

Simscape™ Driveline™

User's Guide



MATLAB® & SIMULINK®

R2023a



How to Contact MathWorks



Latest news: www.mathworks.com
Sales and services: www.mathworks.com/sales_and_services
User community: www.mathworks.com/matlabcentral
Technical support: www.mathworks.com/support/contact_us



Phone: 508-647-7000



The MathWorks, Inc.
1 Apple Hill Drive
Natick, MA 01760-2098

Simscape™ Driveline™ User's Guide

© COPYRIGHT 2004–2023 by The MathWorks, Inc.

The software described in this document is furnished under a license agreement. The software may be used or copied only under the terms of the license agreement. No part of this manual may be photocopied or reproduced in any form without prior written consent from The MathWorks, Inc.

FEDERAL ACQUISITION: This provision applies to all acquisitions of the Program and Documentation by, for, or through the federal government of the United States. By accepting delivery of the Program or Documentation, the government hereby agrees that this software or documentation qualifies as commercial computer software or commercial computer software documentation as such terms are used or defined in FAR 12.212, DFARS Part 227.72, and DFARS 252.227-7014. Accordingly, the terms and conditions of this Agreement and only those rights specified in this Agreement, shall pertain to and govern the use, modification, reproduction, release, performance, display, and disclosure of the Program and Documentation by the federal government (or other entity acquiring for or through the federal government) and shall supersede any conflicting contractual terms or conditions. If this License fails to meet the government's needs or is inconsistent in any respect with federal procurement law, the government agrees to return the Program and Documentation, unused, to The MathWorks, Inc.

Trademarks

MATLAB and Simulink are registered trademarks of The MathWorks, Inc. See www.mathworks.com/trademarks for a list of additional trademarks. Other product or brand names may be trademarks or registered trademarks of their respective holders.

Patents

MathWorks products are protected by one or more U.S. patents. Please see www.mathworks.com/patents for more information.

Revision History

September 2010	Online only	New for Version 2.0 Beta (Release 2010b+)
April 2011	Online only	Revised for Version 2.0 (Release 2011a)
September 2011	Online only	Revised for Version 2.1 (Release 2011b)
March 2012	Online only	Revised for Version 2.2 (Release 2012a)
September 2012	Online only	Revised for Version 2.3 (Release 2012b)
March 2013	Online only	Revised for Version 2.4 (Release 2013a)
September 2013	Online only	Revised for Version 2.5 (Release 2013b)
March 2014	Online only	Revised for Version 2.6 (Release 2014a)
October 2014	Online only	Revised for Version 2.7 (Release 2014b)
March 2015	Online only	Revised for Version 2.8 (Release 2015a)
September 2015	Online only	Revised for Version 2.9 (Release 2015b)
March 2016	Online only	Revised for Version 2.10 (Release 2016a) (Renamed from <i>SimDriveline™ User's Guide</i>)
September 2016	Online only	Revised for Version 2.11 (Release 2016b)
March 2017	Online only	Revised for Version 2.12 (Release 2017a)
September 2017	Online only	Revised for Version 2.13 (Release 2017b)
March 2018	Online only	Revised for Version 2.14 (Release 2018a)
September 2018	Online only	Revised for Version 2.15 (Release 2018b)
March 2019	Online only	Revised for Version 2.16 (Release 2019a)
September 2019	Online only	Revised for Version 3.0 (Release 2019b)
March 2020	Online only	Revised for Version 3.1 (Release 2020a)
September 2020	Online only	Revised for Version 3.2 (Release 2020b)
March 2021	Online only	Revised for Version 3.3 (Release 2021a)
September 2021	Online only	Revised for Version 3.4 (Release 2021b)
March 2022	Online only	Revised for Version 3.5 (Release R2022a)
September 2022	Online only	Revised for Version 3.6 (Release R2022b)
March 2023	Online only	Revised for Version 3.7 (Release R2023a)

Getting Started

1	Introducing Simscape Driveline Software	
	Simscape Driveline Product Description	1-2
	Key Features	1-2
	Related Products	1-3
	Required Products	1-3
	Other Related Products	1-3
	Drivetrain Model	1-4
	What the Model Represents	1-4
	What the Model Illustrates	1-4
	Open CR-CR Transmission Example Model	1-5
	Run the Model	1-8
	Modify the Model	1-10
	Capabilities of Simscape Driveline Software	1-15
	What Simscape Driveline Software Contains	1-15
	Model Driveline Systems	1-16
	Model Inertias and Gears	1-16
	Model Dynamic Driveline Elements	1-17
	Model Custom Driveline Elements	1-17
	Actuate and Sense Motion	1-17
	Simulate and Analyze Motion	1-17

2	Modeling Driveline Systems	
	Start a New Simscape Driveline Model	2-2
	Build a Drivetrain Model	2-3

3

Complete Vehicle Model	3-2
Understanding the Global Structure of the Model	3-2
Model the Throttle/Brake Profile	3-3
Model the Engine	3-3
Model the Transmission	3-3
Couple the Engine to the Transmission	3-4
Model the Tires, Brakes, Wheels, and Road	3-4
Control the Clutches	3-6
Run the Model	3-7

Basic Motion, Torque, and Force Modeling

4

Couple Rotational Motion with Gears	4-2
Gear Coupling Rules	4-2
Couple Two Spinning Inertias with a Simple Gear	4-3
Modeling Two Spinning Inertias	4-3
Coupling Two Spinning Inertias with a Simple Gear	4-5
Torque-Actuating Two Coupled, Spinning Inertias	4-5
Sensing and Actuating Motion and Torque	4-7
Couple Two Spinning Inertias with a Variable Ratio Transmission	4-8
Couple Three Spinning Inertias with a Planetary Gear	4-10

Driveline Actuation

5

Best Practices for Modeling Torque-Force Actuation and Motion	
Actuation	5-2
Actuate a Driveline Using Torques and Forces	5-3
Actuate a Driveline Using Motions	5-4
Set Initial Conditions of Driveline Motion	5-5
Resolving Undetermined Motions in Complex Gears	5-5

Power Transmission Using Pulley Networks

6

Best Practices for Modeling Pulley Networks	6-2
Belt Direction	6-4
Rope Block Tension	6-5
Inertia	6-5
Verify Pulley Configuration in Power Window System	6-7

Gear Coupling Control Using Clutches

7

How a Clutch Works	7-2
Model Friction Clutches at a Fundamental Level	7-3
Engage and Disengage Gears Using a Clutch	7-4
Simulate Gear Engagement and Disengagement	7-4
How the Clutch Mode Indicates Locking and Unlocking	7-6
Brake Motion Using Clutches	7-7
Braking with a Two-Clutch System	7-7

Model Transmissions Using Gear Ratios and Clutch Schedules

8

Transmission Design Principles and Best Practices	8-2
Model a Two-Speed Transmission with Braking	8-3
Setting Up the Gears, Clutches, and Brake	8-3
Controlling the Transmission State with a Clutch Schedule	8-4
Adding Realistic Clutch Signals	8-5
Model a CR-CR 4-Speed Transmission Driveline with Braking	8-6
Replacing Programmed with Manually Controlled Clutch Pressures	8-6

Modeling Driveline Components

9

Specialized and Customized Driveline Components	9-2
Optimal Physical Modeling in the Simscape Environment	9-2
Reasons for Specialized Driveline Components	9-2
Greater Model Fidelity and Performance	9-3

Rotational-Translational Couplings	9-4
Convert Between Rotational and Translation Motion	9-4
Use Simscape and Simscape Driveline Elements to Couple Rotation and Translation	9-4
Modeling Transmissions	9-5
Transmission Templates	9-5
Transmission Ports	9-5
Gear Input Signal	9-5
Initial States	9-6
Clutch Control	9-6
Inertias and Friction Losses	9-7
Real-Time Simulation	9-7

Effective Inertias and Driveshafts

10

Model Driveshafts with Loss	10-2
--	-------------

Specialized Gears

11

Custom Planetary Gear Model	11-2
Model Gears with Losses	11-3
Constant Efficiency	11-3
Load-Dependent Efficiency	11-3
Geometry-Dependent Efficiency	11-4
Viscous Friction	11-4
Constant and Load-Dependent Gear Efficiencies	11-5

Specialized Clutches

12

Clutches, Clutch-Like Elements, and Coulomb Friction	12-2
Model Clutches with Viscous Friction Loss	12-3
Creating a Torque Damping Subsystem	12-3
Connecting and Simulating the Damped Clutch System	12-4
Model Realistic Clutch Pressure Signals	12-7
Automatic Transmission with a Dual Clutch	12-8
Predefined Simulation Options	12-8

Control Vehicle Velocity

13

Control Vehicle Throttle Input Using a Powertrain Blockset Driver	13-2
Open-Loop Simulation Using the Test - Hill Subcomponent Block	13-2
Closed-Loop Simulation Using a Longitudinal Driver Block	13-4
Simulation Comparison	13-8

Drivetrain Disturbances

14

Model Drivetrain Noise	14-2
Model and Detect Drivetrain Faults	14-10

Modeling Driveline Environments

15

Model a Road Profile with Varying Elevation and Friction	15-2
Updates to the Original Model	15-2
Run the Simulation	15-7

Analyzing Driveline Models and Simulations

16

Driveline Simulation Performance	16-2
About Simulation Performance	16-2
Adjust Model Fidelity	16-2
Improve Simulation Performance by Using the Partitioning Solver	16-3
Optimize Simulation of Stiff Drivelines	16-3
Optimize Simulation of Clutches	16-3
Resolve Partitioning Solver Simulation Issues for Simscape Driveline Models	16-6
Resolving Issues for Blocks with Stiffness or Friction	16-6
Using the Partitioning Solver	16-6
Diagnose Unexpected Simulation Behavior Due to Nonlinearities	16-7
Resolve Chatter Due to Stiffness	16-13
Driveline Degrees of Freedom	16-21
About Driveline Degrees of Freedom and Constraints	16-21
Identify Degrees of Freedom	16-21
Define Fundamental Degrees of Freedom	16-22
Define Connected Degrees of Freedom	16-23
Define Constrained Degrees of Freedom	16-24

Actuate, Sense, and Terminate Degrees of Freedom	16-27
Count Independent Degrees of Freedom	16-27
Count Degrees of Freedom in a Simple Driveline with a Clutch	16-28
Driveline States & Effect of Clutches	16-32
Driveline States and Degrees of Freedom	16-32
Find and Use Driveline States	16-32
How Simscape Driveline Simulates a Drivetrain System	16-34
About Simscape Driveline and Simscape Simulation	16-34
Clutch State Determination	16-34
Model Thermal Losses in Driveline Components	16-35
Thermal Ports	16-35
Thermal-Modeling Parameters	16-35
Model Thermal Losses for a Simple Gear	16-36
Simscape Driveline Limitations	16-42
Simscape Driveline and Simulink Limitations	16-42
Additional Simscape Driveline Limitations	16-42

Troubleshoot Driveline Simulation Issues

17

Troubleshoot Driveline Modeling and Simulation Issues	17-2
Troubleshoot Overconstrained and Conflicting Degrees of Freedom ...	17-3
Checking the Number of DoFs	17-3
Checking the Consistency of DoFs	17-3
Troubleshoot Clutch and Transmission Errors	17-4
Troubleshoot Inconsistent Initial Conditions	17-5
Troubleshoot Pulley Network Issues	17-6
Troubleshoot Engine Issues	17-7

Simscape Driveline Examples

18

Anti-Lock Braking System (ABS)	18-2
Abstract Combustion Engine Car	18-6
Automotive Clutch	18-8
Compound Pulley	18-10

Capstan with Slip	18-14
Clutches For Accelerating and Braking	18-16
Crank-Angle-Resolved Engine Model	18-18
CR-CR Four-Speed Transmission	18-21
Custom Clutch	18-24
Custom Planetary Gear	18-27
Double Cardan Joint	18-29
Dynamometer	18-31
Electro-Hydraulic Limited Slip Differential	18-33
Engine Braking	18-37
Epicyclic Gear Efficiency Measurement	18-41
Five-Speed Transmission	18-42
Fixed Caliper Disk Brake	18-46
Flexible Shaft	18-50
Four-Wheel Drive Testbed	18-53
Gear with Backlash	18-58
Gearbox Efficiency Measurement	18-61
Helicopter Transmission	18-63
Hydraulically-Actuated Driveline Clutch	18-66
Hydraulic Clutch System	18-68
Hydromechanical Hoist	18-71
Limited Slip Differential with Clutches	18-73
Manipulator with Unbalanced Load	18-77
Parallel Hybrid Transmission	18-80
Piston Engine Testbed	18-83
Power-Split Hybrid Transmission	18-85
Power Window System	18-88

Ship Propeller	18-91
Ratchet Leadscrew Mechanism	18-94
Ravigneaux Four-Speed Transmission	18-96
Series Hybrid Transmission	18-99
Sheet Metal Feeder	18-102
Shaft with Torsional and Transverse Flexibility	18-104
Simple Gear	18-107
Simpson Three-Speed Transmission	18-109
Single Cylinder Spark Ignition Engine	18-112
Stepping Mechanism with Detents	18-114
Suspension System Comparison	18-116
Torque Converter in Two-Mode Modeling	18-119
Torsen Differential	18-123
Tractor Transmission Energy Flow Chart	18-127
Transmission Testbed	18-131
Transmission Fault Detection Harness	18-134
Two Mode Hybrid Transmission	18-139
Two-Speed Transmission	18-143
Variable Ratio Gear	18-145
Vehicle in Simscape Driveline and Simulink	18-147
Tire Parameterization and Performance	18-149
Vehicle with Dual Clutch Transmission	18-155
Vehicle with Four-Wheel Drive	18-159
Vehicle with Four-Speed Transmission	18-161
Vehicle with Manual Transmission	18-165
Washing Machine Fault Analysis	18-173
Winch with Brake	18-176

Wind Turbine	18-178
Wind Turbine Driveline with Vibrations	18-190

Getting Started

Introducing Simscape Driveline Software

- “Simscape Driveline Product Description” on page 1-2
- “Related Products” on page 1-3
- “Drivetrain Model” on page 1-4
- “Capabilities of Simscape Driveline Software” on page 1-15

Simscape Driveline Product Description

Model and simulate rotational and translational mechanical systems

Simscape Driveline provides component libraries for modeling and simulating rotational and translational mechanical systems. It includes models of worm gears, lead screws, and vehicle components such as engines, tires, transmissions, and torque converters. You can use these components to model the transmission of mechanical power in helicopter drivetrains, industrial machinery, automotive powertrains, and other applications. You can integrate electrical, hydraulic, pneumatic, and other physical systems into your model using components from the Simscape family of products.

Simscape Driveline helps you develop control systems and test system-level performance. You can create custom component models with the MATLAB® based Simscape language, which enables text-based authoring of physical modeling components, domains, and libraries. You can parameterize your models using MATLAB variables and expressions, and design control systems for your physical system in Simulink®. To deploy your models to other simulation environments, including hardware-in-the-loop (HIL) systems, Simscape Driveline supports C-code generation.

Key Features

- Gear models, including planetary, differential, and worm gears with meshing losses, viscous losses, and thermal effects
- Clutch models, including cone, disk friction, synchronizer, unidirectional, and dog clutch
- Vehicle component models, including engine, tire, torque converter, and vehicle dynamics
- Models of translational elements, including leadscrew, rack and pinion, and translational friction
- MATLAB based Simscape language for creating custom component models
- Physical units for parameters and variables, with all unit conversions handled automatically
- Support for C-code generation (with Simulink Coder™)

Related Products

In this section...
“Required Products” on page 1-3
“Other Related Products” on page 1-3

Required Products

To use the Simscape Driveline product, you must have installed current versions of the following products:

- MATLAB
- Simulink
- Simscape

Other Related Products

On the MathWorks website, on the Simscape Driveline product page, the related products that are listed include toolboxes and blocksets that extend the capabilities of MATLAB and Simulink. These products can enhance Simscape Driveline modeling and simulation in various applications.

Physical Modeling Product Family

Use the Physical Modeling product family to model physical systems in Simulink. In addition to Simscape Driveline software, the product family includes:

- Simscape, the platform and unifying environment for Physical Modeling products.
- Simscape Electrical™, for modeling and simulating electronic, mechatronic, and electrical power systems.
- Simscape Fluids™, for modeling and simulating hydromechanical systems.
- Simscape Multibody™, for modeling and simulating mechanical systems.

For Information About MathWorks Products

- If you have the product installed, see the online documentation for that product.
- See the “Products” section at the MathWorks website at www.mathworks.com.

Drivetrain Model

In this section...
“What the Model Represents” on page 1-4
“What the Model Illustrates” on page 1-4
“Open CR-CR Transmission Example Model” on page 1-5
“Run the Model” on page 1-8
“Modify the Model” on page 1-10

What the Model Represents

The “CR-CR Four-Speed Transmission” on page 18-21 example model simulates a complete drivetrain. This example helps you understand how to model driveline components with Simscape Driveline blocks, connect them into a realistic model, use Simulink blocks and variant subsystems in driveline modeling, and simulate and modify a drivetrain model.

This driveline mechanism is part of a full vehicle, without the engine or engine-drivetrain coupling, and without the final differential and wheel assembly. The model includes an actuating torque, driver and driven shafts, a four-speed transmission, and a braking clutch.

For complete vehicle models that use this drivetrain, see the “Vehicle with Four-Speed Transmission” on page 18-161 example model and “Complete Vehicle Model” on page 3-2.

What the Model Illustrates

The `CRCRFourSpeedTransmissionExample` model contains a driveline that accepts a driving torque. The driveline system transfers this torque and the associated angular motion from the input or drive shaft to an output or driven shaft through a transmission. The model includes a CR-CR (carrier-ring-carrier-ring) four-speed transmission subsystem, based on two gears and four clutches. (The example does not use the reverse gear in the CR-CR transmission.) You can set the transmission to four different gear combinations, allowing four different effective torque and angular velocity ratios. A fifth clutch, outside the transmission, acts as a brake on the driven shaft.

The transmission subsystem illustrates a critical feature of transmission design, the *clutch schedule*. To be fully engaged, the transmission, with four clutches and two planetary gears, requires two clutches to be locked and the other two unlocked at any time. (The transmission reverse clutch is not applicable here.) The choice of which two clutches to lock determines the effective gear ratio across the transmission. The clutch schedule is the relationship shown in the table of locked and free clutches corresponding to different gear settings. If all four clutches are unlocked, the transmission is in neutral. If the clutches are disengaged, no torque or motion at all is transferred across the transmission.

Clutch Schedule for the CR-CR 4-Speed Transmission

Gear Setting	Clutch A State	Clutch B State	Clutch C State	Clutch D State	Clutch R State	Drive Ratio
1	<i>L</i>	F	F	<i>L</i>	F	$1 + g_o$
2	<i>L</i>	F	<i>L</i>	F	F	$1 + g_o/(1 + g_i)$
3	<i>L</i>	<i>L</i>	F	F	F	1
4	F	<i>L</i>	<i>L</i>	F	F	$g_i/(1 + g_i)$
Reverse	F	F	F	F	<i>L</i>	$-g_i$

- *L* = locked
- F = free
- g_i = Input Planetary Gear ring-to-sun gear ratio
- g_o = Output Planetary Gear ring-to-sun gear ratio

Clutch Control Variant Subsystem

A Variant Subsystem block governs transmission gear changes. This block, named Clutch Control, contains two child subsystem blocks that provide different clutch control modes, or *variants*:

- Manual — Manually switch transmission clutches.
- Programmed — Automatically switch transmission clutches according to a programmed clutch schedule.

During simulation, one variant becomes active while the other does not. The choice of active variant determines which child subsystem controls the gear changes. By default, the Programmed variant is active and gear changes follow a programmed clutch schedule. To switch gears manually during simulation, change the active variant to Manual.

Open CR-CR Transmission Example Model

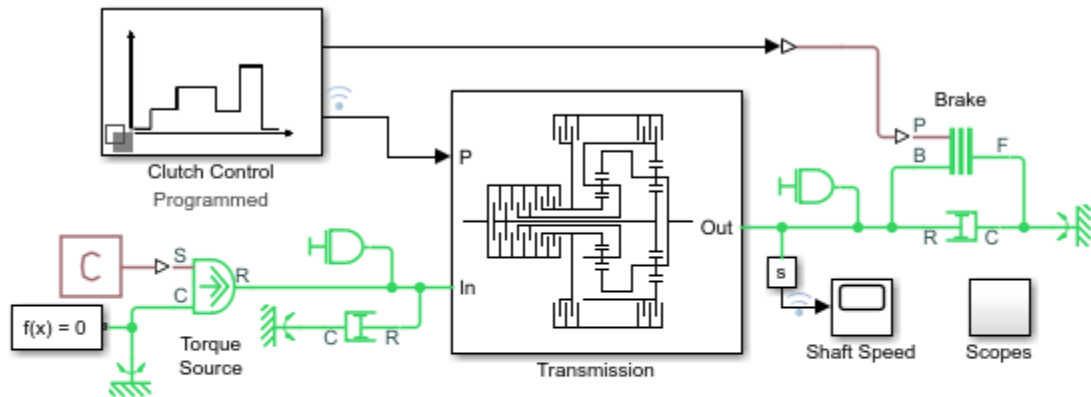
To open the CR-CR transmission example model, at the MATLAB command prompt, enter

```
openExample('sdl/CRCRFourSpeedTransmissionExample')
```

Block Diagram Model

Examine the model and its structure. The main model window contains the transmission subsystem, the input shaft assembly, and the output shaft assembly. Each assembly consists of a driveline axis with applied damping and inertia torques. Each driveshaft balances the torques applied across its ends with the damping and inertia forces. A net torque is transmitted along the driveline.

The main model also includes a brake clutch. When this clutch is locked, the shaft slows, but doesn't necessarily stop. The transmission can be engaged at the same time as the brake. If the transmission is engaged, the clutch remains unlocked.



