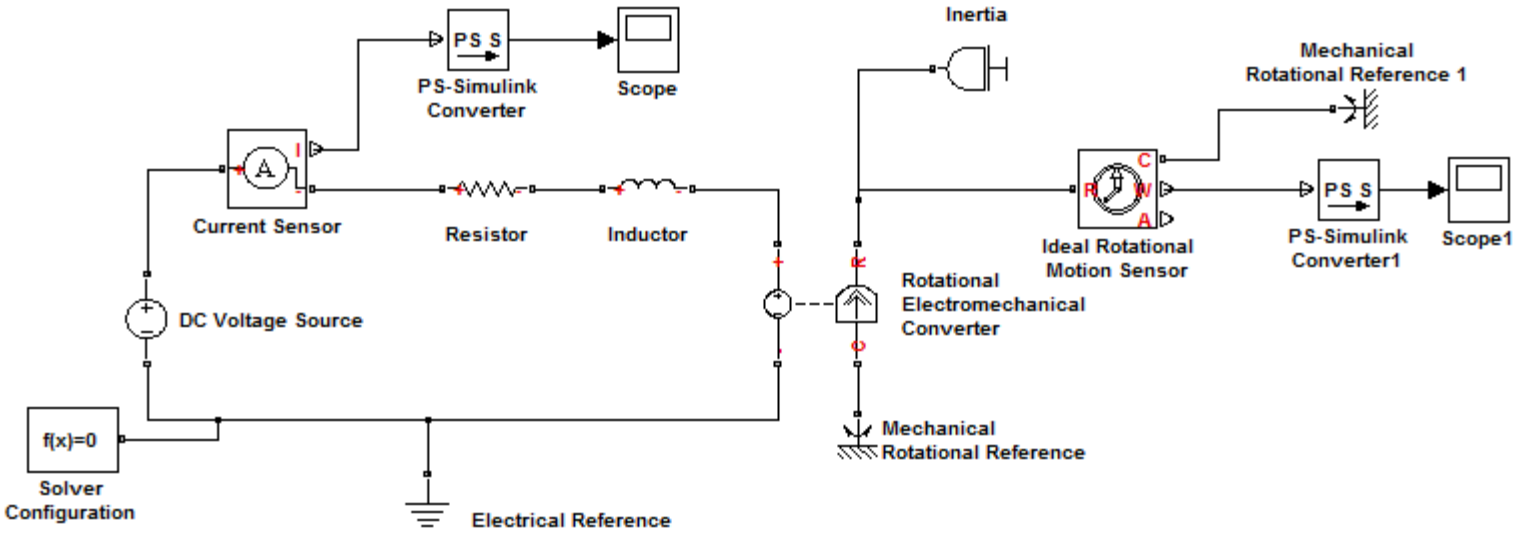
Since the Rotational Electromechanical Converter block does not contain the inertia of the motor's armature, we must add an inertia to the model. Select the Inertia block from the Simscape>Foundation Library>Mechanical>Rotational Elements library, and connect it to the R port of the electromechanical converter, as shown. Open its Block Parameters dialog box and enter 0.0005 $\mathrm{kg \cdot m^2}$ for the inertia value.

Next select and place the Mechanical Rotational Reference block from the Simscape>Foundation Library>Mechanical>Rotational Elements library. This block provides a reference for measuring the across variable *velocity*; thus it is analogous to an electrical ground, which provides a reference for the across variable *voltage*.



Next add the Ideal Rotational Motion Sensor block from the Simscape>Foundation Library>Mechanical>Mechanical Sensors library, and connect it into the network through its R port as shown. The sensor is ideal since it does not include effects that a real sensor would introduce into the system , such as inertia, friction, delays, energy consumption, and so on. Connections R and C are mechanical rotational power-conserving ports, while connections W and A are physical signal output ports for angular velocity and angular displacement, respectively.

Then insert another Rotational Reference, Scope, and Converter block as before, to obtain the final model shown.



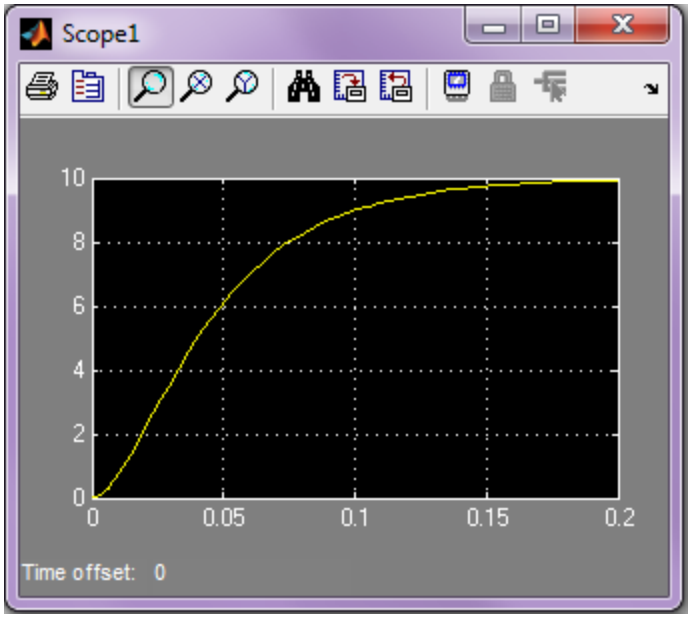
Testing the Motor Model: From Section 6.5 of *System Dynamics*, 3/e, the speed and current transfer functions for a motor having no mechanical damping are

