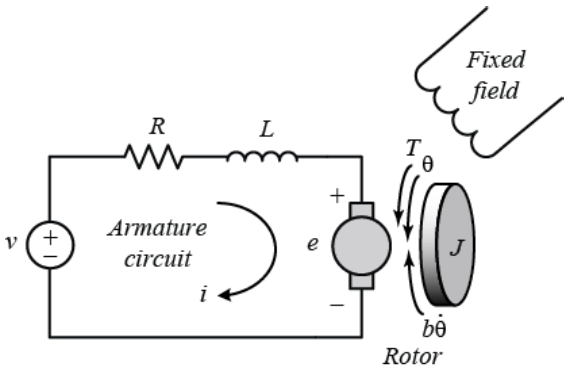
Control Tutorials for MATLAB and Simulink

Contents

Physical setup

In this section, we show how to build the DC Motor model using the physical modeling blocks of the Simscape extension to Simulink. The blocks in the Simscape library represent actual physical components; therefore, complex multi-domain models can be built without the need to build mathematical equations from physical principles as was done previously by applying Newton's laws and Kirchoff's laws to generate the model implemented in DC Motor Speed: Simulink Modeling.

The following figure shows a schematic of the DC Motor system we will be modeling in this page.



The system parameters are as follows:

(R)	Armature Resistance	1 ohms
(L)	Armature Inductance	0.5 H
(K)	Torque Constant/Back emf Constant	0.01 Nm/A or Vs/rad
(J)	Rotor Inertia	0.01 kgm^2
(b)	Rotor damping	0.1 Nms/rad

Open a new Simscape model by typing `ssc_new` in the MATLAB command window. A new model will open, shown below, with a few commonly used blocks already in the model. The PS-Simulink and Simulink-PS blocks define the boundary between Simulink input/output models where the blocks are evaluated sequentially and Simscape models where the equations are evaluated simultaneously.

Create the motor model

To create the motor model a number of blocks have to be added to the model.

Tips for adding blocks:

1. Use Quick Insert to add the blocks. Click in the diagram and type the name of the block (use the letters in **bold** below). A list of blocks will appear and you can select the block you want from the list.
2. After the block is entered, a prompt will appear for you to enter the parameter. Enter the variable names as shown below.
3. To rotate a block or flip blocks, right-click on the block and select the desired option from the **Rotate & Flip** menu.
4. To show the parameter below the block name, see Set Block Annotation Properties in the Simulink documentation

Add the following blocks:

* **DC Mot**or block
* **Cur**rent **Sen**sor block (be sure to use the one from the Foundation Library)
* **Con**trolled **Volt**age Source block (be sure to use the one from the Foundation Library)
* **Elec**trical **Ref**erence block (be sure to use the one from the Foundation Library)
* Ideal **Rot**ational **Mot**ion Sensor block
* Mechanical **Rot**ational **Ref**erence block
* **Step**

The DC Motor block models both the electrical and mechanical characteristics of the motor.

- Double-click on the DC Motor block, ensure **Model parameterization** is set to "By equivalent circuit parameters"
- Set **Armature resistance** to "1 Ohm"
- Set **Armature inductance** to "0.5 H"
- Set **Define back-emf or torque constant** to "Specify back-emf constant"
- Set **Back-emf constant** to "0.01 V/(rad/s)"
- On the **Mechanical tab**, set **Rotor inertia** to "0.01 kg*m^2" and **Rotor damping** to "0.1 N*m/(rad/s)"

Note that since the motor torque constant and the back emf constant are equal if the units are consistent, we only need to define one of the two.

We need to measure the position, speed, and current drawn by the motor.

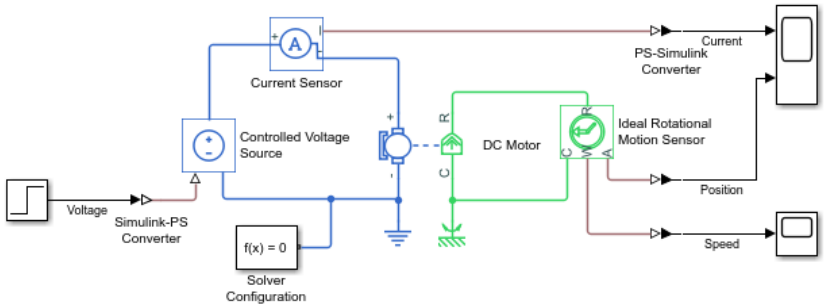
- Connect the R port of the Ideal Rotational Motion Sensor to the R port of the DC Motor
- Connect the C port of the DC Motor to the Mechanical Rotational Reference
- Connect the C port of the Ideal Rotational Motion Sensor to the Mechanical Rotational Reference
- Connect the - port of the Current Sensor to the + port of the DC Motor
- Connect the + port of the Current Sensor to the + port of the Controlled Voltage Source
- Connect the - port of the Controlled Voltage Source to the Electrical Reference
- Connect the - port of the DC Motor to the Electrical Reference