CR-CR Four-Speed Transmission

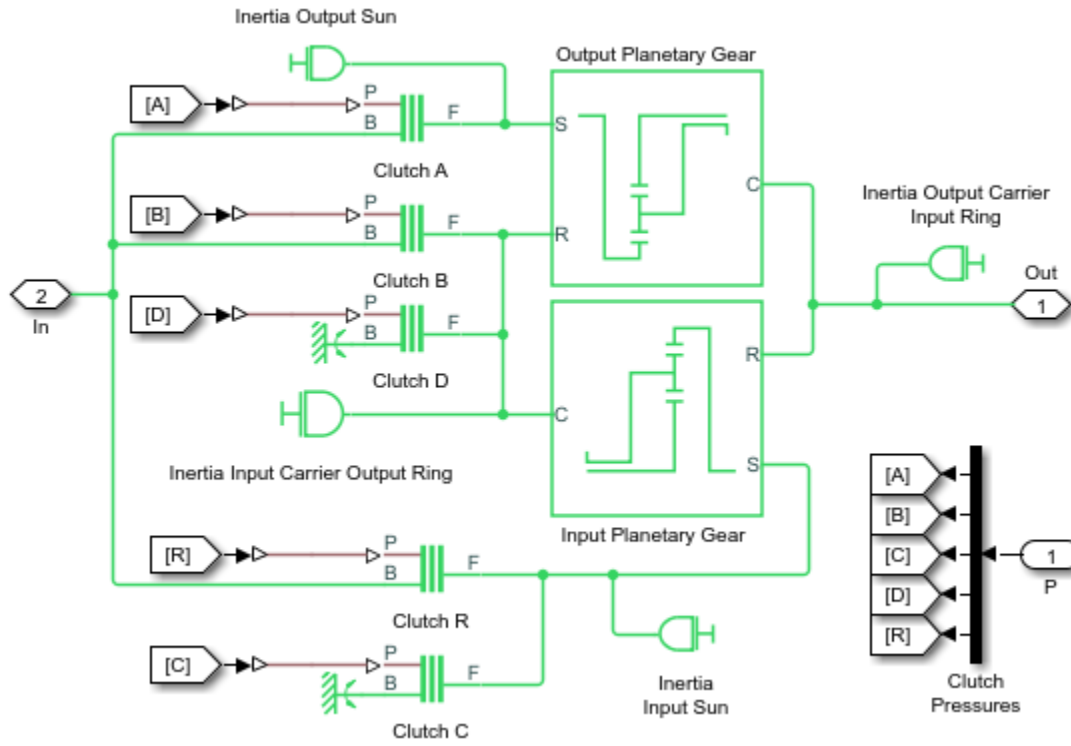
1. [Plot speeds](#) of shafts ([see code](#))
2. Configure Clutch Control: [Programmed](#), [Manual](#)
3. [Explore simulation results](#) using [sscexplore](#)
4. [Learn more](#) about this example

Main Model Window

What the Model Contains—Opening the Subsystems

Open each subsystem.

The transmission subsystem contains four clutches, two planetary gears, and four inertias (rotating bodies). Ignoring the reverse gear and its clutch, this transmission has four possible (forward) gear settings. Exactly two clutches must be locked at any one time for the transmission to engage and to avoid conflicting constraints on the gear motions.

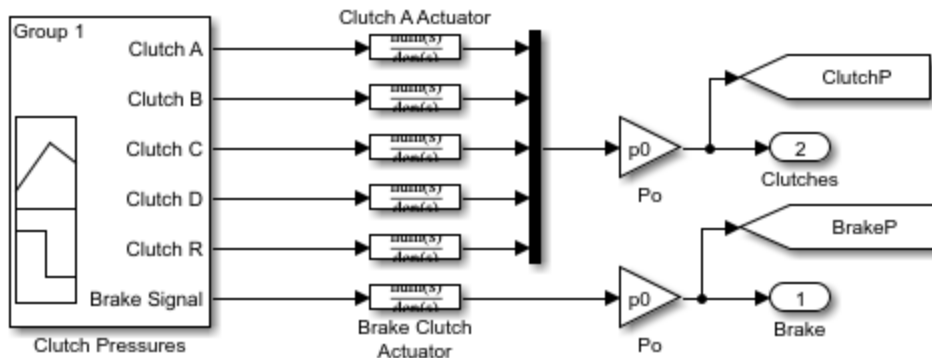


CR-CR 4-Speed Transmission Subsystem

The Clutch Control variant subsystem provides the pressures that lock the necessary clutches. In its default state, the clutch controller is programmed to move the transmission through a fixed sequence of gears, then unlock all the transmission clutches. This control program allows the driven shaft to “coast” for a time, and then engage and lock the brake clutch to stop the driven shaft.

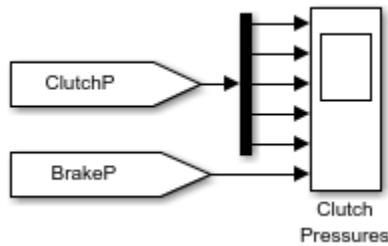
Programmed Clutch Control for the CR-CR Transmission

Use this subsystem to control the CR-CR transmission with a preprogrammed schedule of clutch pressures.



Clutch Control Subsystem

The Scopes subsystem provides Scope blocks to display the clutch pressure and the input and output shaft velocity signals.

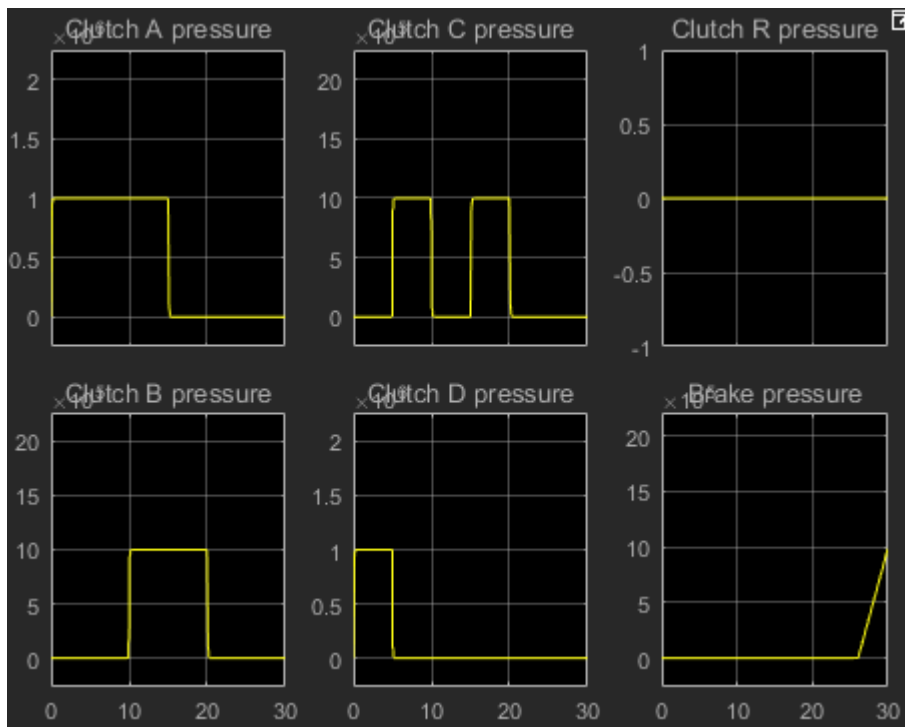


Scopes Subsystem

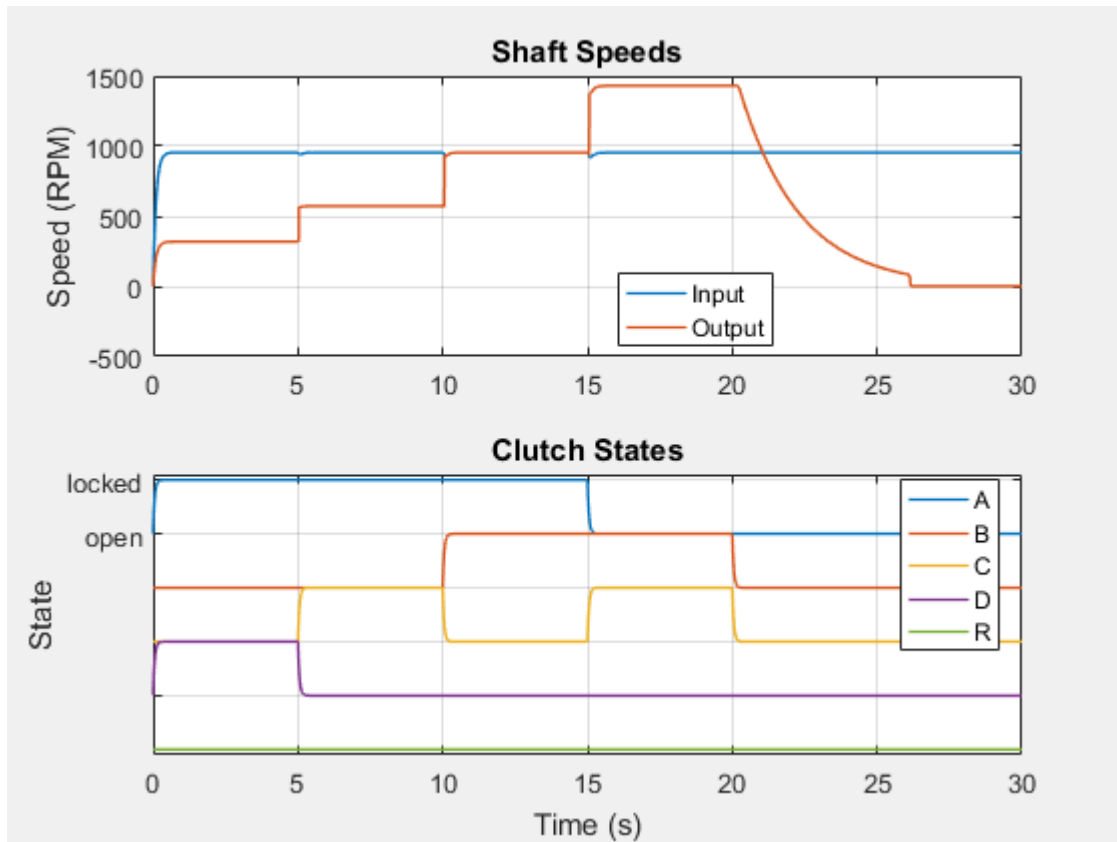
Run the Model

To display the CR-CR driveline model behavior:

- 1 Open the Scopes subsystem and then the Scope block. Close the Scopes subsystem.
- 2 Click **Start**. The model steps through the gears and then brakes.
- 3 Observe how the clutch pressure signals move the transmission into one gear after another, at 0, 5, 10, and 15 seconds of simulation time. To determine which gear settings the model is implementing, compare these clutch pressure signals to the clutch schedule in the CR-CR transmission subsystem. The model steps through gears 1, 2, 3, and 4, before coasting and then braking.

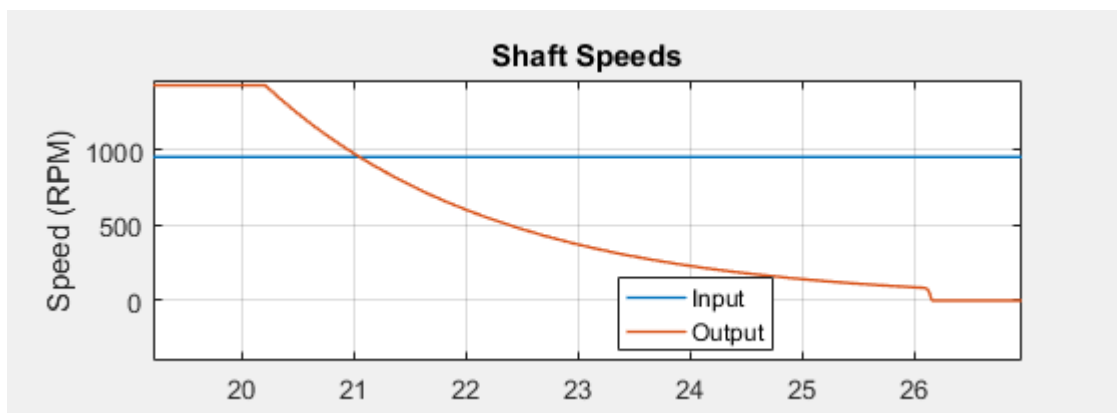


- 4 To compare the angular velocities of the input and output shafts, click **Plot speeds**. A figure that contains the Shaft Speeds and Clutch States opens.



In the transmission, the two planetary gears are coupled in different ways in the different gear settings, producing different relationships between the driven and driver shaft velocities. The effective *drive ratio* of output to input shafts is the *reciprocal* of the ratio of output to input angular velocities.

- 5 Zoom to observe the results for the shaft speeds 20–26 seconds.



The transmission clutch pressures drop to zero, and the transmission disengages. The transmission ceases to transfer angular motion and torque from the driver to the driven shaft, and the driven shaft continues to spin from inertia alone. A small kinetic friction damping gradually slows the driven shaft over the next six seconds.

At 26 seconds of simulation time, the brake clutch pressure begins to rise from zero, and the brake clutch engages. The driven shaft decelerates more drastically now. 26.0–26.2 seconds, the brake clutch locks, and the driven shaft stops rotating completely.

Modify the Model

You can modify this example model to explore other Simscape Driveline features. Here you modify and rerun the model to investigate two aspects of its motion.

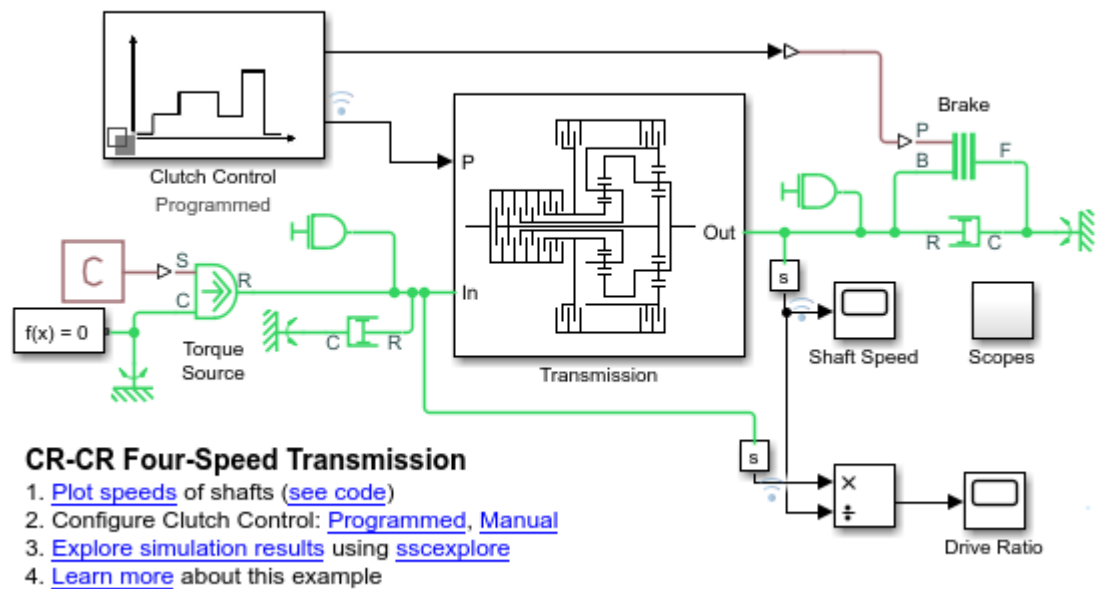
- Measure the effective drive ratio of the CR-CR transmission in each gear setting that it steps through.
- Change the gear sequence.

Measuring the Drive Ratio of the CR-CR Transmission States

A transmission is a set of coupled gears. For a particular transmission gear setting, the ratio of driven (output) shaft velocity to the driver (input) is fixed. The reciprocal ratio, the *drive ratio*, is like a gear ratio of an individual gear coupling, but for the whole transmission.

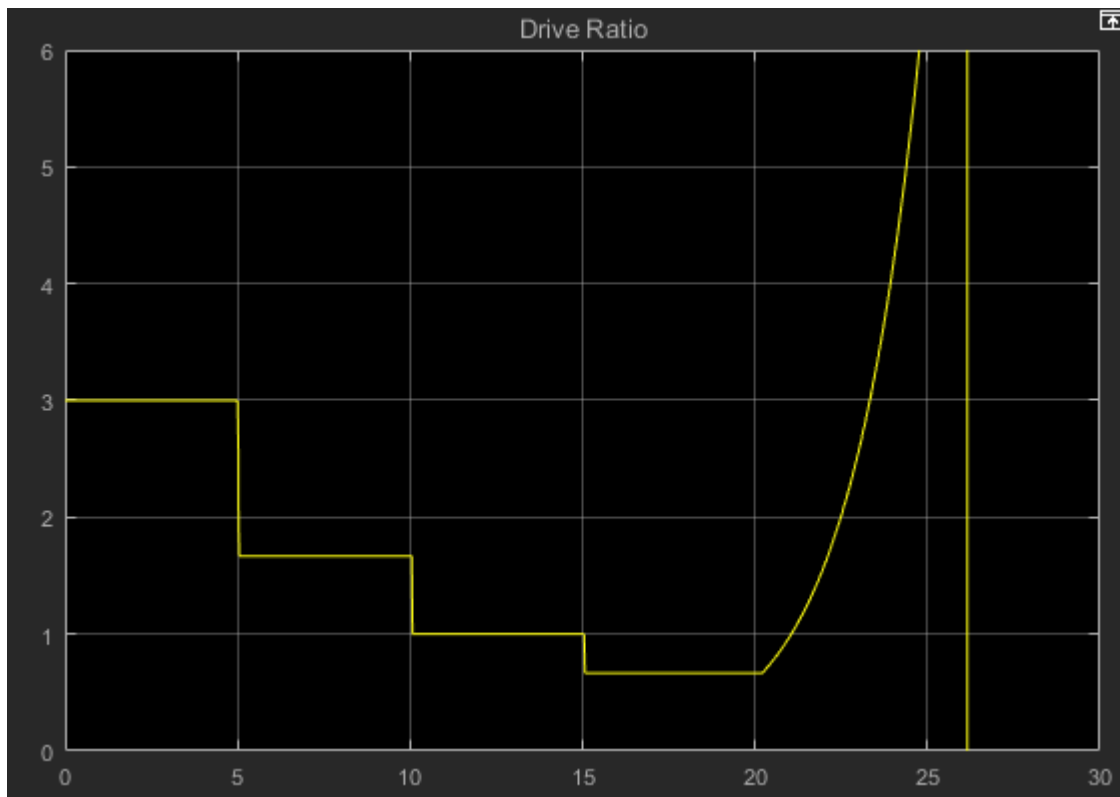
The drive ratio is the ratio of input to output shaft velocities. Add and connect the necessary Simulink blocks to measure the drive ratio for the CR-CR 4-speed transmission.

- 1 Collect data for the angular velocity for the driver shaft:
 - a Make a copy of the S sensor subsystem that is connected to the Out port of the transmission subsystem. The output sensor captures the angular velocity of the driven shaft.
 - b Connect the new sensor subsystem to the connector between the input shaft assembly and the In port of the transmission subsystem.
- 2 To calculate the drive ratio, from the Simulink Library Browser, from the **Simulink > Math Operations** library, add a Divide block.
- 3 To visualize the drive ratio, add and configure a Scope block:
 - a Make a copy of the Shaft Speed scope block.
 - b Change the name the new scope block to `Drive Ratio`.
 - c Open the Drive Ratio block.
 - d Open the configuration parameters for the scope.
 - e On the **Display** tab, set the Y-limits (Minimum) to 0 and the Y-limits (Maximum) to 6.
 - f Connect the block as shown in the figure.
 - g Label the input signal to the Drive Shaft block as `Drive Ratio`.



- 4 Simulate the model. Observe how the drive ratio steps through a sequence of five-second states, in parallel with the clutch pressures and clutch modes, until it reaches 20 seconds. The drive ratio measurement after 20 seconds is not meaningful because the transmission is uncoupled.

Just after 26 seconds, the driven shaft velocity drops to zero, and the Divide block produces divide-by-zero warnings at the MATLAB command line.