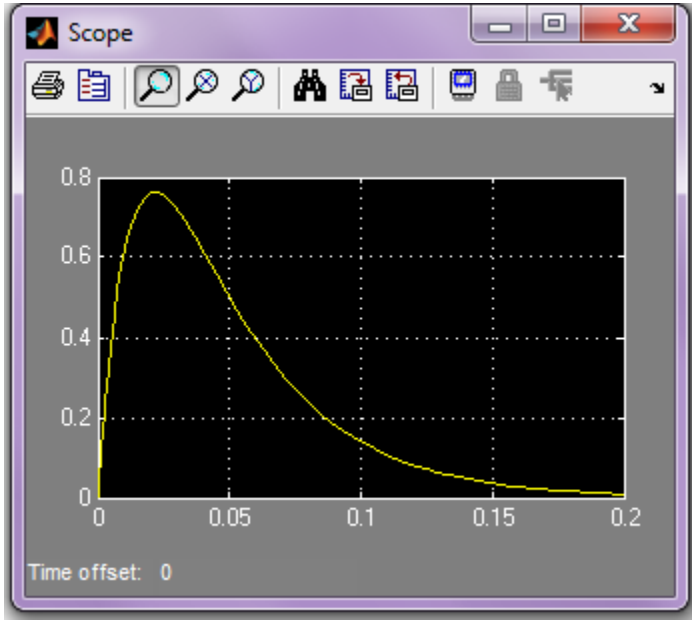
$$\frac{\Omega(s)}{V(s)} = \frac{K}{LIs^2 + RIs + K^2} \qquad \frac{I(s)}{V(s)} = \frac{Is}{LIs^2 + RIs + K^2}$$

Using $R = 1\ \Omega, L = 0.01\ \text{H}, K = 0.1\text{N·m/A}, \text{and}\ I = 0.0005\text{kg·m}^2$, the characteristic roots are $s = -72.36\ \text{and}\ -27.64$. So the time constants are τ =0.0138 and 0.0362 s. The steady-state response with a step voltage input should be reached in about four times the dominant time constant, or $4\tau = 4(0.0362) = 0.1447\ \text{s}$.

Using a voltage input of 1 V, the steady-state speed will be $1/K = 10\ \text{rad/s}$ and the steady-state current will be 0. The presence of numerator dynamics in the current transfer function suggests that the current might have a large overshoot.

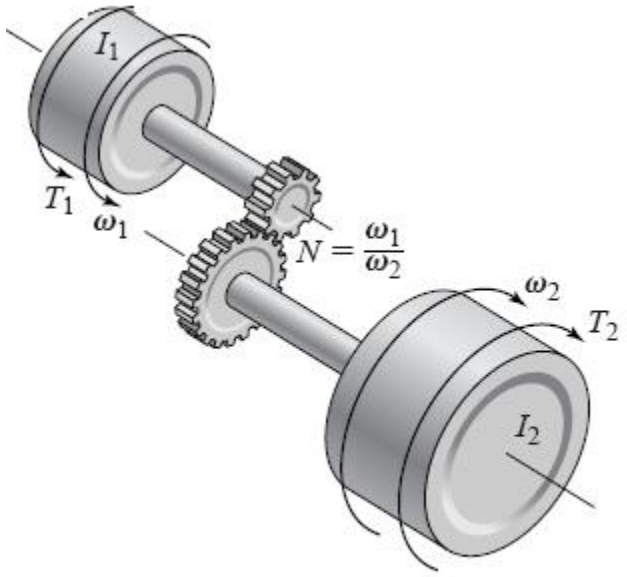
Set the run time to 0.2 s. The speed plot is shown below on the left, and the current plot is shown on the right. The plots confirm the results of the transfer function analysis.



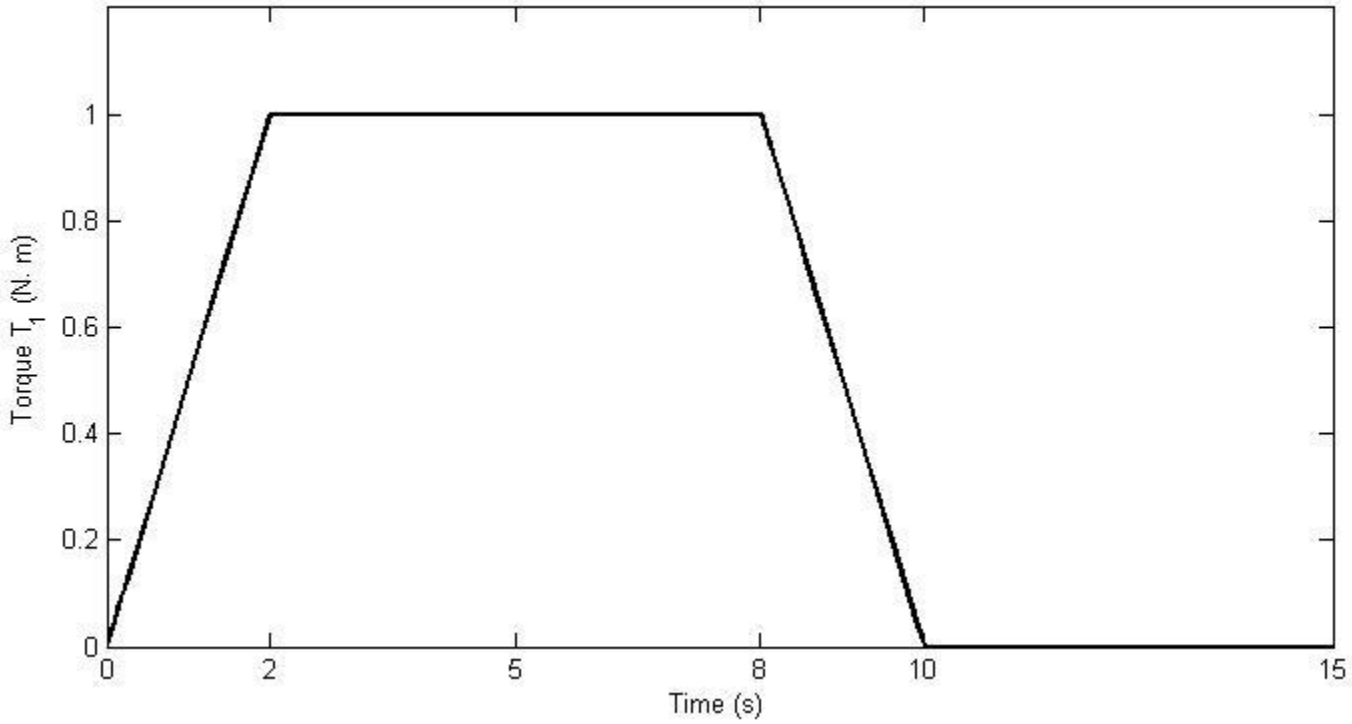


If we can get these results directly from the transfer functions, why use Simulink? Transfer function analysis may be impossible or at best very tedious to use for systems with nonlinearities such as torque limits and/or complicated input voltages such as trapezoidal functions. In such cases, numerical simulation is a powerful, practical approach. See Sections 6.8 and 6.9 in *System Dynamics*, 3/e for relevant examples.