Next, we have to connect the input signals and measurements to the input and output blocks. This will let us control the motor using Simulink.

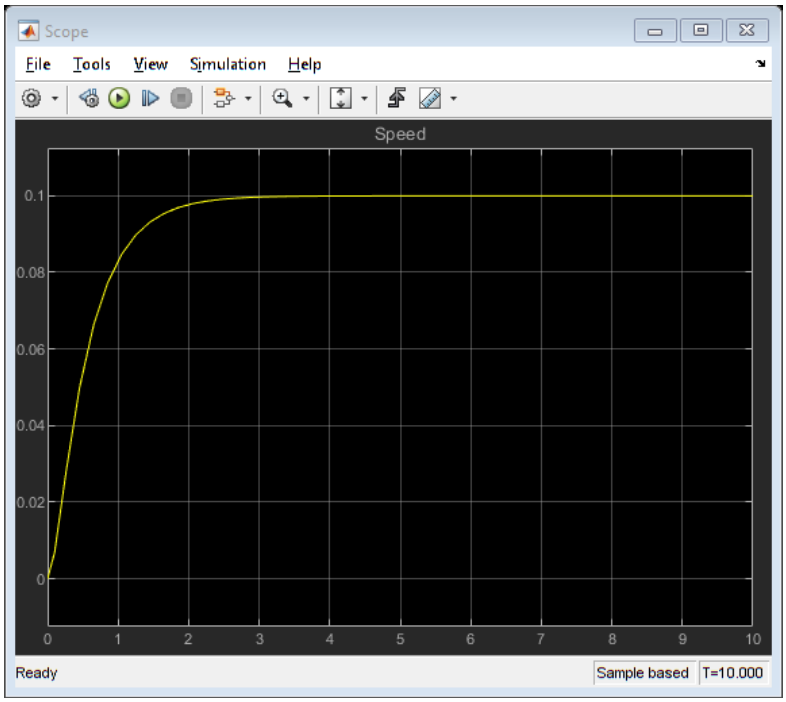
- Connect the Simulink-PS block to the Controlled Voltage Source input
- Double-click that signal connection and set the signal name to "Voltage"
- Double-click on the Simulink-PS block and set **Input signal units** to "V"
- Connect the Step block to the Simulink-PS block
- Double-click on the Step block and set **Step time** to "0"
- Connect the W port of the Ideal Rotational Motion Sensor block to the PS-Simulink block (already in the diagram, connected to a Scope)
- Double-click on the signal connected to the Scope and set the signal name to "Speed"
- Make two additional copies of the PS-Simulink block (you need 3 total)
- Double-click on the original PS-Simulink block and set the **Output signal units** to "rad/s"
- Connect Current Sensor to a PS-Simulink block, then double-click on that PS-Simulink block and set **Output signal units** to "A"
- Connect the A port of the Ideal Rotational Motion Sensor block to a PS-Simulink block, then double-click on that PS-Simulink block and set **Output signal units** to "rad"
- Copy and paste the Scope block
- Connect the PS-Simulink outputs for the current and speed signals to the Scope and name the signals "Current" and "Position"

The Solver Configuration block defines how the equations of a Simscape network are handled. Connect it to any electrical connection. We do not need to modify the parameters; we will use the defaults.

Delete any other unconnected items in the block diagram. The resulting model should appear as follows.



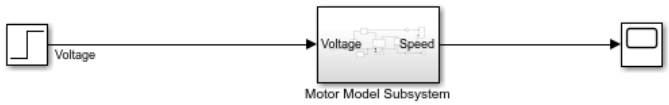
Running the simulation (**CTRL-T** or press the green arrow run button) will produce the following plot.



Create a subsystem

We will now place the motor into a subsystem.