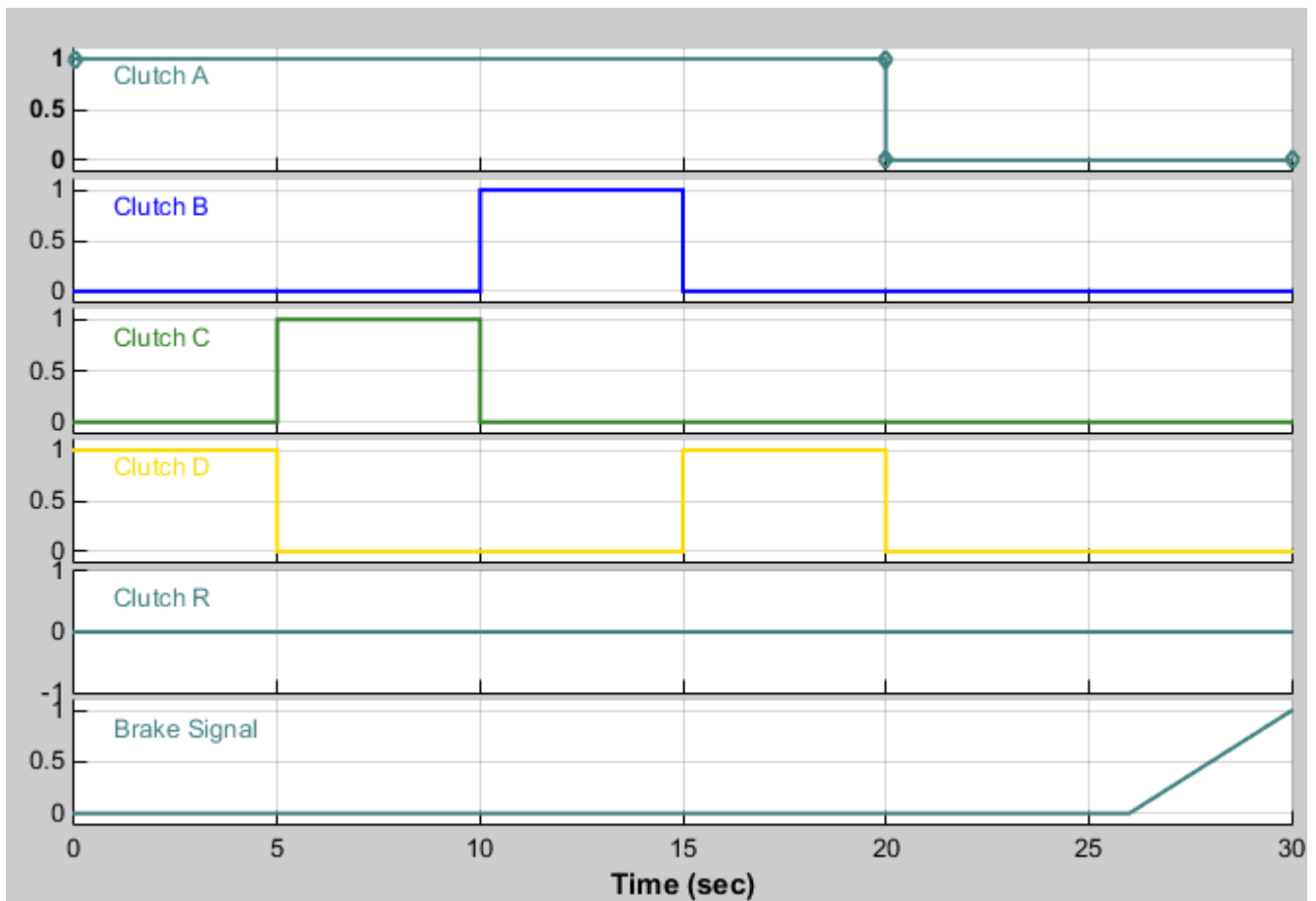
- Consult the table, Clutch Schedule for the CR-CR 4-Speed Transmission. Check the drive ratios for each gear, 1, 2, 3, and 4, in terms of the gear ratios of the two Planetary Gears in the transmission. Determine the numerical values for these drive ratios for gear settings 1, 2, 3, and 4. Then check them against the values displayed in the Drive Ratio scope.

The drive ratio sequence is 3, 5/3, 1, and 2/3, respectively, for the first, second, third, and fourth intervals of five seconds each.

Changing the Transmission Gear Sequence

When you first open the CR-CR FourSpeedTransmission example, the Clutch Control variant subsystem is programmed to step through CR-CR gear settings 1, 2, 3, and 4, before disengaging. Modify it to step through settings 1, 2, 3, and 1, then disengage. The fourth gear requires that A is free, B is locked, C is locked, and D is free. Modify the clutch pressure signal sequence from 15 to 20 seconds so that the transmission is set in first, not fourth, gear. The first gear requires clutches that A and D are locked and clutches B and C are free.

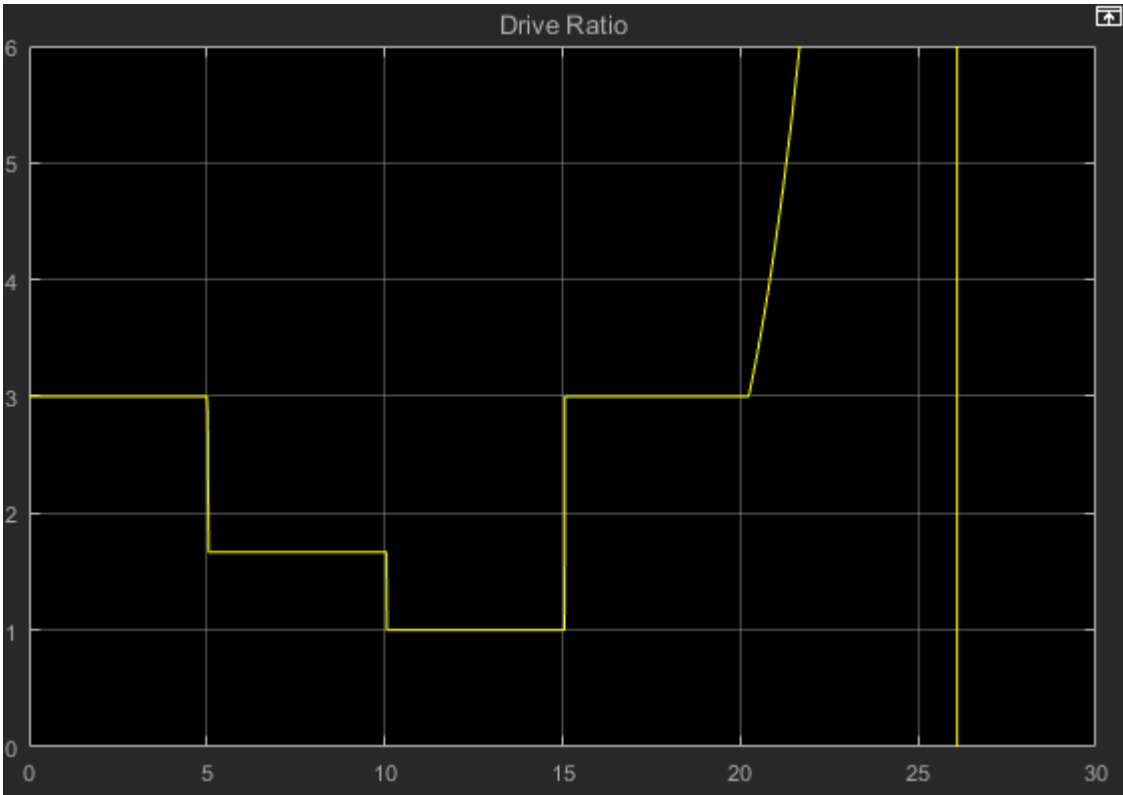
- Determine the clutch states that correspond to first gear. Refer to table Clutch Schedule for the CR-CR 4-Speed Transmission.
- Double-click the Clutch Control subsystem.
- In the Clutch Control subsystem, double-click Programmed.
- In the Programmed subsystem, double-click Clutch Pressures. The signal builder window opens with the clutch pressure signals.
- In the time interval 15–20 seconds, update clutch signals A through D to match first gear. Clutches A and D must lock, while clutches B and C must remain free. Specify a signal value of one to lock a clutch, zero to unlock it.



Modified CR-CR 4-Speed Transmission Clutch Pressures

6 Run simulation.

Clutch pressures, clutch modes, and driven shaft velocities in the time interval 15-20 seconds now correspond to first gear. Refer to the Drive Ratio plot for the updated model. The ratio has changed from $2/3$ (fourth gear) to 3 (first gear) accordingly.



Capabilities of Simscape Driveline Software

In this section...

“What Simscape Driveline Software Contains” on page 1-15
 “Model Driveline Systems” on page 1-16
 “Model Inertias and Gears” on page 1-16
 “Model Dynamic Driveline Elements” on page 1-17
 “Model Custom Driveline Elements” on page 1-17
 “Actuate and Sense Motion” on page 1-17
 “Simulate and Analyze Motion” on page 1-17

What Simscape Driveline Software Contains

Simscape Driveline software is a set of block libraries in the Simulink environment and based on Simscape software. You connect Simscape Driveline blocks to normal Simulink blocks through Simscape physical signal blocks that define physical units.

The blocks in the Simscape Driveline library and the related mechanical blocks in the Simscape Foundation library are the elements to model driveline systems. These systems consist of one or more inertias and masses, rotating about or translating along one or more axes, constrained to rotate or translate together by gears, which transfer torque and forces to different parts of the driveline. You can represent drivelines with components organized into hierarchical subsystems, as in any Simulink model. You can:

- Constrain motion with gears.
- Add complex dynamic elements such as clutches, clutch-like elements, and other couplings.
- Represent such vehicle components as bodies and tires.
- Actuate bodies with torques, forces, and motions.
- Integrate the Newtonian rotational and translational dynamics, then measure the resulting motions.


Relation to Simscape Software

To model and simulate physical systems, Simscape Driveline models use such Simscape technologies as nondirectional physical connections and conserving ports, physical signals carrying physical units, custom component modeling, specialized solvers, and data logging.


The Simscape mechanical rotational and translational domains form the basis of the Simscape Driveline block libraries and models. The Simscape Foundation library includes physical signal blocks; and basic mechanical blocks representing inertia, mass, and simple mechanical couplings. It also includes motion, torque, and force sources and sensors.

For more about modeling and simulating in the Simscape environment, see the “Simscape” documentation.

Physical Connections, Mechanical Conserving Ports, and Physical Signals

On Simscape Driveline blocks, the mechanical conserving ports  anchor physical connection lines that, in this domain, represent mechanical axes. These axes are either rotation axes along which

torque is transferred and around which inertias rotate, or translation axes along which force is transferred and along which masses translate.

Certain blocks defined in Simscape domains also require input or output signals that carry physical units, or physical signals. Simscape physical signal lines and ports  represent and connect physical signals with units. Conversion blocks allow you to convert dimensionless Simulink signals to Simscape physical signals, and back.

Model Driveline Systems

Simscape Driveline software extends Simulink and Simscape software with blocks to model driveline components and properties, represent drivelines as physical networks, and to solve the equations of motion.

To build and run a Simscape Driveline model representation of a driveline:

- 1 Specify rotational inertia or translational mass for each body. Connect the bodies with physical connection lines representing driveline axes at mechanical conserving ports.

If needed, ground the driveline to one or more mechanical references fixed in space.

- 2 Constrain the driveline axes to rotate or translate together by connecting them with gears. Gears impose static constraints on driveline motions and transfer torques and forces at fixed ratios.
- 3 As necessary, add dynamic elements that transfer torque, force, and motion among driveline axes in a nonstatic way. These elements include internal torque-generating components such as damped springs, clutches, clutch-like elements, transmissions, and torque converters. You can also construct and connect your own dynamic elements.

Similarly, add dynamic sources and environmental interactions, such as engines, vehicle bodies, and tires.

- 4 Set up mechanical sources and sensors to initiate and record body motions, and to apply external torques and forces to the driveline.
- 5 Connect the Simscape Solver Configuration to the driveline, then configure it. Start the simulation, calling the Simulink and Simscape solvers to find the motions of the system. Display and analyze the motion.

Model Inertias and Gears

Simscape Driveline software defines a driveline as a collection of rotating and translating bodies, defined by their rotational inertias and translational masses. Rotational and translational degrees of freedom (DoFs) originate on inertias and masses, but are carried by physical connection lines. Directly connecting one body to another constrains both bodies to rotate at the same angular or linear velocity. A torque or force applied to one body is applied to both. You can also ground driveline axes to mechanical references that do not move and that represent infinite effective inertia or mass.

Note All Simscape Driveline DoFs are absolute in an implicit global coordinate system at rest, but are measured and used in a relative way, between one component and another. To measure regarding the global rest frame, ground sensors or other components with mechanical reference blocks.

In a real driveline, the bodies can also be connected indirectly by gears that couple driveline axes. The gears constrain the axes to rotate together. These gears can be simple or complex and can couple two or more axes. The gears have two roles:

- Constraining the connected axes to rotate or translate together at velocities in fixed ratio or ratios.
- Transferring the torques or forces flowing along one or more axes to other axes, also in fixed ratio or ratios.

Model Dynamic Driveline Elements

To create more realistic driveline models, you elaborate on simple drivelines consisting of inertias, masses, and gears. You add complex mechanical elements that generate torques and forces internally within the driveline, between one axis and another. Certain Simscape Driveline blocks encapsulate as subsystems entire models of complex driveline elements:

- Load-dependent loss models of nonideal gears
- Clutches and clutch-like elements that model the locking and unlocking of pairs of driveline axes by applying kinetic and static friction
- Vehicle component models that represent engines, tires, and vehicle bodies
- Specialized torque and force models, such as torque converters, hard stops, and damped spring-like torsion

Model Custom Driveline Elements

The blocks provided in the Simscape Foundation library can serve as starting points for developing variant or entirely new models to simulate the same components. You can also study masked subsystems by looking under their masks. If necessary, break the link between the block and the library before modifying it, and then create your own version. Or, create your own new blocks using Simscape Driveline and Simscape components, or with the Simscape language.

For more information on specialized driveline components, see “Specialized and Customized Driveline Components” on page 9-2.

Actuate and Sense Motion

Simscape motion sources and sensors are the blocks that you use to insert and extract basic kinematic and dynamic information:

- Source blocks impart motion to driveline axes and impose externally defined torques and forces on the bodies of a driveline.
- Sensor blocks measure the motions of, and the torques and forces transferred along, the axes of a driveline system.

Source inputs and sensor outputs are physical signals that carry units.

Simulate and Analyze Motion

Once you specify all the rotational inertias and translational masses of the bodies and interconnect the bodies with gears and other driveline elements, the dynamic problem of finding the system

motion is solvable. To finish a driveline model and prepare it for simulation, you connect the driveline to the Simscape solver. This solver defines certain aspects of the simulation and integrates the Newtonian dynamics for the system, applying all internal and external torques and constraints to find the motions of the bodies.

Once your model is ready for simulation, run it and analyze its motions, torques, and forces.

Inverse Dynamics – Trimming and Linearization

You typically do not know the torques and forces necessary to produce a given set of motions. By motion-actuating your driveline with motion sources and measuring the resulting torques and forces, you can find the torques and forces required to produce specified motions. This technique inverts the canonical approach to dynamics, which consists of finding motions from torques and forces.

A special case of inverse dynamics is *trimming*. This technique involves searching for steady-state motions of the bodies, when their accelerations and the torques and forces they experience vanish. Using the specialized tools in Simscape and Simulink, you can perturb such a steady motion state slightly to find how the system responds to small disturbances. The response indicates the system stability and suitability for controllers.

Generating Code – Real-Time and Hardware-in-the-Loop Simulation

Simscape Driveline software is compatible with Simulink Acceleration modes, Simulink Coder, and Simulink Real-Time™ software. With these products, you can generate code versions of the models that you originally create in Simulink with block diagrams, enhancing simulation speed and model portability. A common application of generated code is defining real-time and hardware-in-the-loop simulations.

The presence of clutches in a driveline model induces locking-unlocking iterations and dynamic discontinuities. These discontinuities place certain restrictions on code generation. For more information about these restrictions, see “Driveline Simulation Performance” on page 16-2 and “Simscape Driveline Limitations” on page 16-42.

Modeling Driveline Systems

- “Start a New Simscape Driveline Model” on page 2-2
- “Build a Drivetrain Model” on page 2-3

Start a New Simscape Driveline Model

You can use the Simscape model template as a starting point for your Simscape Driveline models. The template provides the required Solver Configuration block and the commonly used Simulink-PS Converter and PS-Simulink Converter blocks.



To open the Simscape model template, at the MATLAB command prompt, enter `ssc_new`. Simscape software opens the model template along with the Foundation block library. To open the Simscape Driveline block library, enter `sdl_lib`.

Build a Drivetrain Model



The example model in “Drivetrain Model” on page 1-4 illustrates a typical drivetrain system that you can model with Simscape Driveline software. It also illustrates the key rules for connecting driveline blocks to each other and the dual roles of Simscape physical connection lines in driveline modeling. Within the Simscape mechanical domain:

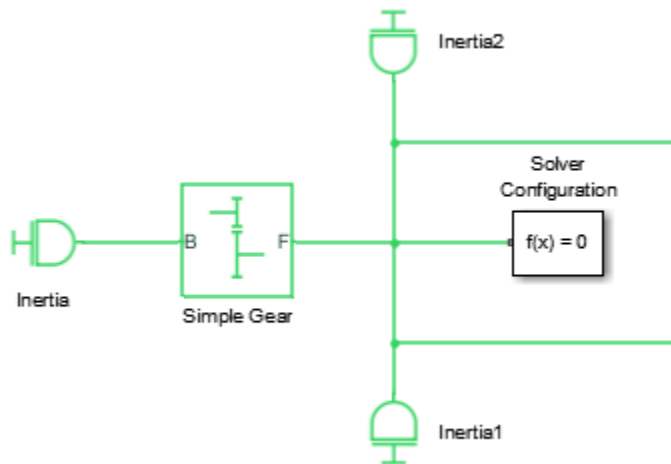
- The *across* variable is angular or linear velocity, depending on the type of mechanical ports you are connecting to, rotational or translational. Along any connection line, the velocity is the same.
- Depending on the type of mechanical ports, you are connecting to, the *through* variable is torque or force. Torques and forces are conserved along a connection line and sum to zero at line branch points.

Before building and running the tutorial models, review these rules.

- Driveline blocks feature mechanical conserving ports  and, in some cases, physical signal ports . You can connect mechanical ports only to other mechanical ports, and physical signal ports only to one another.

You cannot mix rotational and translational ports, or connect a mechanical conserving port to a physical signal port.

- The physical connection lines interconnecting mechanical ports  represent driveline axes and enforce physical relationships. Unlike physical signal and Simulink lines, they do not represent signals or mathematical operations, and they have no inherent directionality.
- A driveline connection line represents an idealized massless and perfectly rigid spinning shaft or translating axis. A driveline connection line between two ports constrains the two driveline components that are connected to the line to rotate or translate at the same velocity.
- You can branch mechanical connection lines. Connect the end of any branch of a mechanical connection line to a mechanical port .
- Branching a driveline connection line modifies the physical constraints that it represents. All driveline components connected to the ends of a set of branched lines rotate or translate at the same velocity. For lines branched from a branch point, the sum of all torques or forces flowing in equals the sum of all torques or forces flowing out. How the torque or force is divided depends on the defining equations of the attached blocks in the rest of the system.
- Mechanical connection lines satisfying the velocity constraint must have the same initial velocities.



Branching Driveline Connection Lines

The Solver Configuration block in this example does not use any torque. It does share the angular velocity constraint from the branch point. Symbolically, the branching conditions on driveline connection lines are:

$$\omega_1 = \omega_2 = \omega_3 \dots$$

and

$$\tau_1 + \tau_2 + \tau_3 + \dots = 0$$

The sign convention is that torques flowing in are positive. Like all driveline axes, these elements have no inherent directionality. Torque flow directions are defined by overall system equations during simulation.

Torque and motion are transferred through the driveline from some driveshafts to others. Certain Simscape Driveline blocks require explicit directionality and represent it by designating one driveline connector port as the input *base* (B) and the other as the output *follower* (F), or some equivalent pair. When required, positive relative motion of driveline axes or shafts is measured as follower relative to base.

Motion Is Absolute Except when relative motion is explicitly required, all motion in Simscape Driveline and Simscape models is measured in implicit absolute coordinates. The Mechanical Rotational Reference and Mechanical Translational Reference blocks define the absolute zero velocity. If they are connected to a driveline axis, these blocks enforce this zero-motion state on that axis.

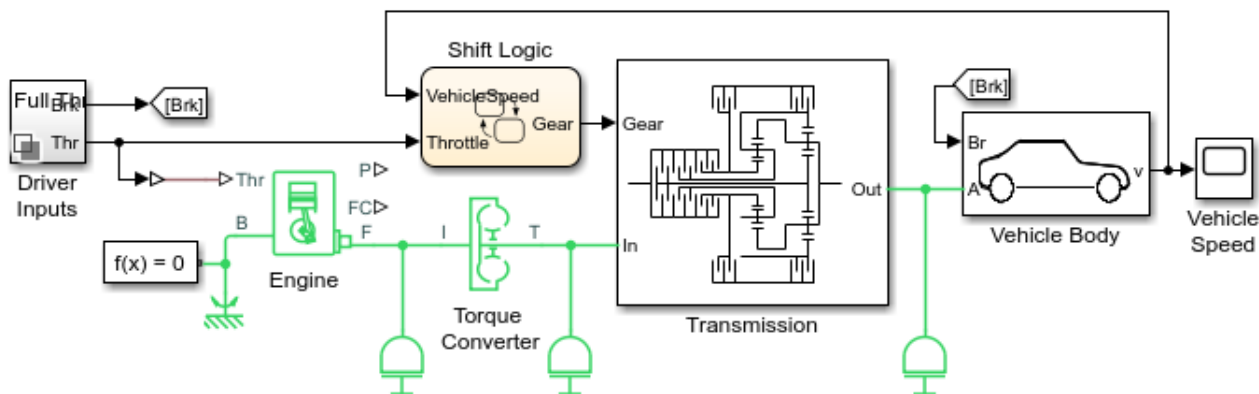
Complete Vehicle Model

Complete Vehicle Model

The full car drivetrain simulation of the “Vehicle with Four-Speed Transmission” on page 18-161 example encompasses all the basic methods of driveline modeling and many key Simscape Driveline features. It includes engine and transmission models and a model of the drivetrain-wheel-road coupling. The engine and transmission are coupled with a torque converter. Programmed clutch control steps the transmission through four gears during the simulation. The clutch pressure signals are smooth and more realistic than the sharp clutch pressure signals in the simpler drivetrain examples. This section describes these features, subsystems, and their relationship and purposes, leading you to actual simulation.

Understanding the Global Structure of the Model

Open the example. The model contains model workspace variables for parameterizing some of the blocks. For information on creating, accessing, and changing model workspace variables, see “Specify Source for Data in Model Workspace” and “Change Model Workspace Data”.



Vehicle with Four-Speed Transmission Model

The main driveline subsystems and components are:

- Driver Inputs — Throttle/brake profile
- Engine — System-level model of spark-ignition and diesel engine
- Torque Converter — Three-part torque converter consisting of an impeller, a turbine, and a stator.
- Transmission subsystem — CR-CR 4-speed transmission
- Shift Logic — Stateflow[®] implemented transmission controller
- Vehicle Body — Vehicle, tire, and brake dynamics

While the engine is idling initially at a nonzero speed, the transmission output and the vehicle as a whole are initially not moving.

Model the Throttle/Brake Profile

The Driver Inputs block is a subcomponent implementation that provides throttle and brake signals to the engine and transmission control system. Open the Driver Inputs block to view the throttle/brake profile for the simulation.