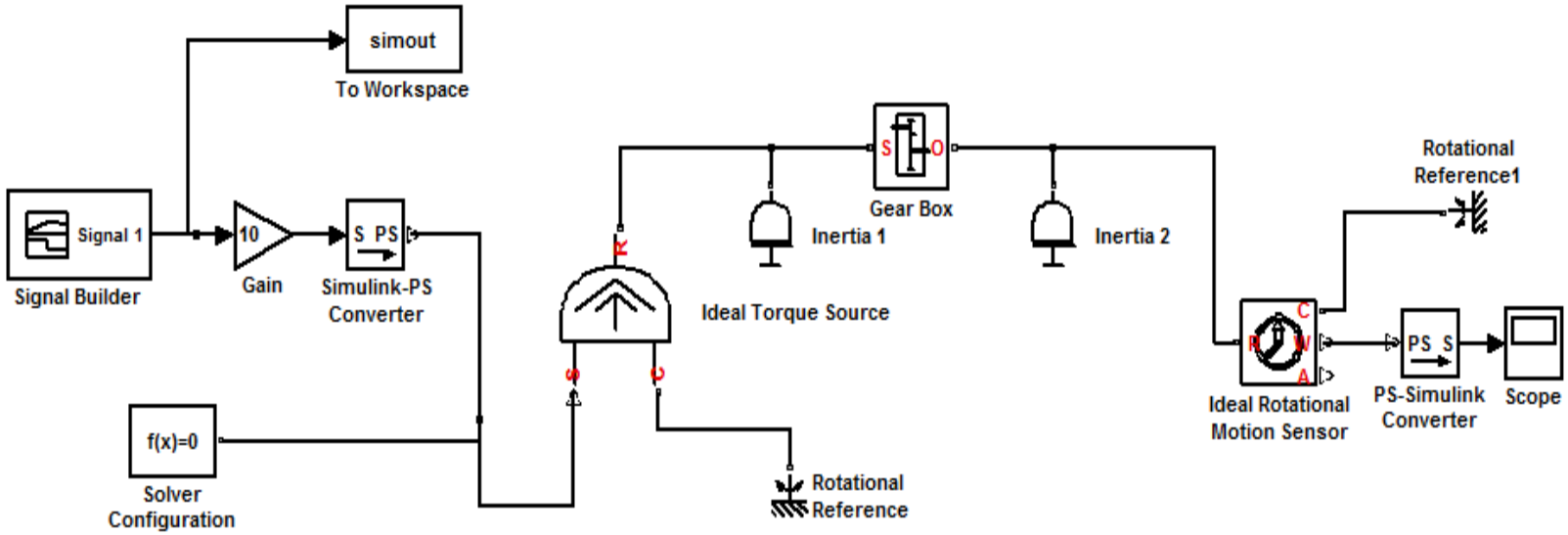
Example 3: A Geared System The figure below shows a representation of a rotational system containing a gear pair. The inertias I_1 and I_2 represent the elements on the driving side and the driven side, respectively. The gear ratio is *N*.



For now let us assume that the load torque T_2 is zero, and that the driving torque T_1 has the trapezoidal profile shown. We will use the Signal Builder block in the Simulink Sources library to create this function. We will use a Gain block to adjust the height of the trapezoid (the maximum torque).



Now let us construct the Simscape model. The only new Simscape elements are the Gear Box block from the Simscape>Foundation Library>Mechanical>Mechanisms library, the Ideal Torque Source block from the Simscape>Foundation Library>Mechanical>Mechanical Sources library, and the Simulink-PS Converter block from the Simscape>Utilities library. The model is shown below. Note that this time we have connected our scope to the *Speed* port (W) of the motion sensor.

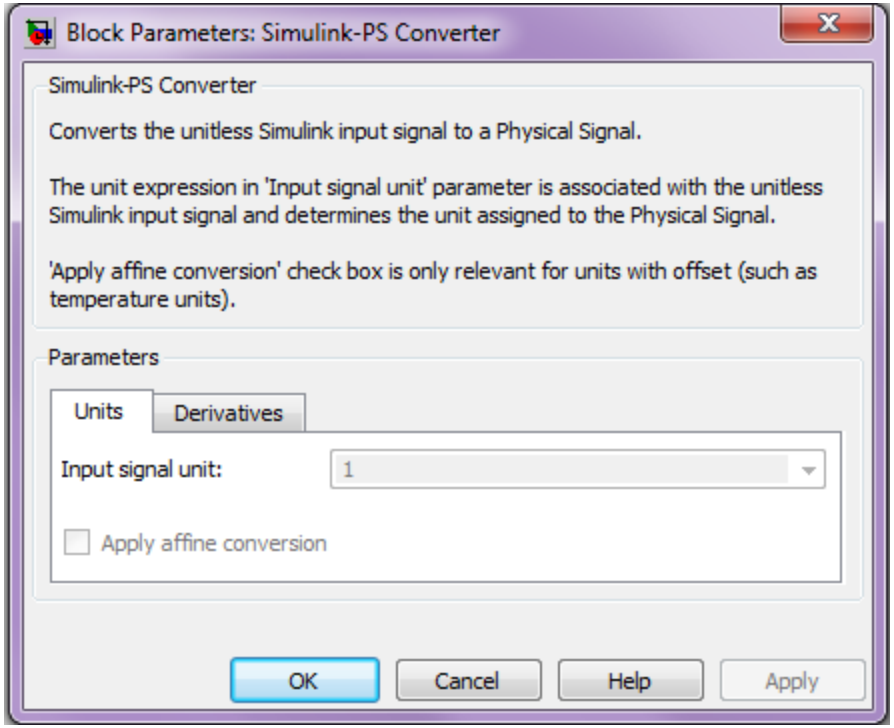


Set the following parameters in the appropriate blocks: Gain = 10, $I_1 = 0.085$ kg$\cdot$ m^2, $I_2 = 0.37$ kg$\cdot$ m^2, and the gear ratio to 2.