- Click once in the diagram (but not on a block) and press **CTRL-A** to select all blocks
- Hold the **Shift** key and click on the Step block and the Scope for the Speed signal to unselect those blocks
- Press **CTRL-G** to create a subsystem
- Rename the subsystem "Motor Model Subsystem"



Set up the controller: lag compensation

To generate the closed-loop step response with the current model, we will add a lag compensator in series with the motor subsystem and will feed back the motor's speed for comparison to a desired reference. The design of the compensator is detailed in the DC Motor Speed: Root Locus Controller Design page.

Insert the following blocks:

- Subtract block
- Transfer Fcn block

Make the following adjustments to model the lag compensator:

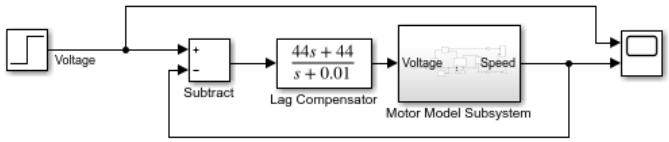
- Double-click on the Transfer Function block
- Set **Numerator coefficients** to "[44 44]"

- Set **Denominator coefficients** to "[1 0.01]"
- Rename this block "Lag Compensator"

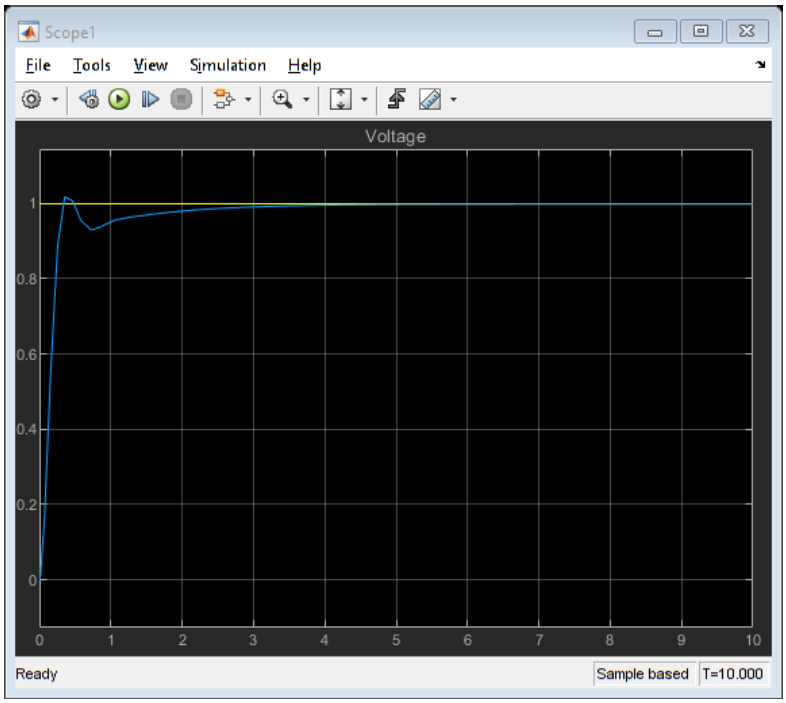
Make the following connections to close the loop:

- Delete the connection between the Step block and the input of the Motor Model Subsystem
- Connect the Step block to the + port of the Subtract block
- Connect the Subtract output to the input of the Lag Compensator
- Connect the Lag Compensator output to the Motor Model Subsystem
- Connect the Speed output of the Motor Model Subsystem to the - port of the Subtract block
- Connect the Step output to the Scope

Your completed model should now have the following form.



Then run the simulation (press **CTRL-T** or press the green arrow run button). Examining the generated plot, shown below, this step response matches the closed-loop performance observed in the DC Motor Speed: Root Locus Controller Design page where the lag compensator was originally designed. Furthermore, the simulation results achieved with this Simscape model also agree with the physics-based Simulink model implemented in the DC Motor Speed: Simulink Controller Design page.