The throttle signal is programmed to produce a realistic acceleration profile and to agree with the gear shifting sequence described in “Control the Clutches” on page 3-6. The throttle signal feeds to the engine and to the transmission controller.

The brake signal supplies the input force that actuates braking in a Double-Shoe Brake block in the Vehicle Body subsystem.

Model the Engine

For the purposes of system modeling, an engine or motor specifies an output torque as a function of driveline speed. The engine has a connection port coupling it rotationally to the rest of the system.

Using an Engine Block from Vehicle Components

The Engines library contains blocks that you control using an input physical signal for the throttle. You can parameterize the Generic Engine block using vectors to specify speed and torque. The block calculates the maximum possible torque as a function of the engine speed at any instant. The throttle signal controls how much of the maximum torque the engine can deliver. The Piston Engine block accounts for the instantaneous torque transmitted to the engine drive shaft. The instantaneous torque enables you to model vibrations in the drivetrain due to piston revolution. To model just the piston mechanism of a combustion engine, use the Piston block.

The `VehicleWithFourSpeedTransmissionExample` example uses a Generic Engine block, configured as spark-ignition type. The block properties specified in the property inspector include the maximum power, speed at maximum power, and maximum possible speed of the engine. To view engine settings, click the Engine block. The engine torque and motion are modeled relative to the rotational ground, which is taken as the base reference of the engine and the starting point of the driveline, or mechanical rotational, connections in this model.

Alternative and Advanced Methods for Modeling Engines

Simscape Driveline allows you to create complex, custom engine models. Several important engine features to consider in a complex model are:

- Distinguishing steady-state behavior from engine start-up, when the engine speed-engine torque function has not yet reached its maximum possible envelope
- Details of mechanical power production, such as air-fuel compression and combustion
- Additional controls beyond what can be represented by a single throttle signal

Model the Transmission

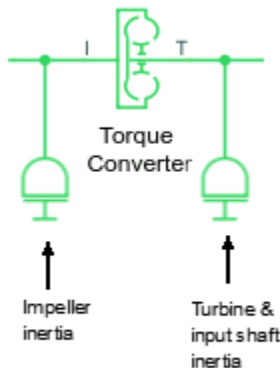
The CR-CR 4-speed transmission subsystem in the `VehicleWithFourSpeedTransmissionExample` model is similar to other examples with the same transmission. The clutch and planetary gear properties are set in the blocks with model workspace variables.

Workspace Variable	Description
eff_tor_rad	Clutch: effective torque radius (m)
num_fric_surf	Clutch: number of friction surfaces in contact
engagement_area	Clutch: friction surface area in contact (m ²)
fric_coeff	Clutch: kinetic friction coefficient of surfaces in contact
peak_normal	Clutch: static (locking) friction coefficient of surfaces in contact
velTol	Clutch: clutch velocity locking tolerance (rad/s)
pressThresh	Clutch: Normalized pressure threshold
p0	Clutch: Physical pressure normalization (Pa)

For more about gears, clutches, and transmissions, see the Disk Friction Clutch block reference page.

Couple the Engine to the Transmission

The `VehicleWithFourSpeedTransmissionExample` model couples the engine and the transmission through a torque converter block.



Torque Converter Stage

Like a clutch, a torque converter couples two independent driveline axes to transfer angular motion and torque from an input to an output shaft. However, unlike a clutch, a torque converter never locks. The torque converter transfers motion by hydrodynamic viscosity, not by surface friction. Thus a torque converter does not step through discrete stages and avoids the motion discontinuities inherent in friction clutches.

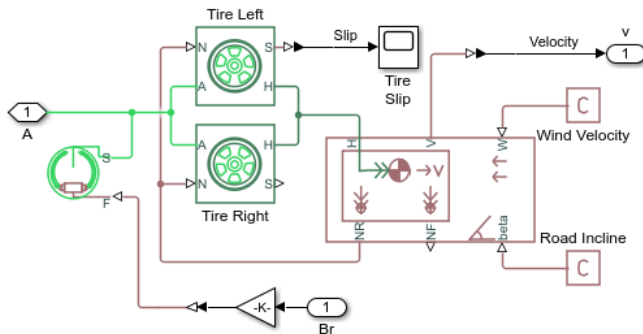
To mimic engine idling at the start of the simulation, the initial condition of the impeller inertia is a nonzero angular velocity. The initial condition of the turbine & input shaft inertia is zero speed.

For more details about these blocks, see the Torque Converter and Inertia block reference pages.

Model the Tires, Brakes, Wheels, and Road

The transmission feeds its output torque to the final drive subsystem, Vehicle Body. This subsystem represents the vehicle inertia (the load on the transmission), the wheels, the brakes, the driving

conditions, and the wheel contact with the road. The subsystem models only the rear wheels as driven by the transmission.



Final Drive Subsystem: Vehicle Body

The subsystem has two major areas.

Modeling the Tires and Brakes

The right and left tire blocks accept the driveline torque and rotation from the transmission at their wheel axle rotational ports (A). Given a normal or vertical load (N), this torque and rotation are converted to a thrust force and translation at the wheel hub translational ports (H).

The tires rotate nonideally, slipping before they fully generate traction and react against the road surface. The tire slip of the left tire is reported as a physical signal and converted to Simulink for use with the Tire slip scope.

The Double-Shoe Brake block represents a brake arranged as two pivoted rigid shoes that are symmetrically installed inside or outside of a drum and operated by one actuator. The brake block converts the braking signal from the Driver Inputs block to an actuator force that exerts a friction torque on the shaft that connects the brake drum to the tire blocks.

Modeling the Vehicle Body and Load

The driveline connection line sequence of the model ends with the Vehicle Body block, which specifies the vehicle geometry, mass, aerodynamic drag, and initial velocity (zero). This block generates the normal forces that the Tire blocks accept as vertical loads. Vehicle Body accepts the developed thrust force and motion at its horizontal motion translational port (H). The vehicle body model also accepts a wind velocity (W) and a road incline (beta), both provided by physical constants.

The rear wheel vertical load force (NR) is reported back to the Tire blocks. The forward wheel vertical load (NF) is not used.

The forward velocity (V) of the vehicle is converted and reported, through the subsystem output, to the Vehicle velocity scope.

Alternative Differential, Wheel, Road, and Braking Models

The `VehicleWithFourSpeedTransmissionExample` example models only the rear wheels, the rear tires, and the vehicle body, without the more realistic drivetrain components of differential gears and brakes. The `VehicleWithFourWheelDriveExample` example illustrates how to model a vehicle that has four wheels and front and rear differential gears.

For information on modelling brake systems using clutches, see “Brake Motion Using Clutches” on page 7-7 and “Model a Two-Speed Transmission with Braking” on page 8-3.

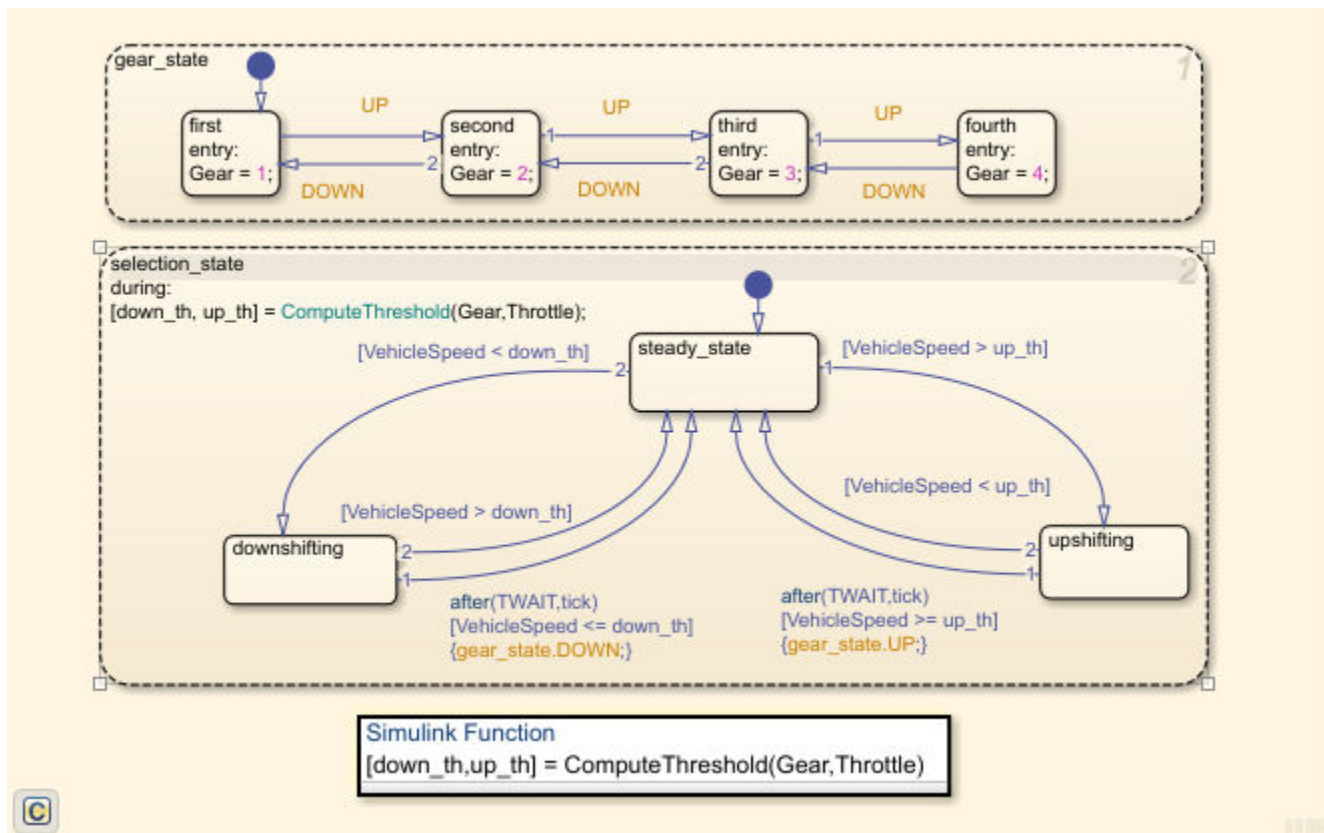
Control the Clutches

To select and engage the appropriate gear set, the model uses a Stateflow block and clutch schedule. To see how these components work, return to the main model of `VehicleWithFourSpeedTransmissionExample`.

State-Controlled Gear Selection

The Stateflow block, which is labeled Shift Logic, implements gear selection for the transmission. The block determines whether to shift up or down based on input from two other components in the model. Driver Inputs block supplies throttle and braking information. The Vehicle Body subsystem supplies the velocity of the vehicle body via a feedback loop.

To open the Stateflow diagram, double-click the Shift Logic block. The Model Explorer is utilized to define the inputs as throttle and vehicle speed and the output as the desired gear number. Two dashed AND states keep track of the gear state and the state of the gear selection process. The overall chart is executed as a discrete-time system. The Stateflow diagram shown in the figure illustrates the functionality of the block.

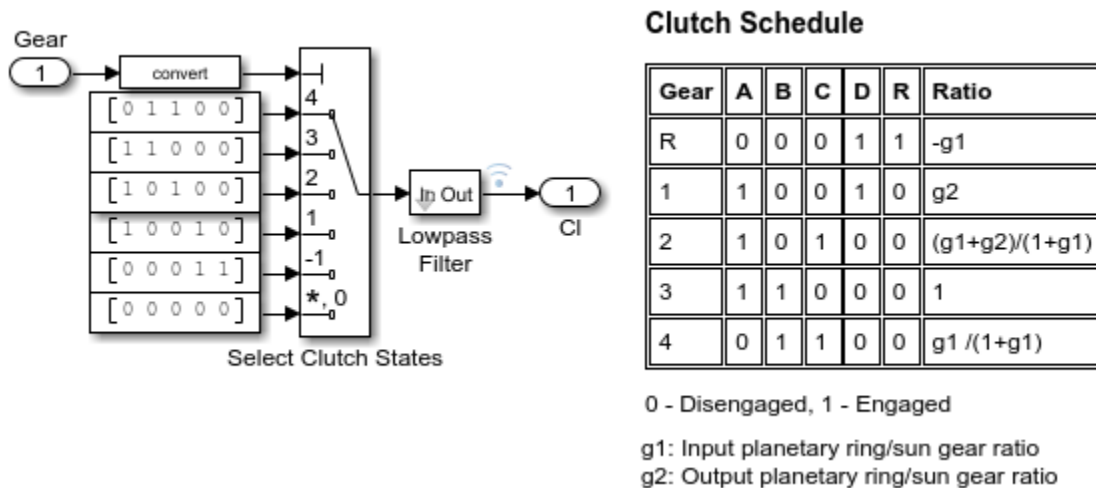


The model computes the upshifting and downshifting speed thresholds as a function of the instantaneous values of gear and throttle. While in `steady_state`, the model compares these values to the present vehicle speed to determine if a shift is required. If so, it enters one of the confirm states (upshifting or downshifting), which records the time of entry.

If the vehicle speed no longer satisfies the shift condition, while in the confirm state, the model ignores the shift and it transitions back to `steady_state`. The steady-state condition prevents extraneous shifts due to noise conditions. If the shift condition remains valid for a duration of `TWAIT` ticks, the model transitions through the lower junction and, depending on the current gear, it broadcasts one of the shift events. The model again activates `steady_state` after a transition through one of the central junctions. The shift event, which is broadcast to the `gear_selection` state, activates a transition to the appropriate new gear. The Stateflow block outputs the gear information to a clutch schedule subsystem that is in the transmission subsystem.

Clutch Schedule Subsystem

The signal from the Stateflow block to the clutch schedule controls the five clutches of the CR-CR 4-Speed transmission. To see the clutch schedule, open the Transmission subsystem, and then the Clutch Schedule subsystem.



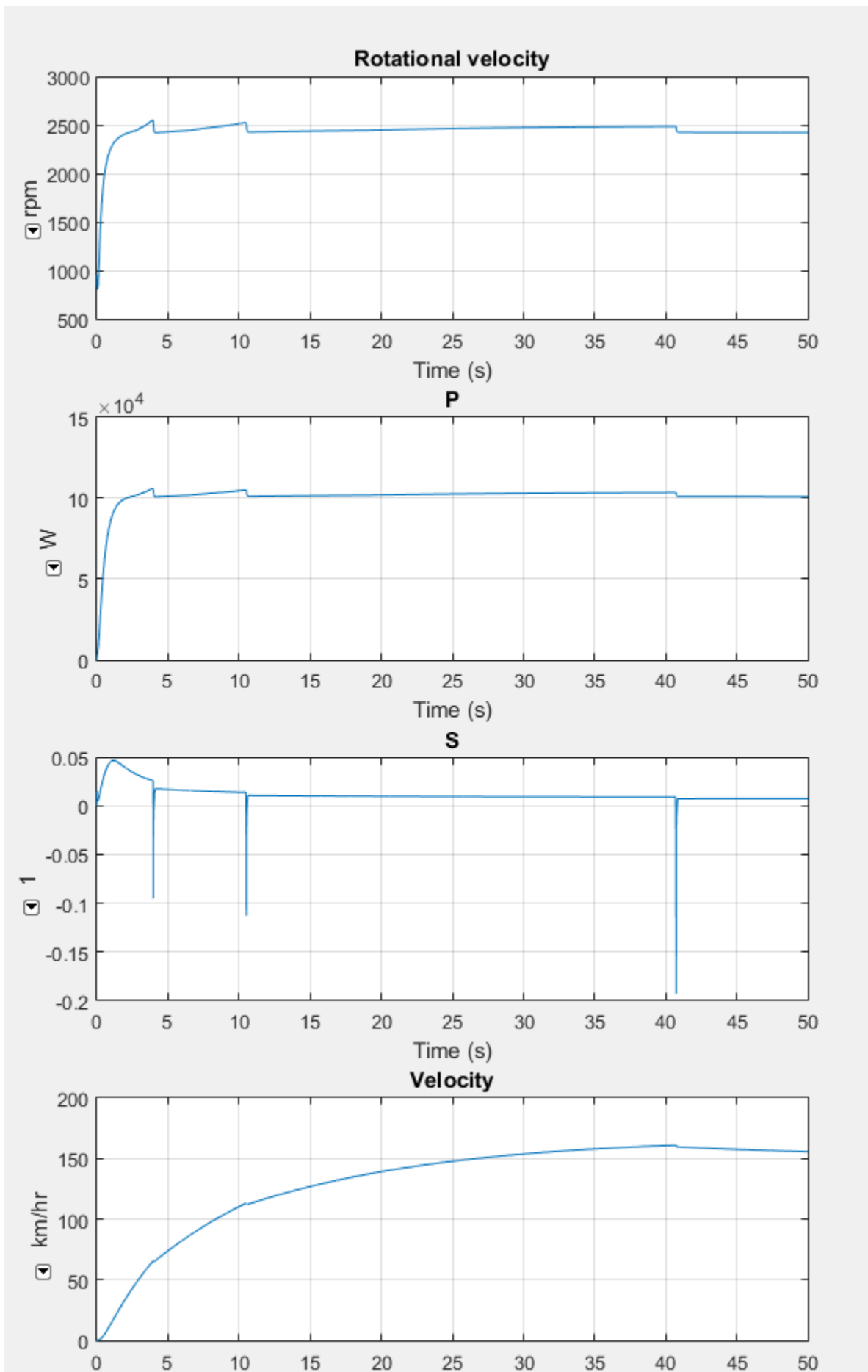
Run the Model

The model is configured to simulate for 50 seconds. The table shows the gear profile for the simulation.

Time Ranges (s)	CR-CR Gear Settings
0-3.96	1
3.96-10.48	2
10.48-40.68	3
40.68-50	4

- 1 Simulate the car.
- 2 To see the results using the Simscape Results Explorer, in the description in the model window, click **Explore simulation results**.
- 3 To plot the rotational velocity in RPMs and power in Watts for the engine:
 - a In the left pane of the Results Explorer window, expand the node for the **Engine**

- b** Click the **F** node, and then the **w** node.
 - c** To change the units for the y-axis to revolutions per minute, click the arrow button below the y-axis label (rad/s) and select rpm.
 - d** To add a plot of the power that the engine delivers to the torque converter, Ctrl+click the **P** node.
- 4** Add a plot of the tire slip.
 - a** Ctrl+click to expand the **Vehicle_body** node.
 - b** Ctrl+click to expand the **Tire_Left** node.
 - c** Ctrl+click the **S** node.
- 5** Add a plot of the vehicle velocity.
 - a** Ctrl+click to expand the second **Vehicle_body** node.
 - b** Ctrl+click the **v** node.
 - c** To change the units to kilometers per hour, click the arrow button below the y-axis label (m/s), select Specify, and for **Specify your unit**, enter km/hr.



The plots show that for:

- Engine speed and power — When the transmission shifts to second gear at 3.96 seconds, the engine reaches its maximum speed and power.
- Tire slip — As the transmission steps into higher gears, the speed ratio rises. The drive ratio falls, and the tire slip decreases. The tire motion more closely approaches ideal (nonslipping) motion at higher speeds.
- Vehicle velocity — The speed increases less with each upshift for gears one, two, and three. The velocity decreases slightly before it starts to stabilize when the car is in fourth gear.

Basic Motion, Torque, and Force Modeling

The purpose of a gear set is to transfer rotational motion and torque at a known ratio from one driveline axis to another. Simscape Driveline allows you to model simple and custom gears for coupling bodies that are rotating on a driveline axis.

The rules that apply to angular gears in relation to rotational motion and torque are analogous to the rules apply to linear gears in relation to translational motion and force.

Couple Rotational Motion with Gears

A gear set consists of two or more meshed gears rotating together at some specified gear ratios. By convention, Simscape Driveline gear ratios are constant. The gear ratios determine how angular velocity and torque are transferred from one driveline component to another.

Gear Coupling Rules

Ideal gears mesh and rotate together at a point of contact without frictional loss or slippage.

The simplest gear coupling consists of two circular gear wheels of radii r_1 and r_2 , spinning with angular velocities ω_1 and ω_2 , respectively, and lying in the same plane. Their connected shafts are parallel and carry torques τ_1 and τ_2 . The *gear ratio* of gear 2 to gear 1 is the ratio of their respective radii: $g_{12} = r_2/r_1$. The power transferred along either shaft is $\omega \cdot \tau$.

The gear coupling is often specified in terms of the number of gear teeth on each gear, N_1 and N_2 . The gear ratio of gear 2 to gear 1 is then $g_{12} = N_2/N_1 = r_2/r_1$.

The fundamental conditions on the simple gear coupling of rotational motion are $\omega_2/\omega_1 = \pm 1/g_{12}$ and $\tau_2/\tau_1 = \pm g_{12}$. That is, the ratio of angular velocities is the reciprocal of the ratio of radii, while the ratio of torques is the ratio of radii. The transferred power, being the product of angular velocity and torque, is the same on either shaft.

The choice of signs indicates that the gears can spin in the same or in opposite directions. If the gears are external to one another (rotating together on their respective outside surfaces), they rotate in opposite directions. If the gears are internal to one another (rotating together with the outside of the smaller gear meshing with inside of the larger gear), they rotate in the same direction.

Caution Gear ratios in driveline model blocks must be strictly positive. Vanishing or negative gear ratios cause Simscape Driveline simulation to stop with an error at model initialization. If you need to reverse the relative rotation direction of a shaft connected to a gear, you can change the direction in the gear block property inspector.