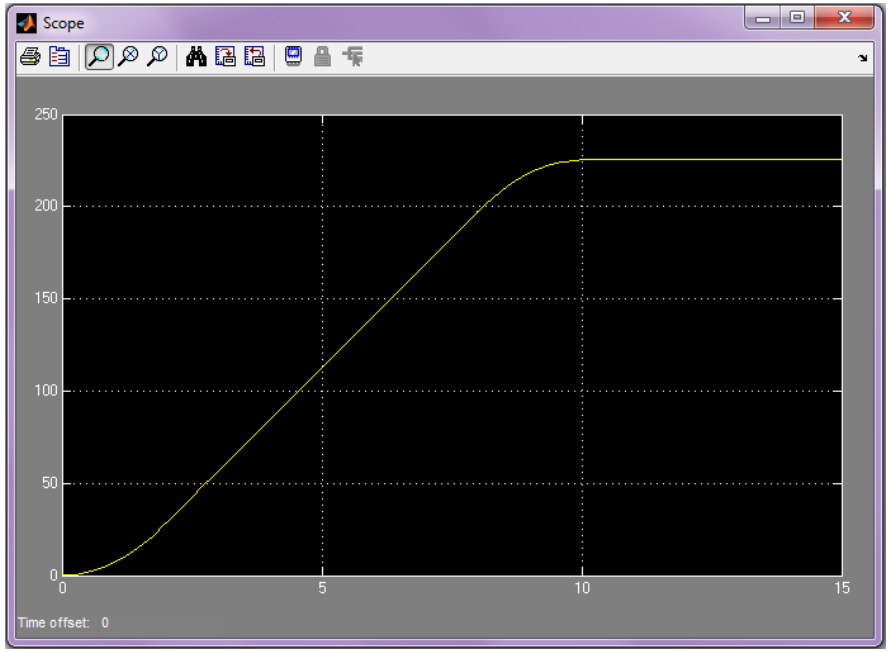
The Gear Box block contains only one parameter, the gear ratio. Thus, it represents a kinematic constraint only. In particular, the block does not model gear friction or the gear inertias. The former may be captured by a Rotational Friction element, while the latter must be included in the inertias I_1 and I_2 connected to the gear box.

The Ideal Torque Source block has no parameters. It represents a source capable of providing the torque specified at its physical-signal input port regardless of the angular velocity across its terminals. Since power is the product of torque and angular velocity, the Torque Source is ideal in the sense that it is sufficiently powerful to deliver the specified torque at any speed.

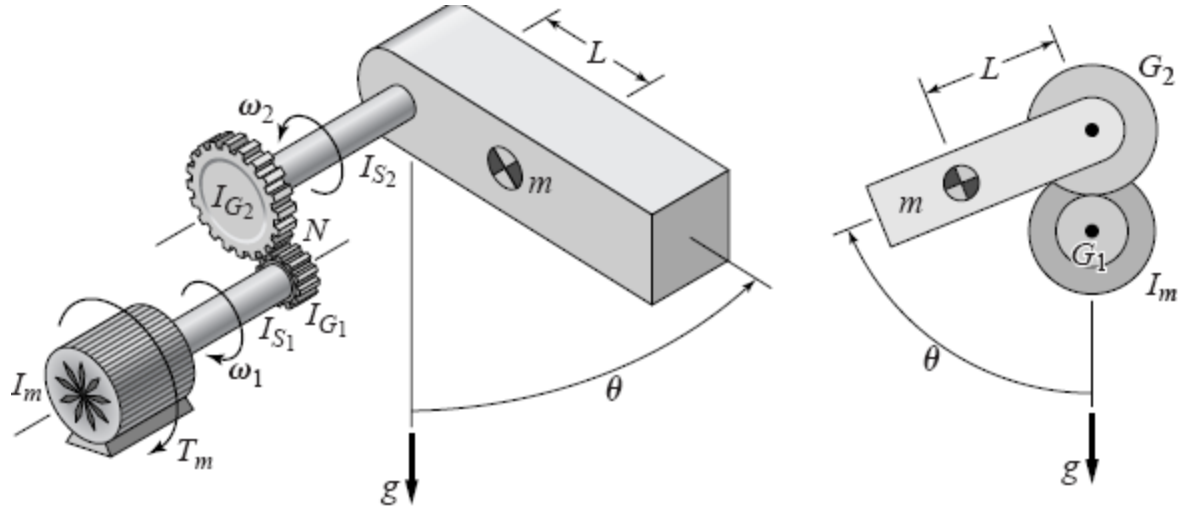
The Simulink-PS Converter block converts a unit-less Simulink signal to a *physical signal (PS).* Its Block Parameters dialog box is shown below.