Set up controller: lead compensation

The lag compensator we have designed meets all of the stated design requirements. Instead of a lag compensator, we could have also designed a lead compensator to meet the given requirements. More specifically, we could have designed a lead compensator to achieve a similar DC gain and phase margin to that achieved by the lag compensator, but with a larger gain crossover frequency. You can refer back to the DC Motor Speed: Frequency Domain Methods for Controller Design page for more details on the design of the lag compensator, but the fact that the DC gains and phase margins are similar indicate that the responses under lag and lead control would have similar amounts of error in steady state and similar amounts of overshoot. The difference in response would come in that the larger gain crossover frequency provided by the lead compensator would make the system response faster than with the lag compensator. We will specifically use the lead compensator that was used in the DC Motor Speed: Simulink Controller Design page.

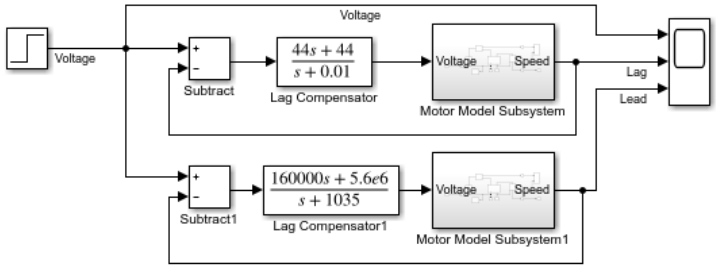
To see the precise effect of the lead compensator as compared to our lag compensator, make the following changes to the model

- Click and drag to select the Subtract, Lag Compensator, and Motor Model subsystem blocks, and all connections between them
- Right-click on one of the selected blocks and drag to copy these elements
- Connect the Step input to the + port of the new Subtract block
- Connect the Speed output of Motor Model Subsystem1 to the Scope

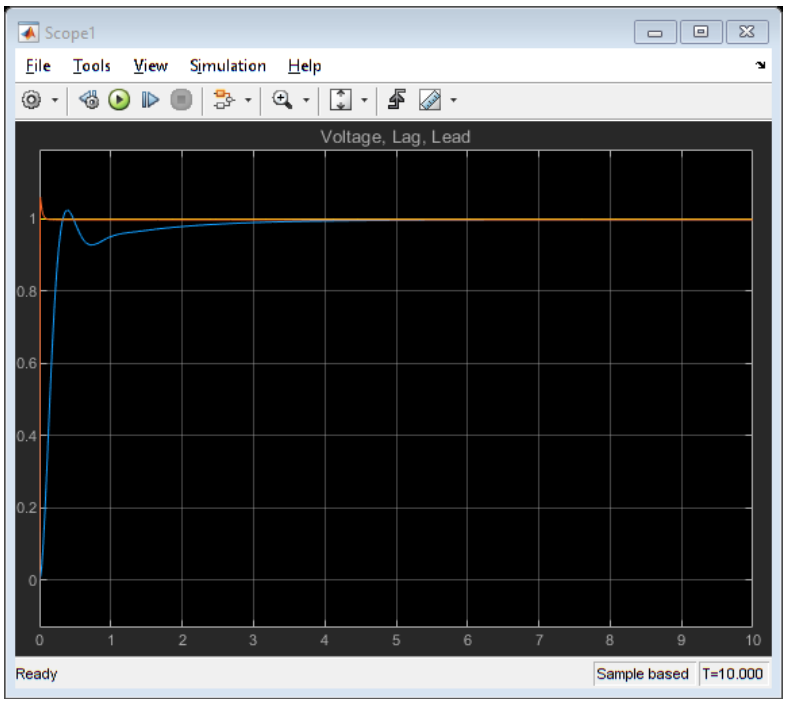
Next, modify the compensator as follows:

- Double-click on the copied Transfer Function block
- Set **Numerator coefficients** to "[160000 5.6e6]"
- Set **Denominator coefficients** to "[1 1035]"

Your model should appear as shown in the following figure:



Running the simulation produces the following plots. From inspection, you will see that both responses have a steady-state error that approaches zero. Comparing the two graphs, the response belonging to the lead compensated system has a much smaller settle time and slightly larger, but similar, overshoot as compared to the blue response produced by the lag compensated system.



Lead compensation vs. lag compensation

It is generally preferred that a system respond to a command quickly. Why then might we prefer to use the lag compensator even though it is slower than the lead compensator? The advantage of the lag compensator in this case is that by responding more slowly it requires less control effort than the lead compensator. Less control effort means that less power is consumed and that the various components can be sized smaller since they do not have to supply as much energy or withstand the higher voltages and current required of the lead compensator.

You can download the final Simscape model created here by right-clicking here and then selecting **Save link as ...**.