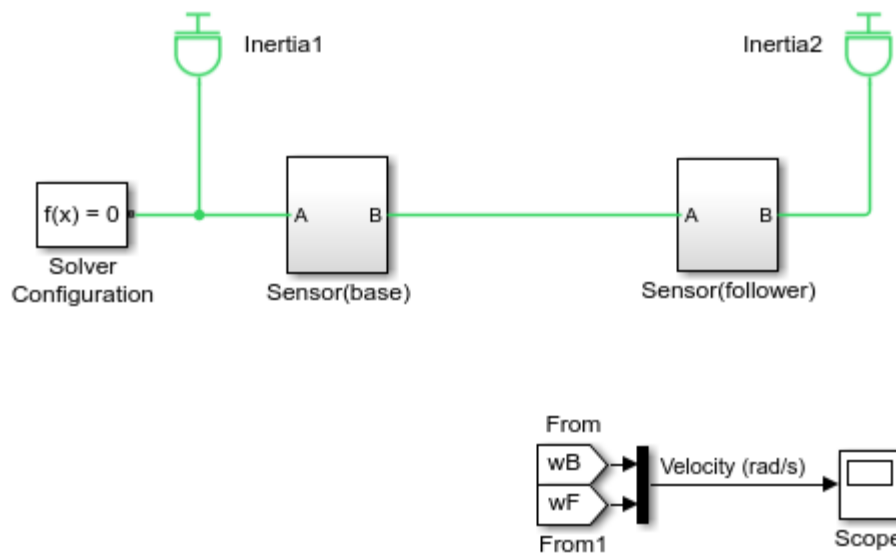
Couple Two Spinning Inertias with a Simple Gear

In this example, you couple two spinning inertias. In the first coupling, the inertias spin with the same angular velocity along a single shaft (driveline axis). Then the inertias spin at different velocities as they spin along two shafts and are coupled by a gear. Finally, the inertias are coupled by a gear and actuated by an external torque, so that they spin at different rates and experience different torques. For each model, the example uses basic Simscape mechanical and Simscape Driveline blocks, such as Inertia, Simple Gear, and Solver Configuration.

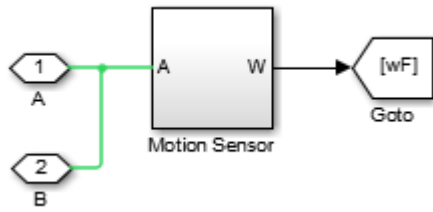
Modeling Two Spinning Inertias

Create the first version of the simplest, nontrivial driveline model, two inertias spinning together along the same axis. Open the Simscape Driveline, Simscape, and Simulink block libraries and a new Simulink model window.

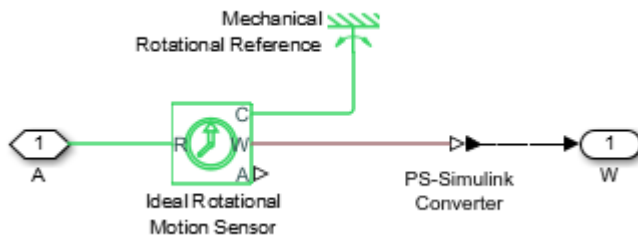
- 1 Drag and drop two Inertia, two Ideal Rotational Motion Sensor, two Mechanical Rotational Reference, and two PS-Simulink Converter blocks into the model window.
- 2 From the Simscape Utilities library, drag a Solver Configuration block. Every topologically distinct driveline block diagram requires exactly one instance of this block.
- 3 From the Simulink library, drag and drop a Scope, a Mux, and two pairs of Goto and From blocks. Connect the blocks as shown in the following figures. The sensor subsystems are arranged hierarchically.



Model with Two Spinning Inertias

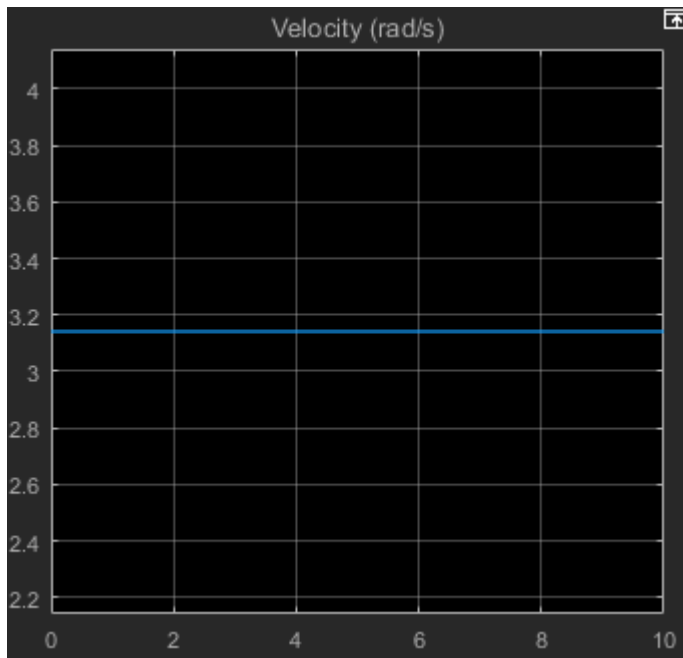


Sensor Subsystem



Motion Sensor Subsystem

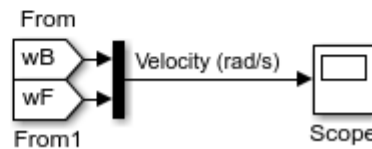
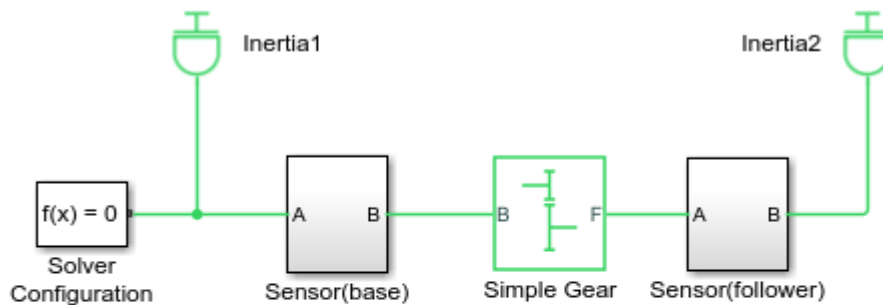
- 4 At the start of the simulation, because there is no damping, the inertias rotate at the initial velocity that you specify. The connection line between the two Inertia blocks requires them to have the same rotational velocity. To specify the initial rotational velocity, open each Inertia block. In the **Variables** tab, select the **Rotational velocity** check box and set the **Value** parameter to pi radians/second (rad/s).
- 5 Open the Scope block and start the simulation. The two angular velocities are constant at 3.14 radians/second.



Coupling Two Spinning Inertias with a Simple Gear

Modify the model you created by coupling the two spinning inertias with a simple, ideal gear with a fixed gear ratio.

- 1 From the Simscape Driveline block library, drag and drop a Simple Gear block into your model. Open the block. Change the default follower-base gear ratio value to 1. Change the **Output shaft rotates** menu to **In same direction as input shaft** and click **OK**. The simple gear then represents two gear wheels rotating together at the same rate in the same direction, with one wheel inside the other. Connect the blocks as shown in the following figure.



Model with Two Spinning Inertias Coupled by a Gear

Leave the initial angular velocities at π in the Inertia blocks.

- 2 Open the Scope and start the simulation. The two angular velocities are constant at 3.14 radians/second for both Inertias.
- 3 Change the **Output shaft rotates** menu back to **In opposite direction to input shaft**. The simple gear then becomes two wheels rotating together in opposite directions, with the two wheels meshed on their respective outer surfaces. Change initial velocity in Inertia2 to $-\pi$.
- 4 Restart the simulation. The two angular velocities are 3.14 and -3.14 radians/second for Inertia1 and Inertia2, respectively. The second angular velocity is the same, but with opposite sign, because the two bodies are spinning in opposite directions.
- 5 Change the **Output shaft rotates** menu again to **In same direction as input shaft**.

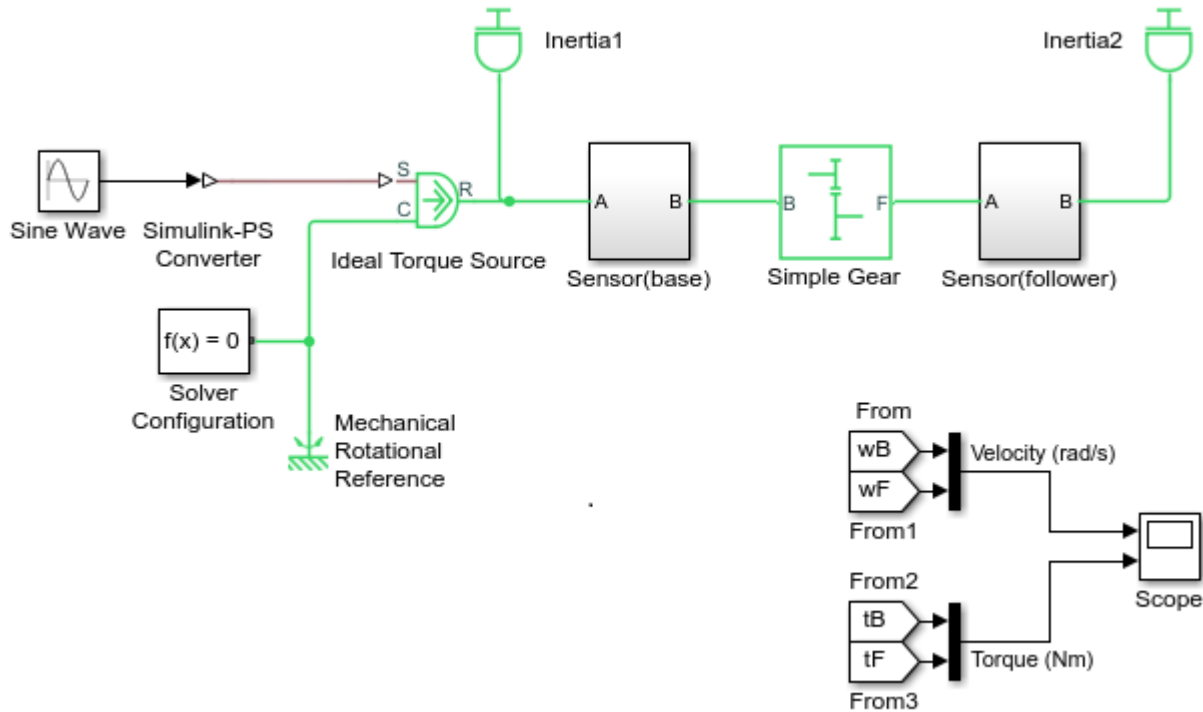
Torque-Actuating Two Coupled, Spinning Inertias

In the final version of the simple gear model, you actuate the inertias with an external torque instead of starting them with fixed initial angular velocities. The external torque varies sinusoidally. You can find a completed version of this model in the `SimpleGearExample` model.

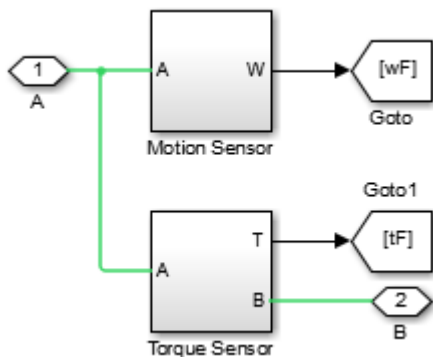
- 1 From the Simscape Foundation library, copy an Ideal Torque Source and two Ideal Torque Sensor blocks, plus a Simulink-PS Converter block and another Mechanical Rotational Reference block.

From the Simulink library, drag and drop a Sine Wave block and two more pairs of Goto and From blocks.

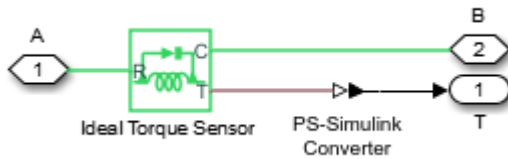
- 2 Connect the blocks as shown in the following figures. The Torque Sensor subsystems are arranged in parallel with the Motion Sensor subsystems inside the Sensor subsystem blocks. Set the initial velocities of both Inertias to zero. Change the default follower-base gear ratio value to 2. Modify the Scope block to add another axis for measuring the torques. Connect the other blocks as shown.



Model with Two Spinning Inertias Coupled by a Gear and Actuated with Torque



Updated Sensor Subsystem



Torque Sensor Subsystem



- 3 Open the Scope block and start the simulation.

The measured torques and angular velocities vary sinusoidally. As in the preceding models, the angular velocity of Inertia2 is half that of Inertia1. The torque in the second (follower) shaft is twice that in the first, as required by the laws of gear coupling.

For the Simple Gear block, change the **Output shaft rotates** menu to **In opposite direction to input shaft** and restart the simulation. The same angular velocities and torques result, except that the values associated with Inertia2 and the second shaft are negative because the second body and second shaft are spinning in opposite directions.

Sensing and Actuating Motion and Torque

The mechanical sensor and source blocks that you use in the preceding models illustrate their dual nature. They act as driveline components themselves, but also let you inject and extract physical signals associated with motion and torque, including the appropriate physical units. You can use these physical signals with other blocks in the Simscape physical modeling environment, or convert them to dimensionless Simulink signals for use in the nonphysical part of your model. Both sensor and source blocks have pairs of mechanical ports and are connected either in series with or across physical connection lines.

- Mechanical sensor and source blocks have both mechanical conserving ports  and physical signal ports .

Many Simscape Driveline blocks also feature a mix of mechanical conserving and physical signal ports.

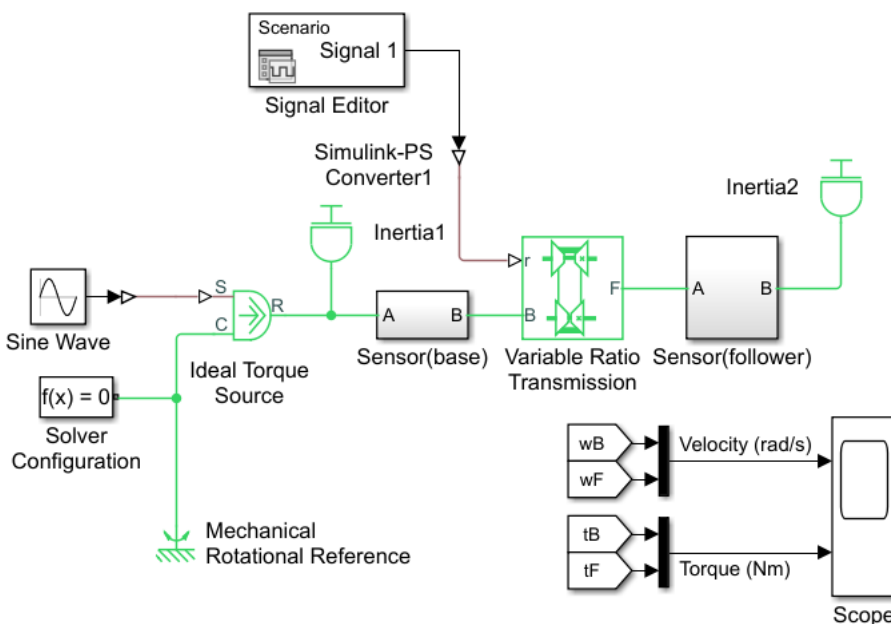
- An Ideal Torque Source injects torque along, or in series with, the driveline connection line. An Ideal Torque Sensor measures the torque flowing along, or in series with, the driveline connection line.
- An Ideal Rotational Motion Sensor reports the *difference* between the motions at its two connection ports.

To extract the absolute motion at its R port, connect the C port to a mechanical reference block that grounds that port to zero motion.

Couple Two Spinning Inertias with a Variable Ratio Transmission

You can modify the gear model from the “Couple Two Spinning Inertias with a Simple Gear” on page 4-3 example by replacing the fixed-ratio gear with a transmission whose gear ratio varies in time. You specify the gear ratio variation with a Simulink signal converted to a unitless physical signal. Start with the gear model that you built in the “Couple Two Spinning Inertias with a Simple Gear” on page 4-3 example or by opening and editing the SimpleGearExample model.

- 1 From the Simscape Driveline block library, drag and drop a Variable Ratio Transmission block and replace the Simple Gear block with it. The two shafts spin in the same direction because the **Output shaft rotates** parameter for the Variable Ratio Transmission block is set to **In same direction as input** (the default setting).
- 2 The Variable Ratio Transmission block accepts the continuously varying gear ratio as a physical signal Simulink signal through the extra physical signal input labeled *r*. For this example, create a variable signal for the gear ratio with a Signal Editor block from the Simulink block library and Simulink-PS Converter block. Build a signal that rises with constant slope from 1 to 2 over 10 seconds. Then connect the converted physical signal to the *r* port.



Simple Variable Ratio Transmission Model

- 3 Do not change the other, original settings of the simple gear model. Open the Scope and start the simulation.

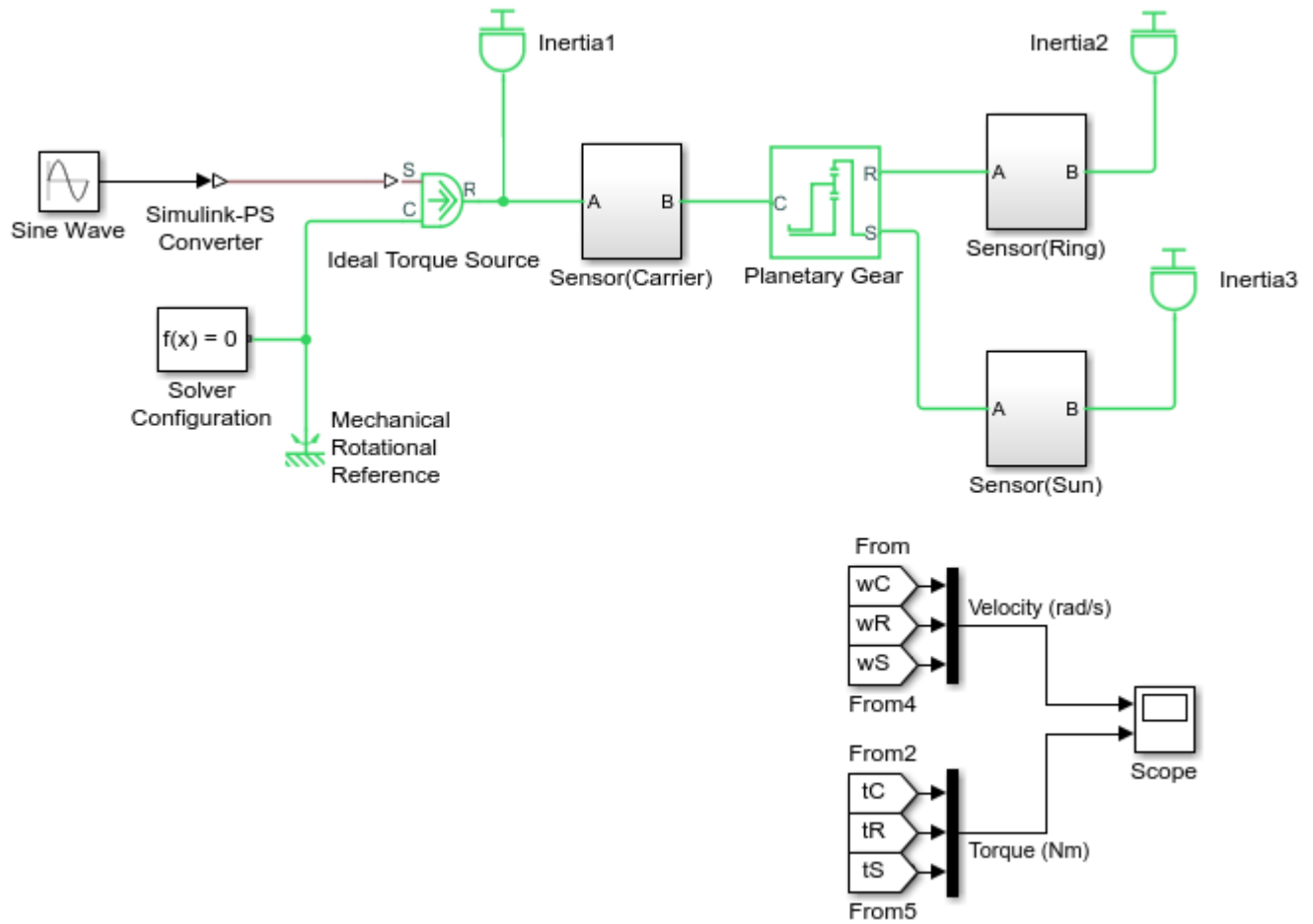
The angular velocities and torques of the two shafts have the same signs. The ratios of angular velocities and torques start at 1, because the initial gear ratio is 1. As the gear ratio increases toward 2, the angular velocity of Inertia2 becomes smaller than the velocity of Inertia1, while the associated torque in the second shaft becomes larger than the torque in the first shaft. Because of the changing gear ratio, the motion and the torques are no longer strictly sinusoidal, even though the actuating external torque is.

The “Variable Ratio Gear” on page 18-145 example is a full model of this type. To learn more about how to use variable gear ratios, consult the Variable Ratio Transmission block reference page.