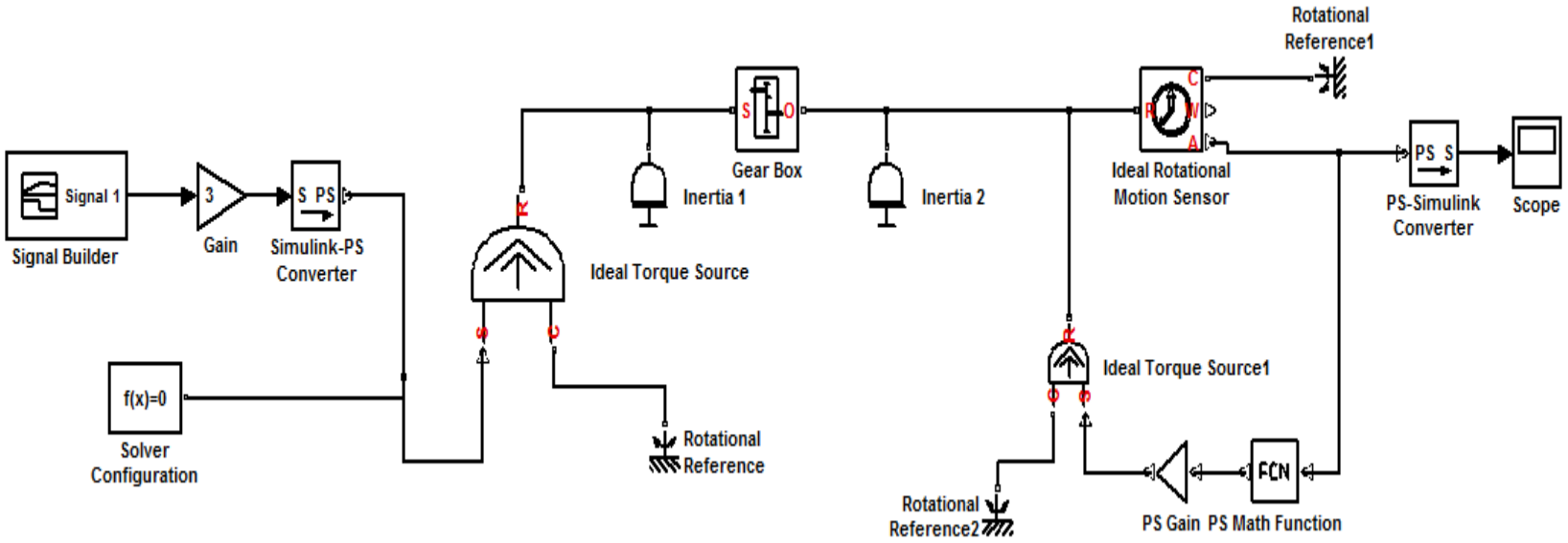
Set the Stop Time to 15 and run the model. You should see the following display in the Scope. The speed is measured in rad/s. The maximum speed is about 230 rad/s, which corresponds to approximately 2200 rpm. The speed levels off as it should, because the applied torque becomes zero after ten seconds.



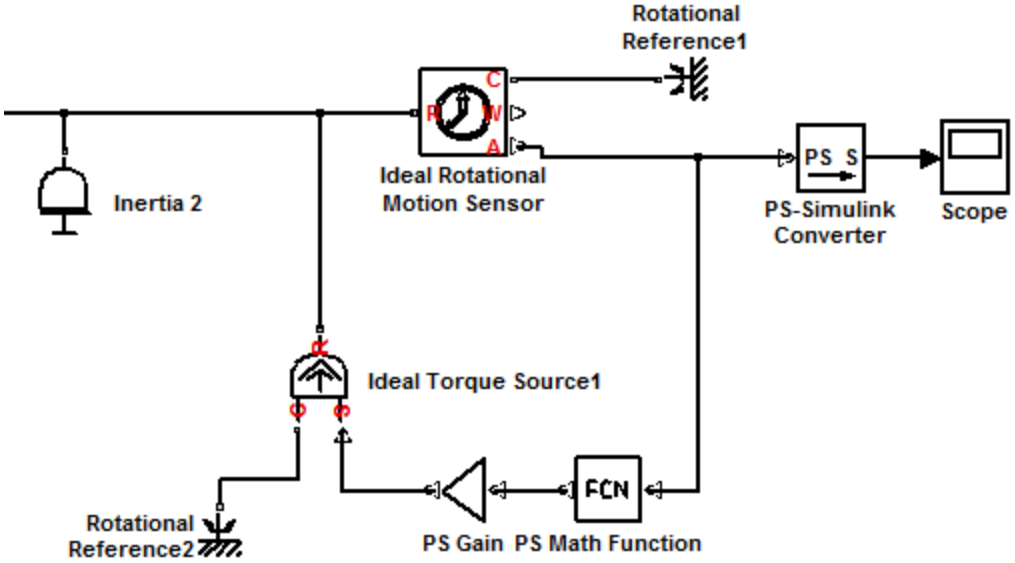
Example 4: Dynamics of a Robot Arm Joint Now we are ready to model the dynamics of the robot arm joint shown below. We will include the dc motor model later.



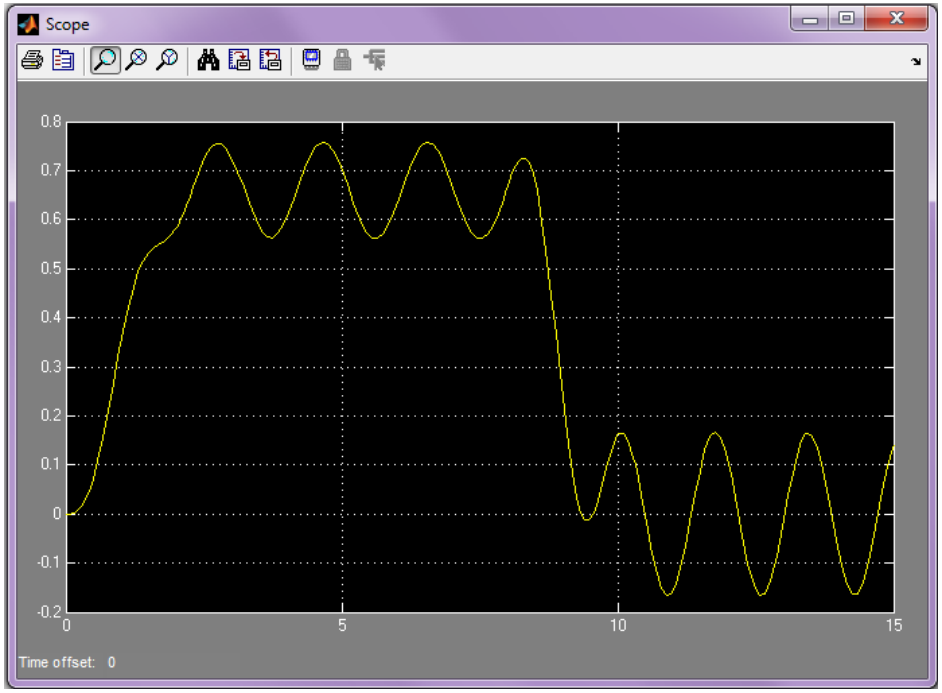
The completed model is shown below. We will use the following parameter values: Gain = 3, $I_1 = 0.085$ kg$\cdot$ m^2, $I_2 = 0.37$ kg$\cdot$ m^2, gear ratio = 2, m = 4 kg, L = 0.25 m, and g = 9.81 m/s^2. The Signal Builder produces the same trapezoidal profile used in Example 3.