Couple Three Spinning Inertias with a Planetary Gear

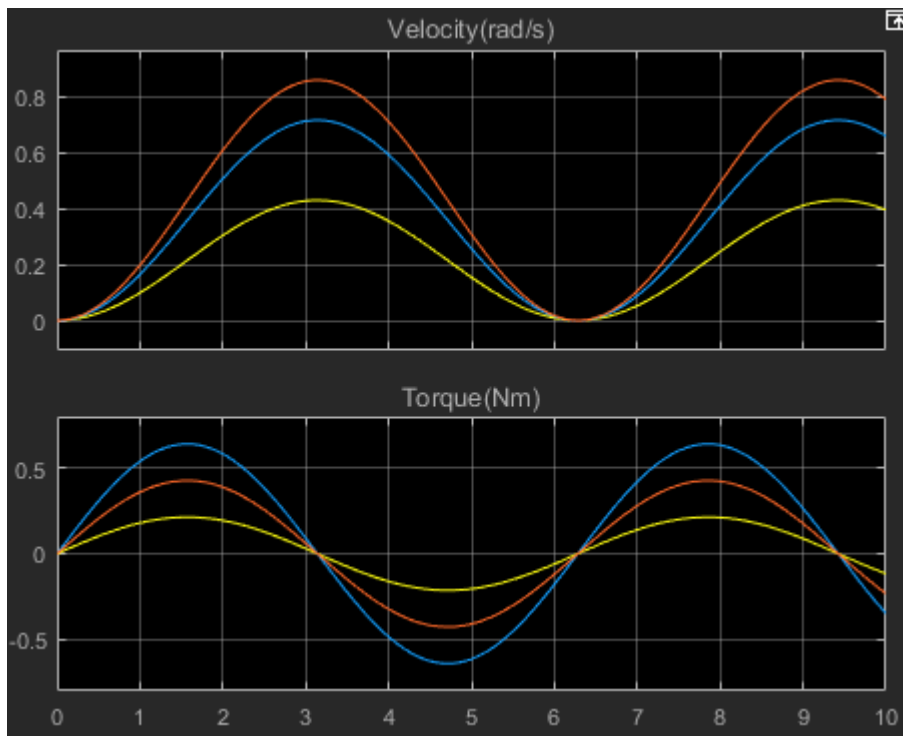
You can modify the gear model from the “Couple Two Spinning Inertias with a Simple Gear” on page 4-3 example and use it as a starting point for studying complex gear sets. One of the most important complex gear sets is the planetary gear, which has three wheels, the ring, the sun, and the planet, all held in place by a common carrier body. The planetary gear is important because it is a common component in complex, realistic transmissions.

- 1** Start with the simple gear model you built or by opening the `SimpleGear` model.
- 2** Replace the Simple Gear in your the model with a Planetary Gear from the Simscape Driveline block library. A planetary gear splits input angular motion from the carrier between the ring and sun wheels, each connected to their respective bodies.
- 3** Copy the Sensor (Follower) subsystem, the connected Inertia block, and a From block.
- 4** Rename the sensor subsystems to match the gears that they attach to: carrier, ring, and sun. Rename the signals on the Goto and From block tags to match the gears that the signals represent:
 - For the carrier gear, rename the tags signals as w_C and t_C .
 - For the ring gear, rename the tags signals as w_R and t_R .
 - For the sun gear, rename the tags signals as w_S and t_S .
- 5** Add an input to each of the two Mux blocks.
- 6** Connect the blocks to form the new diagram as shown in the figure.

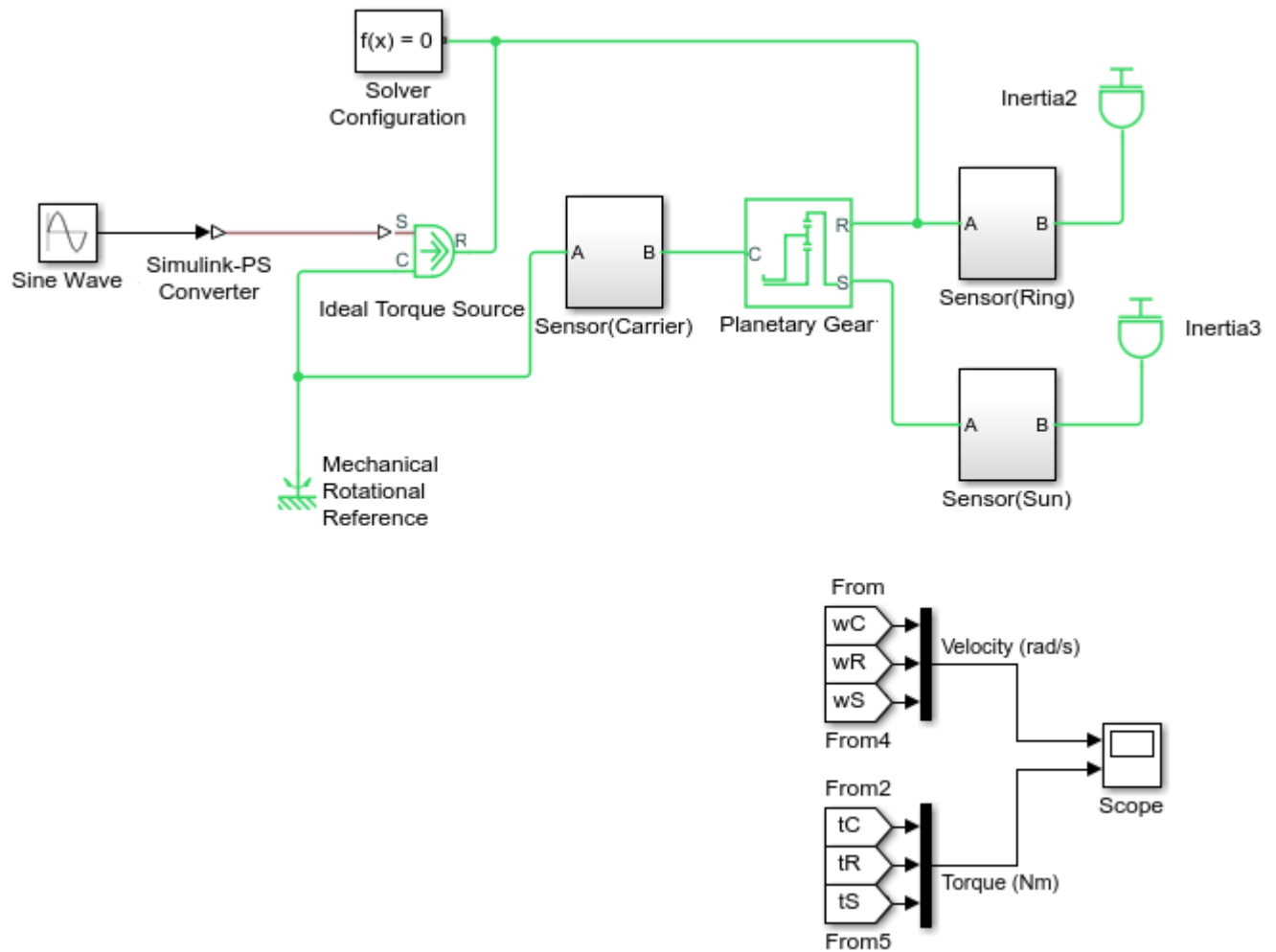


Simple Planetary Gear Model

- 7 Open the Scope and start the simulation to observe the angular velocities of the ring, carrier, and sun, from largest to smallest. The ratio of the ring to sun gear velocities is always 2.

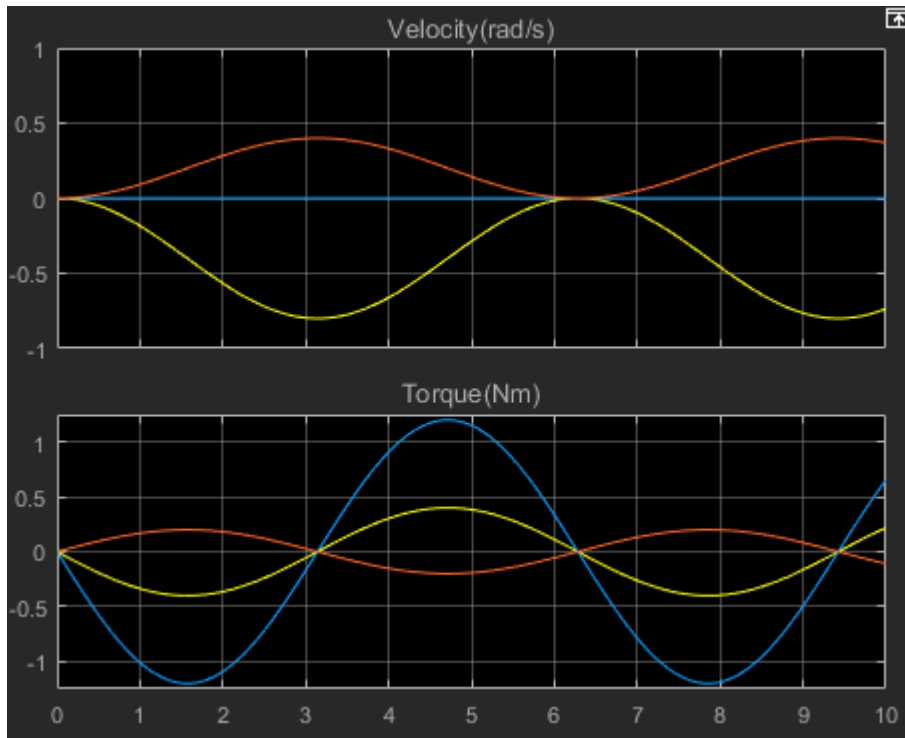


- 8 To see the ring and sun wheels spinning alone, you must lock the carrier. To do so:
 - 1 Delete the Inertia1 block and the associated connector.
 - 2 Delete the connector between the Ideal Torque Source block and the Sensor (Carrier) subsystem.
 - 3 Connect the Sensor (Carrier) subsystem to the connector between the Mechanical Rotational Reference block and the Ideal Torque Source block.
 - 4 Connect the Ideal Torque Source block to the connector between the Planetary Gear block and the Sensor (Ring) subsystem.
 - 5 Delete the connector that connects the Solver Configuration block to the connector between the Mechanical Rotational Reference block and the Ideal Torque Source block.
 - 6 Reposition the Solver Configuration block.
 - 7 Connect the Solver Configuration block to the connector between the Ideal Torque Source block and the Sensor (Ring) subsystem.



Simple Planetary Gear Model with Locked Carrier

- 9 Open the Scope and start your model. Observe the angular velocities of the ring, carrier, and sun.



The carrier, connected to Mechanical Rotational Reference, does not move. The ring is driven with a sinusoidal torque, and the sun responds by spinning in the opposite direction (ring and sun gear wheels are external to one another) at twice the rate. The ring wheel has twice the radius (or twice the number of teeth) as the sun, so it spins half as fast.

Driveline Actuation

From the torques and forces applied to driveline inertias and masses, a Simscape Driveline simulation determines the resulting motion from the driveline component connections and defining equations. However, a simulation can also accept motions imposed on a driveline and solve for the torques and forces to produce those motions. In general, a driveline simulation is a mixture of these two requirements, solving dynamics both forward (torque and force to motion) and inverse (motion to torque and force). Imposing motions and applying torques and forces to a driveline are together forms of mechanical *actuation*.

Best Practices for Modeling Torque-Force Actuation and Motion Actuation

All actuation, except for initial conditions, requires physical signal inputs to define time-varying functions that carry physical units.

Torque-force actuation and motion actuation are complementary and mutually exclusive. In all cases, exercise care as you apply a mixture of actuation types to a driveline and its degrees of freedom (DoFs). The complete effect of the actuation types must be such that:

- Driveline DoFs actuated by torques and forces are not also subject to motion actuation. (They can be subject to motion initial condition settings.)
- Driveline DoFs actuated by motions are not also subject to torque or force actuation.

For a Simscape Driveline model to simulate nontrivial motion, torque and motion actuation types complement one another exactly to account consistently for the motion of all the DoFs. If this criterion is not satisfied, one of these outcomes results:

- The motion of the driveline is trivial, staying in its initial motion state for the entire simulation.
- The actuation types are inconsistent with each other, and the simulation stops with an error.
- The actuation types leave the driveline motion underdetermined or overdetermined, and the simulation stops with an error.

For more information, see “Troubleshoot Driveline Modeling and Simulation Issues” on page 17-2.

Actuate a Driveline Using Torques and Forces

You can apply a torque to a rotational driveshaft, or a force to a translational driveshaft, in the following ways:

- Directly, with an Ideal Torque Source or Ideal Force Source block.
- Indirectly, with a dynamic element that generates torque or force. Such blocks include torque converters, clutches and clutch-like elements, and engines.

A torque or force source accepts a physical signal input and originates, from its mechanical conserving port, a mechanical connection line carrying that torque or force.

The Simscape Driveline simulation solves for the motion of the spinning or sliding driveshaft, given the torque or force that it is subject to. Therefore you cannot also subject that same driveshaft to motion actuation.

Note Simscape Driveline might generate an error if the combined torques and forces at any given node have a nonzero sum.

Actuate a Driveline Using Motions

You can apply a motion to a driveshaft directly, with an Ideal Angular Velocity Source or Ideal Translational Velocity Source block.

A motion source accepts a physical signal input and originates, from its mechanical conserving port, a mechanical connection line spinning or sliding with the specified motion.

The Simscape Driveline simulation solves for the torque or force carried by the spinning or sliding driveshaft, given its motion. Therefore you cannot also subject that same driveshaft to torque or force actuation.

Set Initial Conditions of Driveline Motion

When driveline simulation starts, the complete driveline determines the initial motion of all driveshafts by a combination of constraints, motion sources, and initial condition settings. You set the initial conditions for the rotational and translational motion of inertias and masses in their respective Inertia and Mass blocks. The block default for initial velocities is zero (no initial motion).

For more information about constraints and degrees of freedom, see “Driveline Degrees of Freedom” on page 16-21.

Note Ensure that the initial conditions that you impose on the Inertia and Mass blocks in your driveline are consistent with all of the constraints and motion sources for the driveline. If an inconsistency occurs, Simscape Driveline simulation stops with an error at model initialization.

Resolving Undetermined Motions in Complex Gears

A simple gear has two ports and imposes one constraint between them, leaving one independent DoF. Once one port is connected to a driveshaft, the motion of the driveshaft of the other port is determined.

A complex gear has three or more ports and imposes one or more constraints among them. A complex gear can have any number of independent DoFs, including none.

If a simulation apportions the initial motions of a complex gear in an unsatisfactory way, enforce your preferred division by setting initial conditions on the connected Inertia and Mass blocks.

However you divide the initial motion among the gear shafts, ensure that this division is consistent with all constraints in your driveline, and with any motion sources.

For more information about complex gears, see “Basic Motion, Torque, and Force Modeling” .

Power Transmission Using Pulley Networks

The Simscape Driveline Couplings and Drives library includes pulley blocks that allow you to transmit power. The Belt Pulley block represents a simple belt-driven friction pulley. The Belt Drive block combines two Belt Pulley blocks that you can configure as an open or crossed belt system. You can model a simple hoist using the Belt Pulley and two Rope blocks. To model more complex mechanisms, combine multiple Belt Pulley and Rope blocks to form a representative network. Build high-fidelity models by including other Simscape and Simscape Driveline blocks.

- “Best Practices for Modeling Pulley Networks” on page 6-2
- “Verify Pulley Configuration in Power Window System” on page 6-7

Best Practices for Modeling Pulley Networks

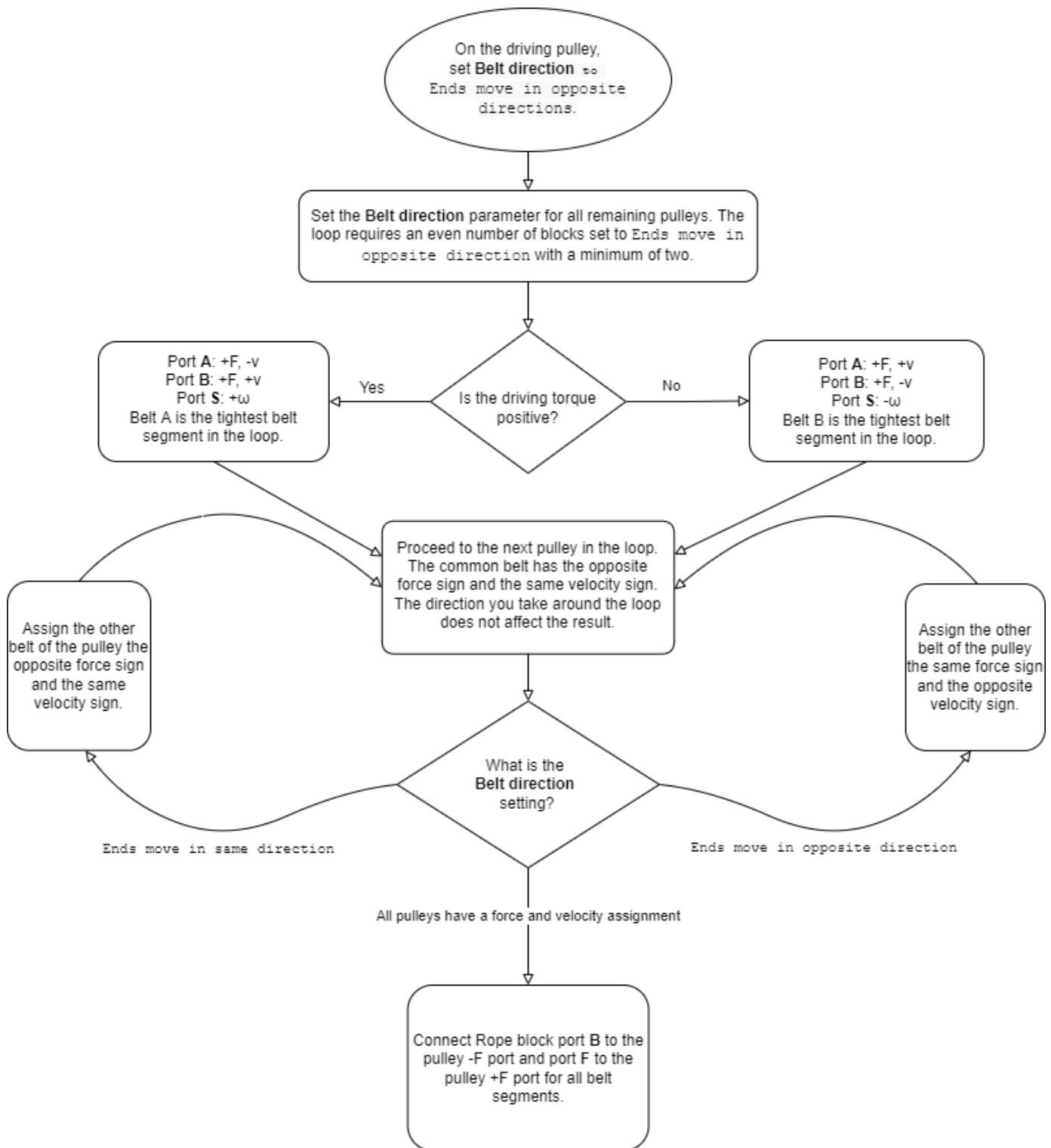
In this section...
“Belt Direction” on page 6-4
“Rope Block Tension” on page 6-5
“Inertia” on page 6-5

A closed-network pulley system relies on belt tension to convey forces throughout the network. The weight of the belt adds inertia to the system when the belt is in motion. In Simscape, you can model pulley networks with the Belt Pulley block, which uses belt tension and inertia to simulate motion and the distribution of forces. Like a physical pulley, the block requires tension at both ends when operating. However, because the block has inherent directionality, you must align the positive and negative velocity ports with each other and correctly orient each block.

To initialize your system and orient the blocks:

- Set the **Belt direction** for each Belt Pulley block in the loop to obtain a non-trivial solution.
- Obtain a physically accurate direction of rotational velocity, ω , for a given pulley.
- Orient the Rope blocks between pulleys to transfer the correct tension.

This flow chart shows an overview of the process. The flow chart assumes the pulley network has one torque source, the driving pulley.

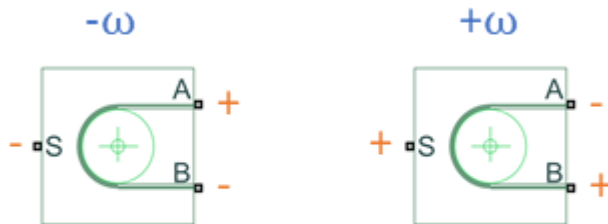


You cannot specify a rotational velocity or torque for each pulley in the network. By tracking the force and velocity direction for each Belt Pulley and Rope block in the network, you can ensure that your model generates physical results.

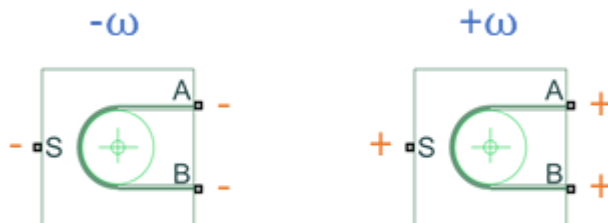
The “Power Window System” on page 18-88 example shows a detailed pulley network model that you can adjust.

Belt Direction

Belt direction is a pulley sign convention that is not constrained by physics or geometry. For a Belt Pulley block with the **Belt direction** parameter set to *Ends move in opposite direction*, the velocities at port **A** and port **B** have opposite signs that depend on the sign of the angular velocity, ω , at port **S**.



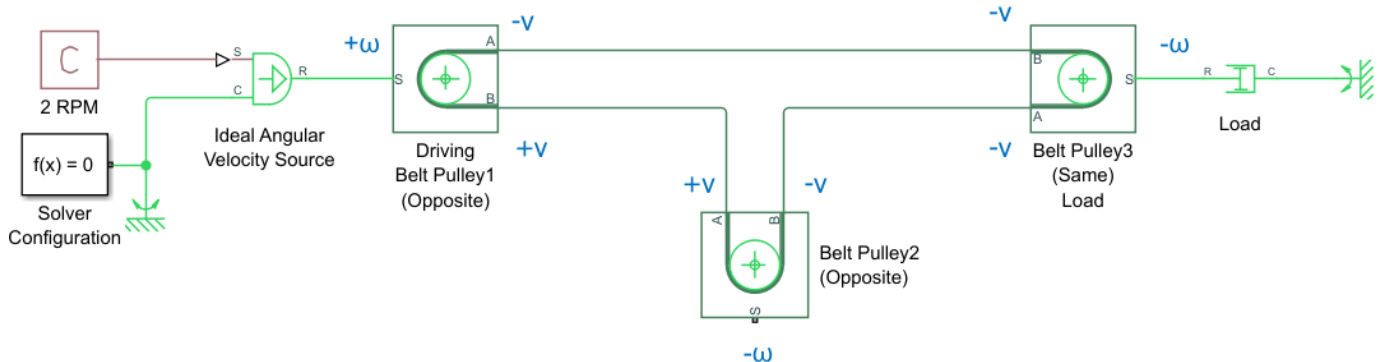
For a Belt Pulley block with **Belt direction** set to *Ends move in same direction*, the velocities at port **A** and port **B** have the same signs. The signs of the ports depend on the sign of the angular velocity at port **S**.