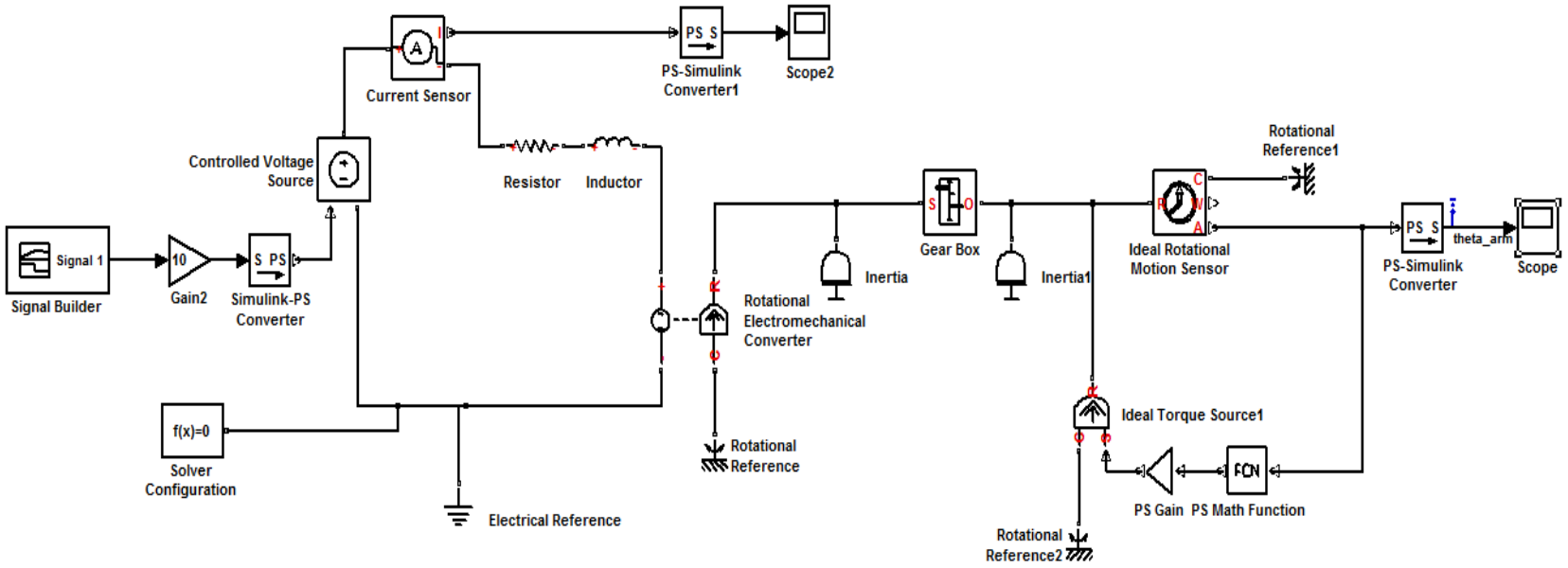
Let's examine the new blocks introduced to model the gravitational torque $-mgL\sin\theta$. The following figure shows that part of the model containing the new blocks. These are: the PS Gain block from the Simscape>Foundation Library>Physical Signals>Functions library, and the PS Math Function block from the same library. The PS Gain block multiplies the input physical signal by a constant called the Gain. For our model, enter –m*g*L in the Block Parameters dialog box. Then assign the values of m, g, and L in the MATLAB Command window. The PS Math Function block applies a mathematical function to the input u. Enter sin(u) in its Parameter dialog box. Note that we have now connected the scope to the *angle* port A.



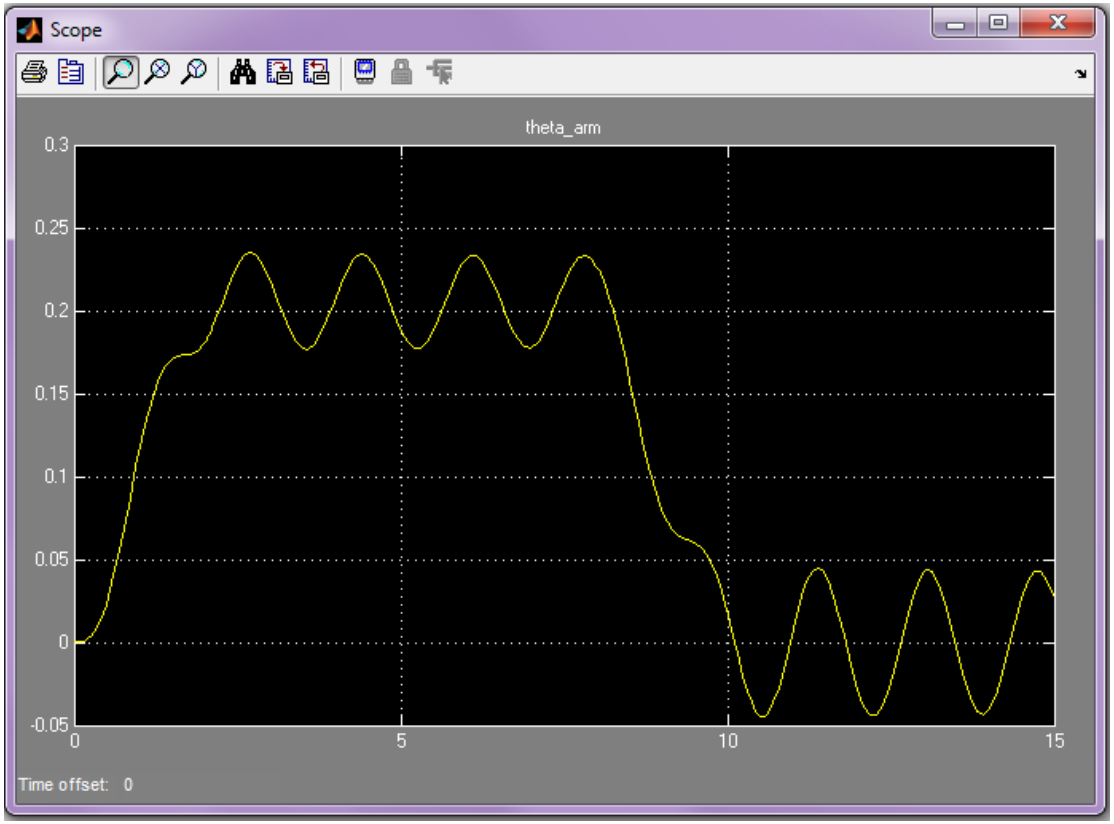
Run the model using a Stop Time of 15 s. You should see the following in the Scope. The arm angle is in radians. The constant amplitude oscillations are due to the fact that the system has no damping. When the applied torque goes to zero after 10 seconds, the arm oscillates like a pendulum about $\theta = 0$. For small angles, the differential equation model predicts a period of 1.69 s. This agrees exactly with the simulation results.