To change the sign of the rotational velocity for a Belt Pulley block, change the setting of the **Belt direction** parameter.

Belt Direction in Closed Pulley Networks

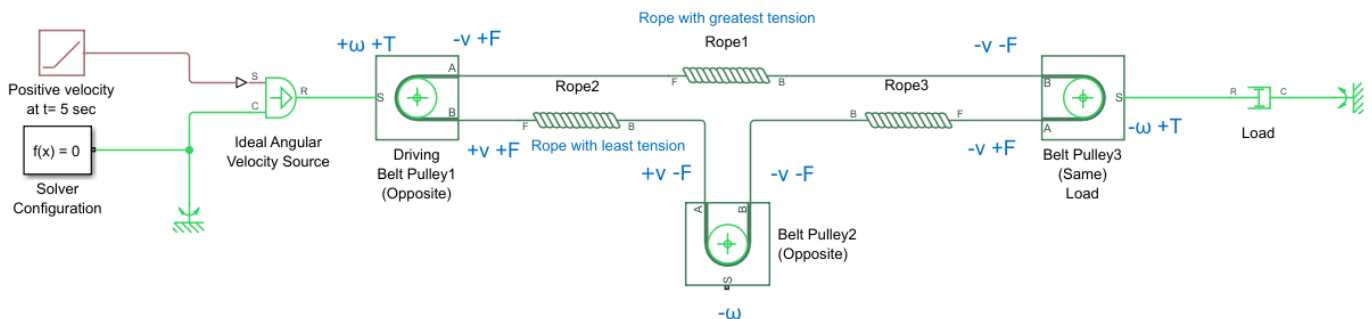
To obtain a nonzero response in a closed-loop system with two pulleys, set **Belt direction** to *Ends move in opposite direction*. When dealing with three or more pulleys, ensure the loop has an even number of Belt Pulley blocks with the **Belt direction** parameter set to *Ends move in opposite direction*. To check whether the **Belt direction** parameter is responsible for trivial simulation results, choose *Ends move in same direction*. If there are no problems with the pulley network, the simulation generates non-trivial results when you complete this step. The figure demonstrates a pulley loop with an even number of Belt Pulley blocks using the *Ends move in opposite direction* setting.



Rope Block Tension

Maintain belt contact by including Rope blocks in your pulley system. You need at least one fewer Rope blocks than the number of pulley pairs, at a minimum. For example, if there are five pulley pairs, include at least four Rope blocks. You can confirm that the belts start and remain in tension by using the **Simscape Results Explorer**. To configure your system such that all Rope blocks have positive, stable tension at the start of simulation:

- Determine the sign of the force for each node of each Belt Pulley block.
- Add Rope blocks between each Belt Pulley block. Connect port **F** of the Rope block to the positive force end of the Belt Pulley block.
- Compare your results for rope tension with your expectation for a real system.



Inertia

To facilitate motion, include inertia in the pulley system. You can include inertia in a Belt Pulley block by specifying a nonzero value for the **Inertia** parameter in the block configuration settings. Another way to include inertia is to add a downstream inertia source from the Simscape Driveline Inertias and Loads library or from the Rotational Elements library. Attribute some initial velocity to the inertia, as needed, to initiate motion in your pulley system.

You can also add mass to the tensioning device to aid with numerical initialization. If you are using the Rope block as a tensioning device, set **Model mass** to 0n.

See Also

Belt Drive | Belt Pulley | Rope | Rope Drum

Related Examples

- “Power Window System” on page 18-88
- “Compound Pulley” on page 18-10

More About

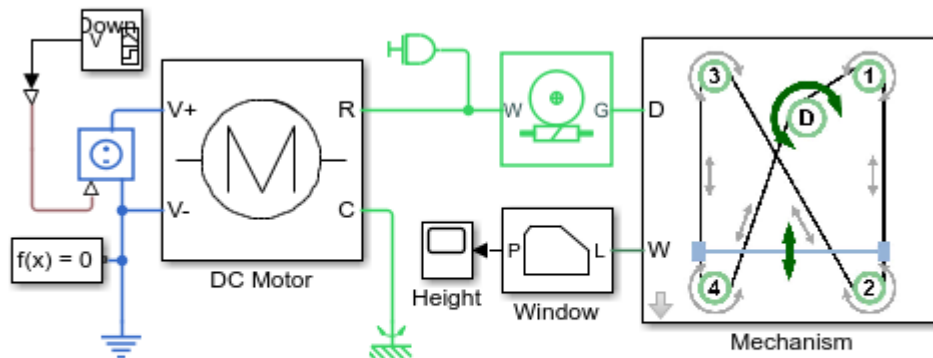
- “Troubleshoot Pulley Network Issues” on page 17-6

Verify Pulley Configuration in Power Window System

The Simscape Driveline power window example contains a pulley network that uses tension and inertia and follows recommended belt-direction practices.

- 1 To open the model, at the MATLAB command prompt, enter

```
sdl_power_window
```

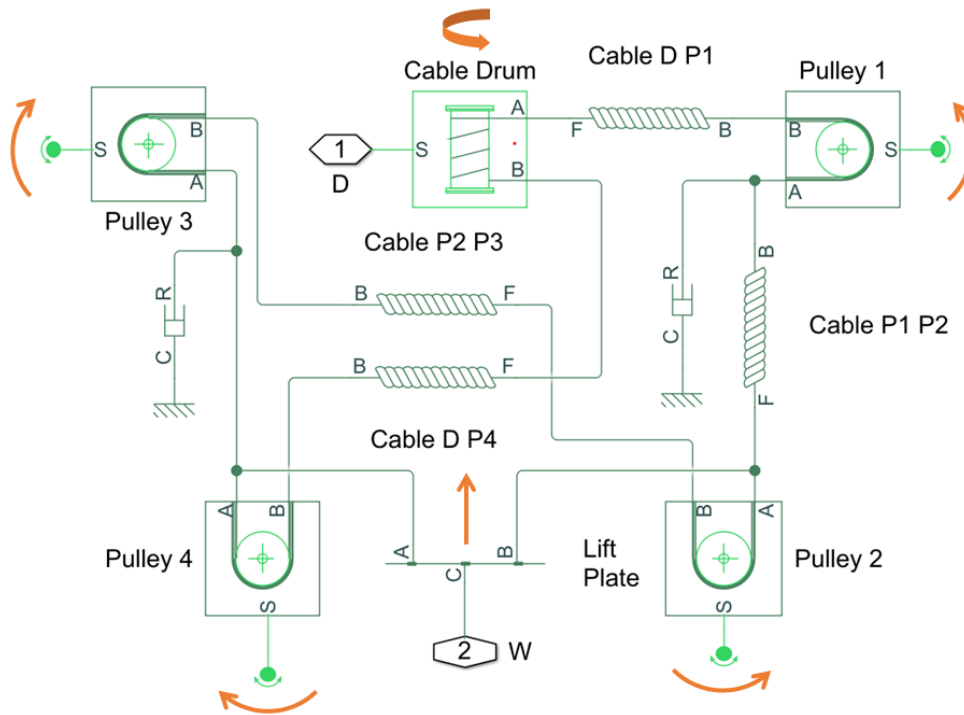


Power Window System

1. [Plot motor torque](#) and cable drum speed ([see code](#))
2. [Explore simulation results](#) using [sscexplore](#)
3. [Learn more](#) about this example

The model contains the **Mechanism** subsystem, a masked pulley network subsystem. The DC Motor subsystem and a Worm Gear block work together to initiate motion in the pulley system. The system also contains an Inertia block.

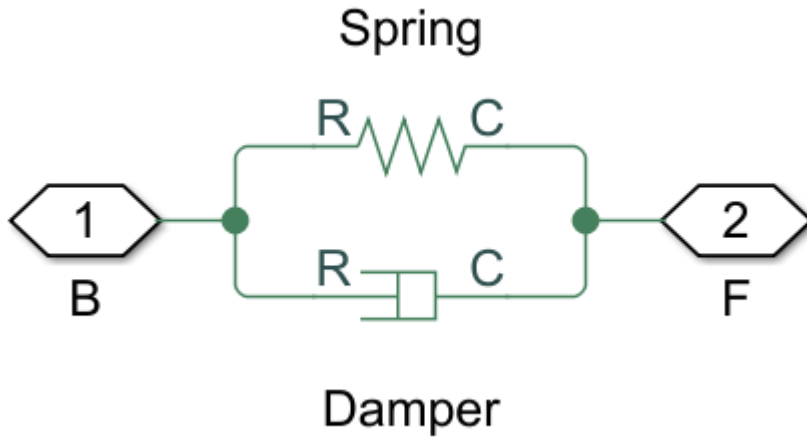
- 2 To look inside the mask of the **Mechanism** subsystem, click the arrow in the lower-left corner of the block.



The arrows show how the four Belt Pulley blocks rotate in response to the rotation of the **Cable Drum** block. If the drum rotates in the opposite direction, the pulley directions reverse, and the **Lift Plate** lowers. There are six pulley pairs:

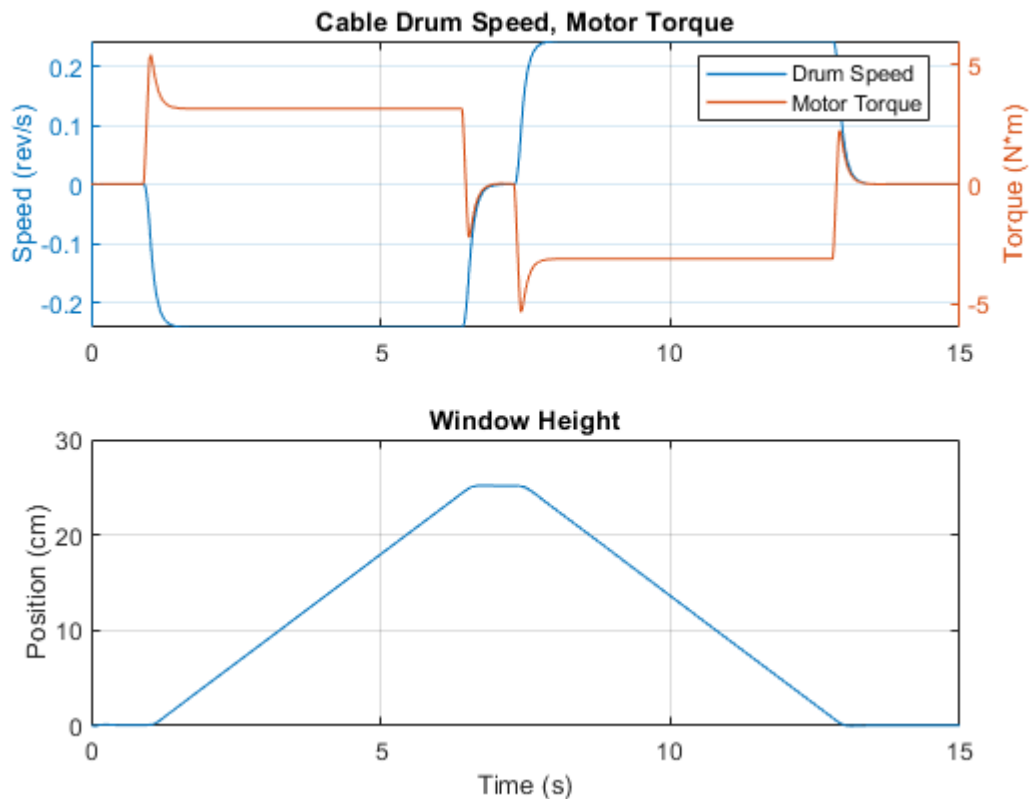
- **Cable Drum** and **Pulley 1**
- **Pulley 1** and **Pulley 2**
- **Pulley 2** and **Pulley 3**
- **Pulley 3** and **Pulley 4**
- **Pulley 4** and **Cable Drum**
- **Pulley 2** and **Pulley 4**

You need at least five Rope blocks to build this system. The **Lift Plate** acts as a tensioner for the **Pulley 2** and **Pulley 4** pulley pair. The system contains four additional Rope blocks. In the figure, you can see how these blocks are implemented in the system.




Each Rope block contains a spring and damper network separating the base port from the follower port.

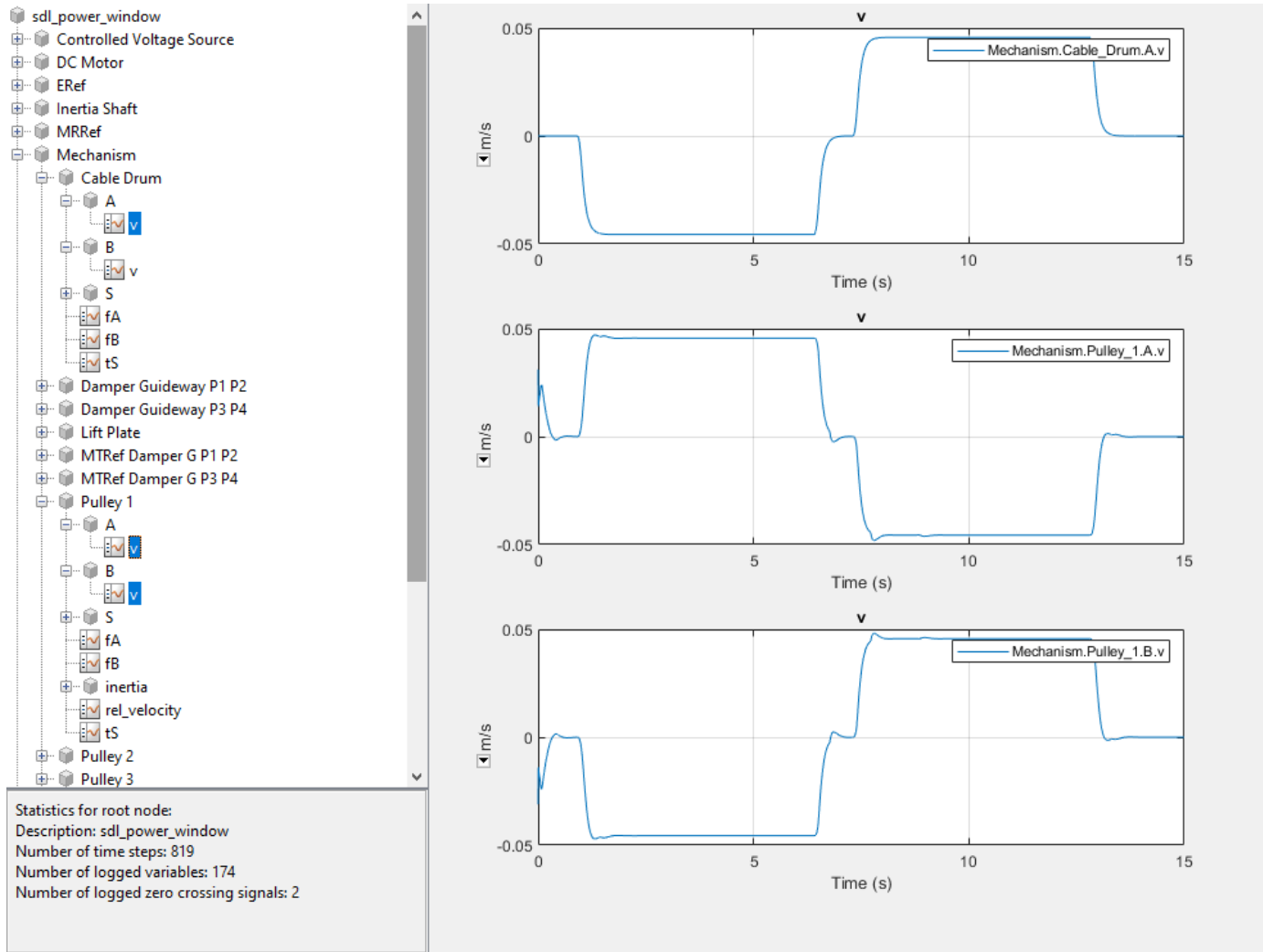
- 3 Run the simulation and plot the results by clicking the **Plot motor torque** link in the model canvas. When the **Cable Drum** has a negative angular velocity, the **Lift Plate** raises the window. When the **Cable Drum** has a positive velocity, the **Lift Plate** lowers the window.



- 4 Open the Results Explorer by clicking the **Explore simulation results** link in the model canvas.

On the Results Explorer toolbar, click the settings button  and, for the **Plot signals** parameter, select **Separate**. Use **Ctrl+click** to open plots for:

- **Mechanism > Cable Drum > A > v**
- **Pulley 1 > A > v**
- **Pulley 1 > B > v**



As expected, the velocity of the drum belt end at port **A** matches the velocity of the **Pulley 1** belt end at port **B** and is the opposite of the velocity of the **Pulley 1** belt end at port **A**.

See Also

Belt Drive | Belt Pulley | Rope | Rope Drum | Shock Absorber | Rotational Free End | Inertia | Variable Inertia | Worm Gear | Translational Damper | Mechanical Translational Reference

Related Examples

- “Power Window System” on page 18-88
- “Compound Pulley” on page 18-10

More About

- “Troubleshoot Pulley Network Issues” on page 17-6

Gear Coupling Control Using Clutches

- “How a Clutch Works” on page 7-2
- “Model Friction Clutches at a Fundamental Level” on page 7-3
- “Engage and Disengage Gears Using a Clutch” on page 7-4
- “Brake Motion Using Clutches” on page 7-7

An important requirement of a practical drivetrain is the ability to transfer rotational motion and torque among spinning components at different speeds and gear ratios. In general, a single set of gears is not sufficient to accomplish this transfer. Clutches allow the drivetrain to transfer motion, torque, and force at different gear ratios under manual or automatic control.

How a Clutch Works

A clutch makes two shafts spinning at different rates spin at a single rate by applying torques that tend to accelerate one shaft and decelerate the other. The most common way for a clutch to accomplish this action is with surface friction. Such a clutch can operate in one of these modes of motion:

- *Disengaged*: the clutch applies no friction at all.
- *Engaged but unlocked*: the clutch applies kinetic friction, and the two shafts spin at different rates.
- *Engaged and locked*: the clutch applies static friction, and the two shafts spin together.

A clutch consists of mated frictional surfaces overlapping one another and connected on either side to a shaft. If the clutch is disengaged, the frictional surfaces have no contact and the shafts spin independently. To engage the clutch, contact between two surfaces is induced by applying pressure normal to the clutch surfaces. The two surfaces in contact and moving relative to one another experience kinetic friction, which causes them to narrow their relative velocity. The friction acts to reduce the relative motion between the two clutch plates and their connected shafts. At some critical combination of reduced relative speed and pressure (normal force), the clutch locks, so that the two shafts are spinning at the same rate. The shafts remain locked together as long as the transmitted torque remains less than the static friction, which is proportional to the applied normal force. If the clutch unlocks but is still engaged, it again applies kinetic rather than static friction.

The transition between unlocked and locked states is discontinuous. Modeling a clutch locking or unlocking event requires searching for the correct combination of pressure and torque acting on the clutch. The locking and unlocking events are determined during simulation by repeated and accurate zero-crossing detection. On simulating events and solving constraints together with dynamics in Simscape models, see “Desktop Simulation”.

Model Friction Clutches at a Fundamental Level

The Disk Friction Clutch block requires only a single pressure signal to modulate the kinetic friction. You fix all its other characteristics before starting simulation.

Modeling a friction clutch at a fundamental level requires direct control over the kinetic and static friction torques. The Fundamental Friction Clutch block gives you that control. With this block, you must specify, by either external signals or internal dynamics, the kinetic friction and static friction limits (positive and negative) of the clutch as functions of time.

Engage and Disengage Gears Using a Clutch

This example shows a gear being engaged, then disengaged, by a custom clutch. Torque and motion are transferred from one shaft to another over a finite time interval.

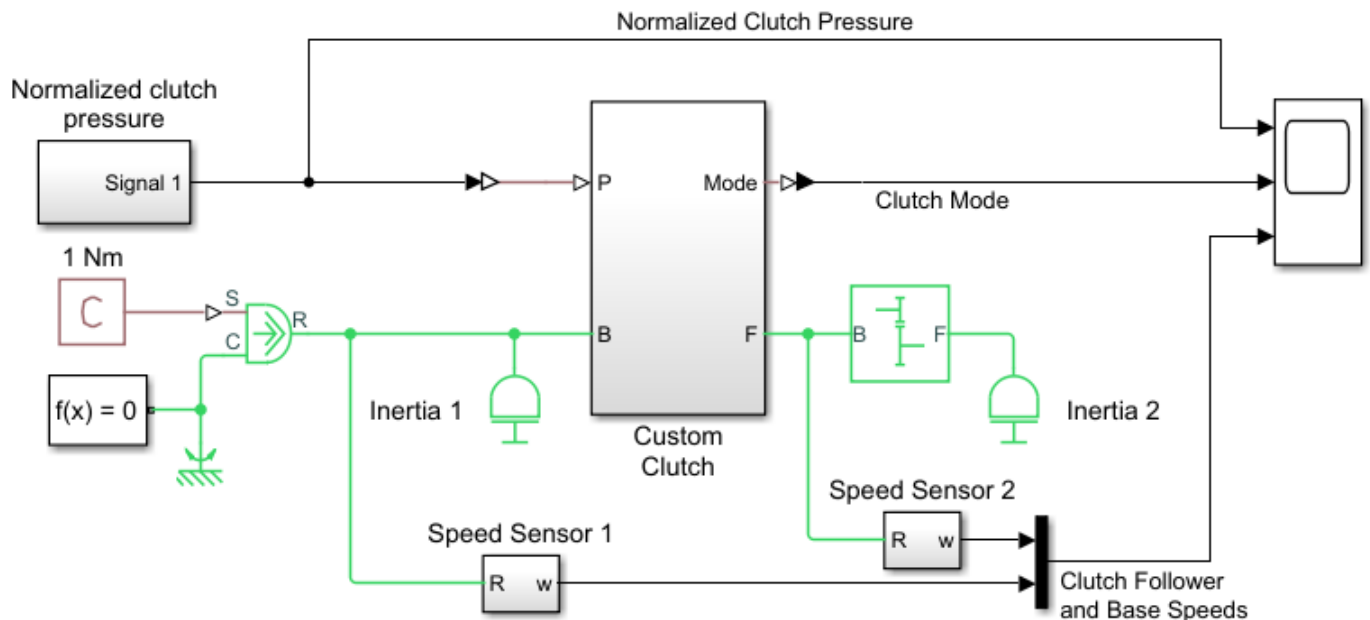
A common task in drivetrain design is transferring motion and torque at different fixed gear ratios. Drivetrains are typically designed to switch among a set of distinct gear ratios. Implementing the switch from one gear ratio to another requires gradually disengaging one set of driveline couplings and engaging another set. Clutches allow you to engage and disengage driveline shafts from one another gradually. The Disk Friction Clutch block represents a standard surface friction-based clutch that models this behavior.