Example 5: Adding the Motor to the Arm Model The complete model is shown below. The only new block is the Controlled Voltage Source block from the Simscape>Foundation Library>Electrical>Electrical Sources library. This block represents an ideal voltage source that is powerful enough to maintain the specified voltage at its output regardless of the current passing through it. (Recall that electrical power is the product of voltage and current.)



Use $R = 0.5\ \Omega, L = 0.002$ H, $K = 0.05$N·m/A. Use Gain = 3, $I_1 = 0.0851$ kg· m^2, $I_2 = 0.37$ kg· m^2, gear ratio = 2, m = 4 kg, L = 0.25 m, and g = 9.81 $\mathrm{m/s}^2$. The inertia I_1 now includes the very small motor inertia 9×10^{-5} kg· m^2. The Signal Builder produces the same trapezoidal profile used in Example 3. Set the Stop Time to 15 and run the model. You should see the following in the Scope. The arm angle is in radians.

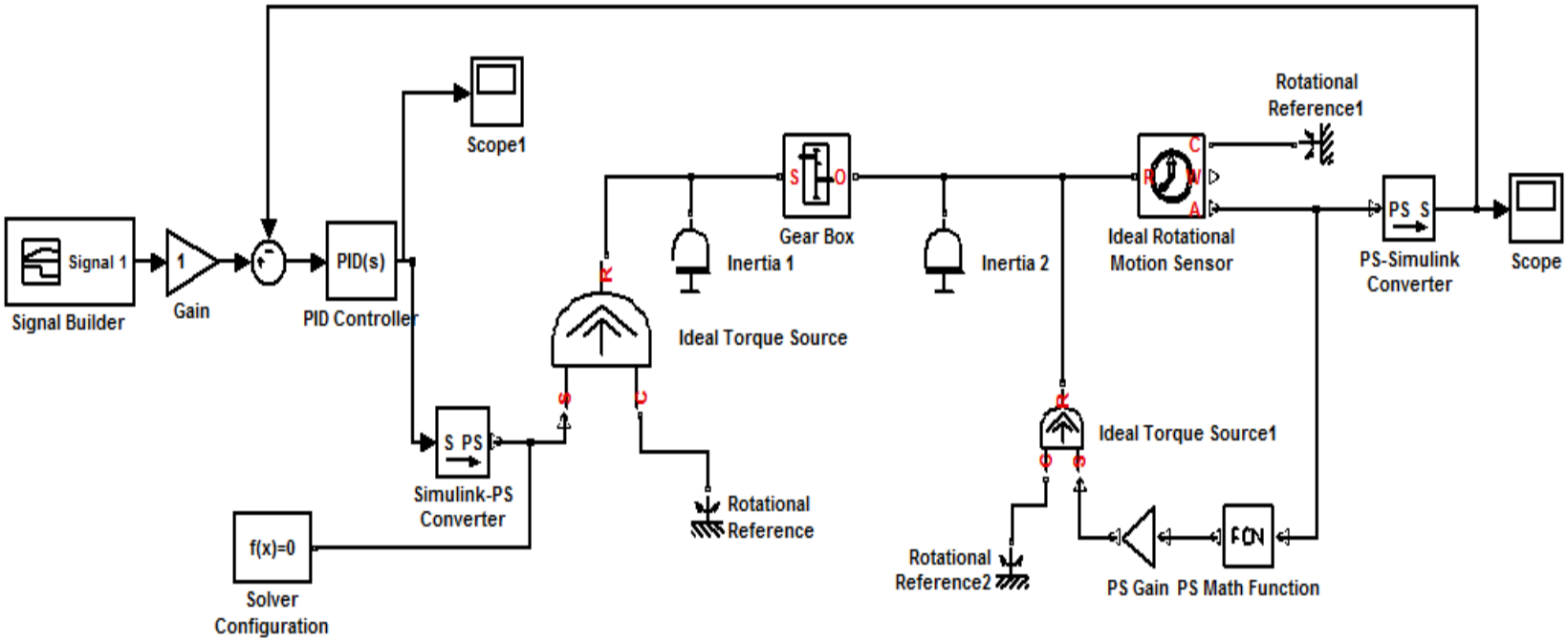


The plot is slightly different from Example 4. The arm now oscillates about a smaller angle because of the motor gain, but the period is about the same despite the dynamics of the motor. This is because the armature circuit time constant $L/R = 0.004$ s is very small compared to the period of the mechanical subsystem.

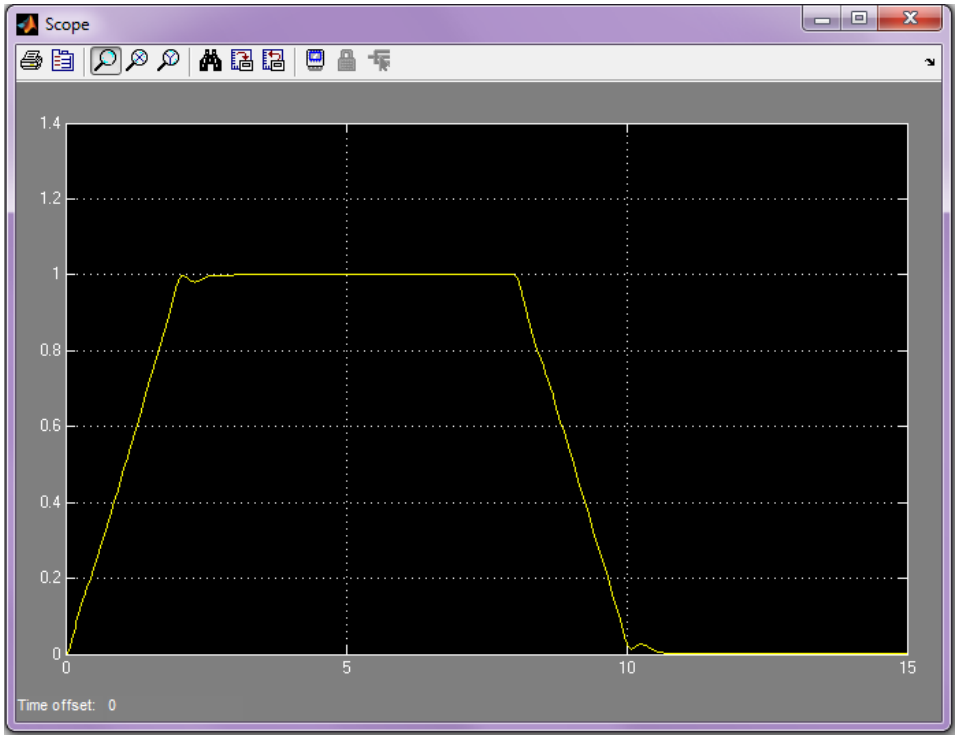
This suggests that we can ignore the electrical dynamics of the motor when developing a model of a control system for the arm. We will take this approach to keep the model complexity to a minimum.

Of course, if you need to know the motor current as part of the simulation, then you must retain the circuit model. But we will not do this here.

Example 6: Position Control of the Robot Arm: Neglecting the circuit dynamics enables us to construct the model shown below. Insert the PID Controller block from the Simulink>Continuous library and insert the Sum block and another Scope (to measure the control torque) as shown.

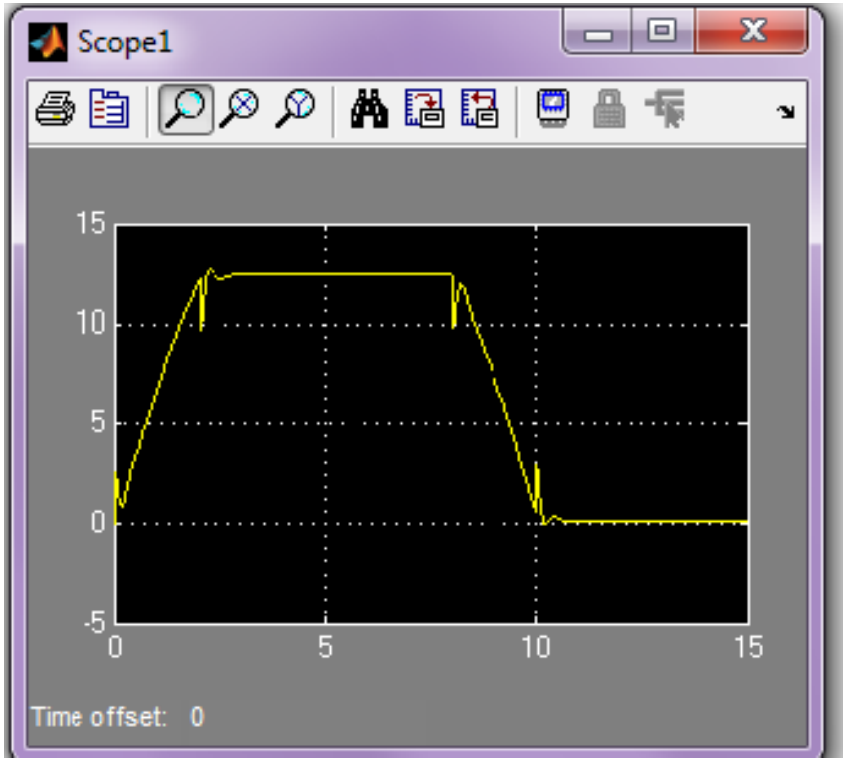


Open the Block Parameters dialog box of the PID Controller block and set the gains to the following values: Proportional = 50, Integral = 200, and Derivative = 5. Set the Gain to 1. Then run the model with a Stop Time of 15 s. You should see the following in the Scope for the arm angle. Clearly the angle follows the desired profile very closely.



Now open Scope1, which shows the torque required to achieve this performance. Controller engineers must always consider the actuator requirements before finalizing the design. Here the maximum torque required is about 13 N·m. In addition, the plot shows that the torque must change quickly, and this may not be possible for the chosen motor.

At this point the engineer may want to consider including the motor circuit model in order to determine the maximum required current. (Note the benefit of Model-Based Design: we are able to evaluate alternatives and make informed design decisions well in advance of hardware implementation and testing.) Assessing motor requirements is covered in detail in Section 6.6 of *System Dynamics*, 3/e.



We have not discussed the selection of the PID controller gains. That is the subject of Chapters 10 and 11 in *System Dynamics*, 3/e. There are many ways of doing this, including the MATLAB® `pidtool` and `sisotool` design tools, and Simulink Control Design.

This completes our modeling of the robot arm joint, which is an example of a common electromechanical system. We have used only those blocks available in the basic Simulink libraries, plus those in the Simscape Foundation library and the Simscape Utilities library.

MathWorks provides additional Simscape libraries for physical modeling. Two of these are SimElectronics and SimMechanics. These contain additional blocks that enable modeling of more complex multi-domain systems.

For example, SimElectronics includes a DC Motor block similar to the model we just developed, except that it also accounts for internal motor friction. In fact, SimElectronics offers more than 55 electronic and electromechanical components, including a variety of semiconductor, motor, drive, sensor, and actuator elements, as well as building blocks to implement your own custom subsystems.

SimMechanics contains blocks for modeling rigid body dynamics, for both planar and three-dimensional motion. Its sub-libraries include blocks for modeling constraints, kinematics, different types of joints, drivers, sensors, and actuators. In fact, SimMechanics would allow us to model the entire 3-dimensional mechanical robot shown previously on slide 3.