The model in this example uses a custom clutch subsystem that contains a Fundamental Friction Clutch block. The Fundamental Friction Clutch block requires you to specify the static and kinetic clutch friction more completely than the Disk Friction Clutch block requires because it models clutches in greater detail. See also “Model Friction Clutches at a Fundamental Level” on page 7-3.

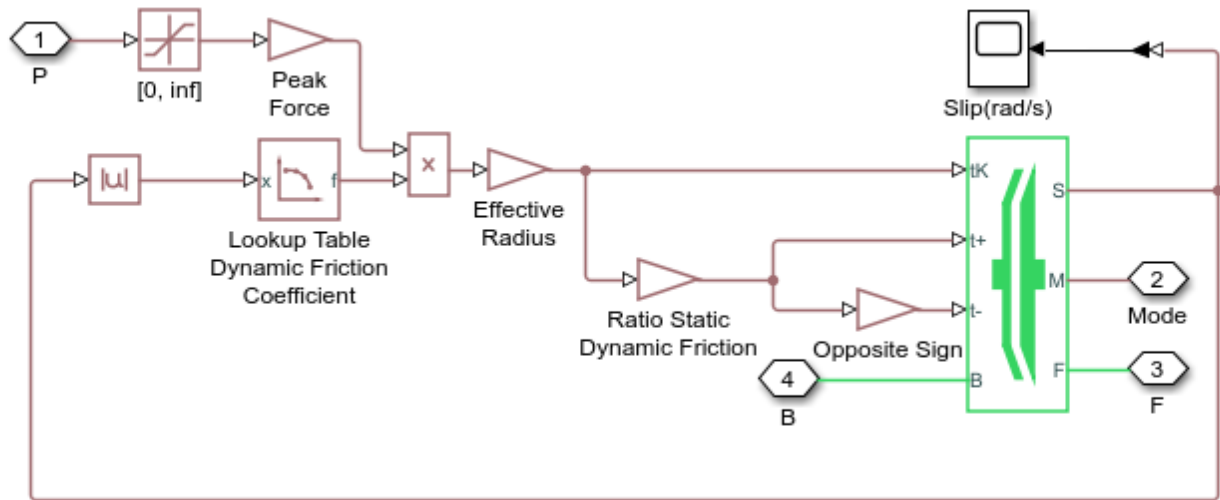
Note You can model continuous motion-torque transfer with the Torque Converter block, which simulates fluid viscosity instead of surface friction and does not lock.

Simulate Gear Engagement and Disengagement

- 1 Open the model. At the MATLAB command prompt, enter
`openExample('sdl/CustomClutchExample')`



Custom Clutch Model with Programmed Clutch Pressure



Custom Clutch Subsystem

Model Components

- The clutch subsystem is positioned between the Inertia 1 and Simple Gear blocks and reports the clutch mode (forward, reverse, locked).
- The PS Constant block replaces the sinusoidal signal as the torque input. The torque sensor blocks are omitted.
- Simulink-PS Converter and PS-Simulink Converter blocks communicate between physical signals in the Simscape environment and Simulink blocks such as a Scope block.
- The Normalized clutch pressure block provides the programmed clutch pressure signal, normalized between 0 and 1, as shown in the following table. This signal is converted to a physical pressure inside the clutch subsystem.

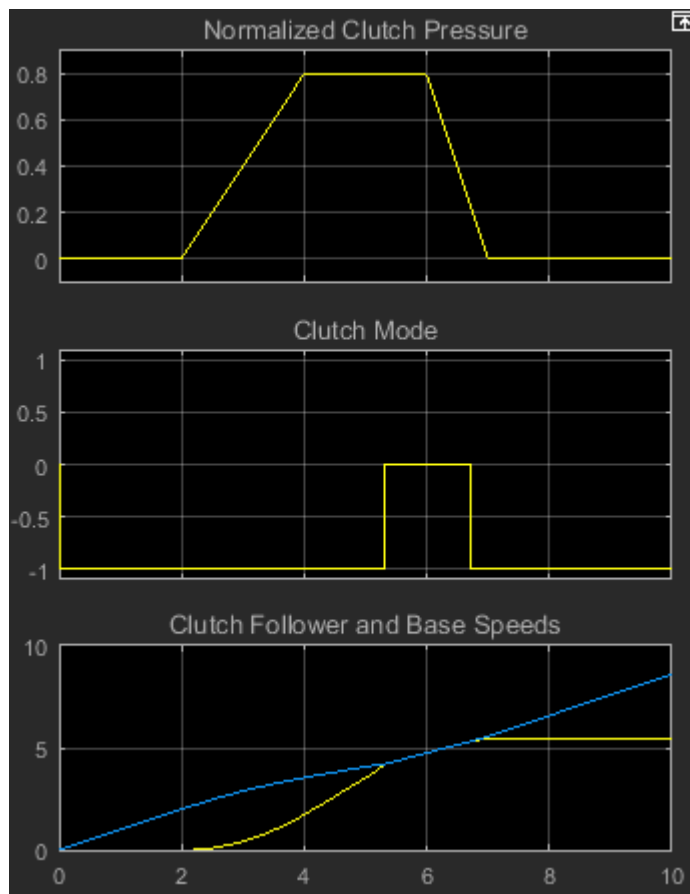
Time Range (Seconds)	Signal Value
0-2	0
2-4	0-0.8 with constant slope
4-6	0.8
6-7	0.8-0 with constant slope
7-10	0

- 2 Open the Scopes and start the simulation. The normalized clutch pressure signal follows the profile that you created in Signal Builder and determines the behavior of the model.
 - a From 0 to 2 seconds, the velocity of Inertia 1 increases linearly because it is subject to a constant torque.
 - b At 2 seconds, the clutch begins to engage, and Inertia 2 begins to spin. The velocity of Inertia 1 continues to rise, although at a slower rate, because the two inertias now share the external torque.

- c At 4 seconds, the pressure reaches its maximum. At about 5.32 seconds, the clutch locks. The driveshafts connected by the clutch now spin together. Inertia 1 and Inertia 2 continue to speed up at constant accelerations, Inertia 2 at half the velocity of Inertia 1.
- d At 6 seconds, the clutch begins to disengage as the pressure drops. Inertia 1 and Inertia 2 continue to accelerate with the applied torque.

The clutch unlocks at about 6.73 seconds and fully disengages at 7 seconds. (The clutch unlocks a little before completely disengaging because the pressure, even before vanishing, becomes too small to maintain the lock.) Inertia 1 is still accelerating. But Inertia 2, now free of the driveshaft and its torque, no longer accelerates and instead spins at a constant rate without frictional loss.

While the two shafts are locked, from 5.32–6.73 seconds, Inertia 1 and Inertia 2 spin in a fixed 2:1 ratio, because of the Simple Gear.



How the Clutch Mode Indicates Locking and Unlocking

The Clutch mode signal indicates the relative motion of its two connected shafts. From 0 to 5.32 seconds, the two shafts are moving relative to one another. The follower (driven) shaft is slower than the base (drive) shaft, so the mode signal is -1. Once the two shafts lock, their relative velocity is 0, and the mode signal switches to 0. At 6.73 seconds, they unlock, and the drive (base) shaft starts accelerating faster than the driven (follower) shaft. The mode signal switches back to -1.

Brake Motion Using Clutches

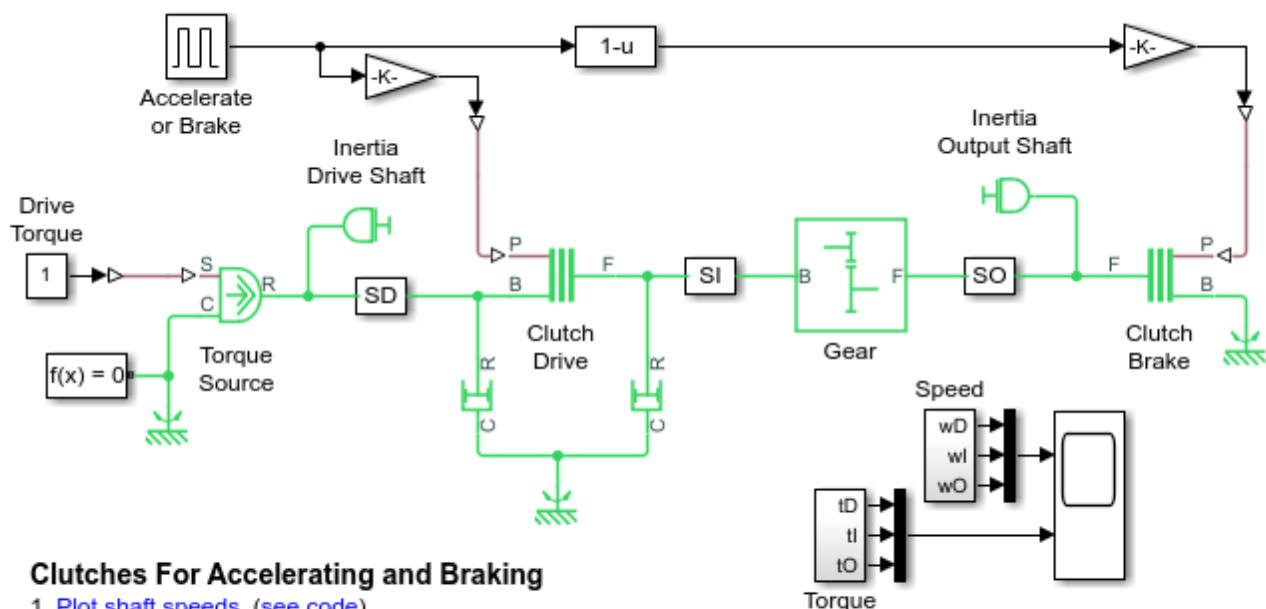
A special case of transferring motion occurs when you want to brake the spinning of a driveline component, slowing it down until it stops. The common way to brake the motion is to couple the spinning component to a rotational ground. You can represent a rotational ground with a Mechanical Rotational Reference block from the Simscape Foundation library. Because a rotational ground cannot move, a driveline axis locked to a rotational ground also cannot move. You can implement the gradual engagement or disengagement of a driveline component with a rotational ground using a clutch, just as you use a clutch to couple or uncouple two spinning shafts gradually.

Braking with a Two-Clutch System

- 1 Open the model. At the MATLAB command prompt, enter

```
openExample('sdl/ClutchesForAcceleratingAndBrakingExample')
```

The model features two clutches, one of which acts as a brake. The model also includes frictional damping for greater realism. The simulation time is set to `inf` (infinity). For simplicity, the model uses the Disk Friction Clutch block.



Clutch Model with Brake Clutch

This model uses the basic structure of inertia—clutch—gear—inertia. The first body, Inertia Drive Shaft block, is driven by an external torque, and the initial velocities are 0. There is, however, another clutch for the second body, Inertia Output Shaft block, that can couple Inertia Output Shaft to the Mechanical Rotational Reference block and bring it to a stop.

The switching assembly is based on the clutch switch. You can change this switch to apply a constant clutch pressure signal to either the Clutch Drive block or the Clutch Brake block. The Fcn 1-u block ensures that the full clutch pressure is applied to either one or the other, but not both at once. The Damper blocks apply viscous (velocity-dependent) friction to the spinning of the Inertia Drive Shaft and the Inertia Output Shaft.

The Accelerate or Brake block is programmed to provide a signal of 1 for the first 40 seconds of the simulation. It provides a signal of 0 for the following 60 seconds of the simulation.

2 Start the model.

During the first 40 seconds, when the Accelerate or Brake block is set to 1, the clutch pressure is applied to the Gear clutch. The Gear clutch engages and locks the driver and driven shafts and causes them to rotate at the same velocity.

The Inertia Output Shaft is on the other side of the Simple Gear. The angular velocity of the Inertia Drive Shaft is twice that of the Inertia Output Shaft because the gear ratio of the Simple Gear block is 2, follower to base. In this switch mode, no clutch pressure is applied to Brake Clutch, which remains unengaged.

After an initial transient, the system settles into a steady state of motion where the external torque balances the friction losses.

At $t = 40$ seconds, the Accelerate or Brake block switches to 0 to disengage the gear clutch and engage the brake clutch. The system undergoes another transient while the Gear clutch disengages and Brake clutch engages.

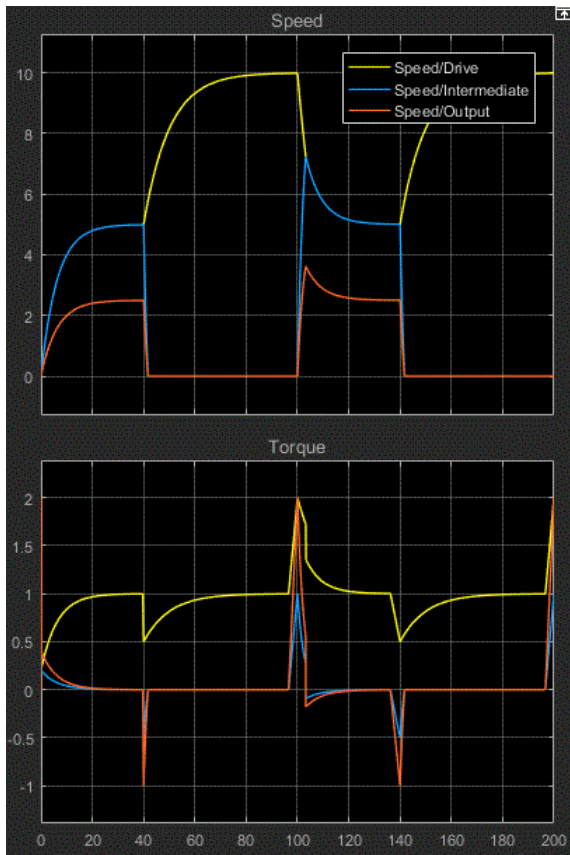
The angular velocity of Inertia Drive Shaft and the driver shaft settles down to a new steady state of 10 radians/second, twice its old speed.

Because the Gear clutch is now disengaged, the driven shaft and the Inertia Output Shaft are no longer subject to a driving torque through Gear clutch. But the Brake clutch is engaged and couples the Inertia Output Shaft to the immobile Mechanical Rotational Reference. Once engaged, the kinetic friction of the Clutch Brake brings the driven shaft and the Inertia Output Shaft to a stop.

To see the transient behavior at simulation start and when you switch the clutches:

- 1** Start the simulation and let it run for a short time. Then switch Clutch Switch to the other mode.
- 2** After a short time, stop the simulation. Use the **Autoscale** feature of the Scopes to capture the entire simulation sequence. The transients from the starting behavior and the switching transition are visible.

For example, in these plots, the model was started with Clutch Switch set to 1 (Gear clutch locked, Brake clutch disengaged, no braking). The velocities quickly climbed to their steady-state values. Then Clutch Switch was changed at 40 seconds of simulation time. Gear clutch disengaged and Brake clutch engaged, braking Inertia2. The angular velocity of the driver shaft rose from 5 to 10 radians/second. The angular velocity of the driven shaft dropped from 5 to 0. The angular velocity of Inertia2 dropped from 2.5 to 0.



Model Transmissions Using Gear Ratios and Clutch Schedules

- “Transmission Design Principles and Best Practices” on page 8-2
- “Model a Two-Speed Transmission with Braking” on page 8-3
- “Model a CR-CR 4-Speed Transmission Driveline with Braking” on page 8-6

In a real drivetrain, you couple an input or drive shaft to one of many output or driven shafts, or to one driven shaft with a choice of several gear ratios. The drivetrain then requires several clutches to switch between gears. You couple one of the driven shafts or one of the gear sets by engaging one of the clutches. You then switch to another output shaft or another gear ratio by disengaging one clutch and engaging another.

You can also engage more than one clutch at a time to use multiple gear sets simultaneously. Transmissions engage multiple gear sets at the same time to produce a single effective gear ratio, or *drive ratio*. Changing gears requires disengaging one set of clutches and engaging another set. You can specify the set of clutches to engage and disengage for each gear ratio in a *clutch schedule*. Designing a clutch schedule and shaping and sequencing the clutch pressure signals frequently constitute the most difficult part of transmission design. A realistic transmission model must also include losses due to friction and imperfect gear meshing.

To learn how to model transmissions using blocks from the Simscape Driveline gear and clutch libraries, see “Model a Two-Speed Transmission with Braking” on page 8-3 and “Model a CR-CR 4-Speed Transmission Driveline with Braking” on page 8-6.

Transmission Design Principles and Best Practices

When creating or modifying a transmission model:

- For a realistic simulation and to prevent acceleration singularities when torques are applied, connect inertia blocks with nonzero inertia values to gear shafts.
- To specify the engaged and free clutches for proper transmission gearing, set up a clutch schedule. Set all clutch pressures to 0 only if you want to disengage the transmission completely, that is, place it in neutral.

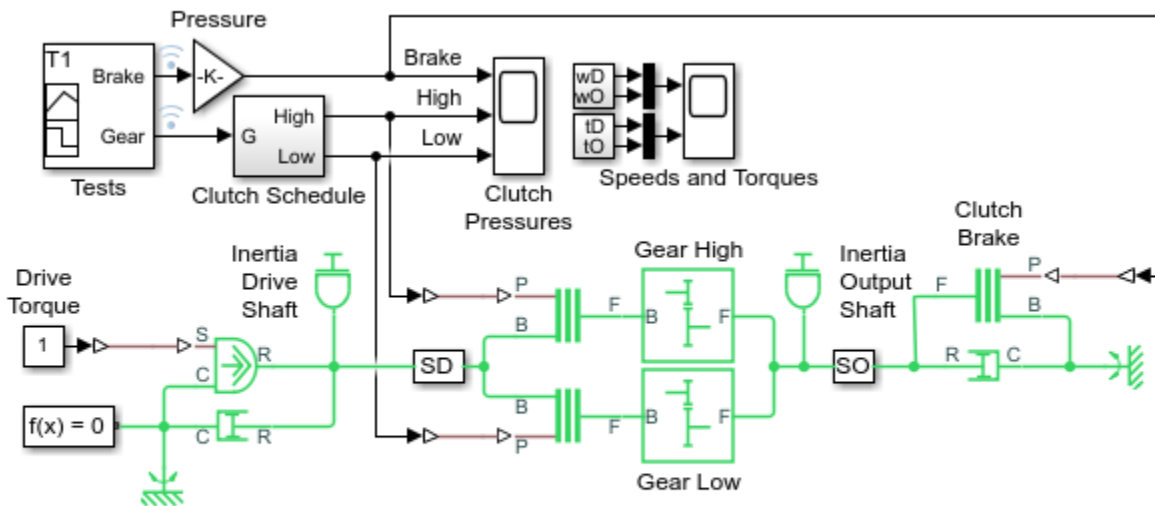
Do not engage any more or fewer clutches than necessary, at any time during simulation.

- If you redesign a transmission by adding or removing gears, you might also have to:
 - Add or remove clutches.
 - Redesign the clutch schedule.
 - Add or remove gear shaft inertias.

On the relationship clutch pressure signals to solver choices and settings, see “Driveline Simulation Performance” on page 16-2.

Model a Two-Speed Transmission with Braking

The example model `TwoSpeedTransmissionExample` contains a driveline system that makes up a simple yet complete transmission.



Two-Speed Transmission

1. [Plot shaft speeds](#) (see code)
2. [Explore simulation results](#) using `sscexplore`
3. [Learn more](#) about this example

Simple Transmission with Two Gear-Clutch Pairs and Braking

The model is built on the `ClutchesForAcceleratingAndBrakingExample` example model. This model contains two driveline shafts or axes, with a constant actuating torque of 1 Newton-meter applied to the driver shaft. Both the driver and the driven shafts are subject to small viscous damping torques. The viscous torque constant μ is 0.001 newton-meters/(radians/second). In the steady state, the driving and damping torques balance one another; the two shafts spin at constant rates, the driver shaft at $(1 \text{ N-m}) / (0.001 \text{ N-m}/(\text{rad/s})) = 1000 \text{ rad/s}$. If braking occurs, the driven shaft stops. There are now two selectable gears to couple the two axes, instead of one. For more information on modeling viscous losses with nonideal gear bearings instead of dampers, see “Model Gears with Losses” on page 11-3 and “Constant and Load-Dependent Gear Efficiencies” on page 11-5.

This transmission model couples the gears in a simple way, with each gear and the brake associated with its own clutch. Coupling one gear requires engaging and locking the corresponding clutch, while ensuring that the other two clutches are disengaged. The brake clutch is directly activated by its own switch.

Setting Up the Gears, Clutches, and Brake

The two gears are Simple Gear blocks with different gear ratios, each connected in series with its corresponding clutch. The two gear-clutch pairs are coupled in parallel. This parallel assembly then couples the driver shaft to the driven shaft, with their two spinning inertias. One gear is a “low” gear,

the other a “high” gear. Following common usage for automobile gears, the “low” and “high” labels refer to the angular velocity ratios.

Note The ratio of speeds in a gear is the *reciprocal* of the gear ratio.

- The low gear is the Gear High block. You can couple the low gear by engaging its corresponding clutch, modeled by the Low gear clutch block. The gear ratio is 5:1, so that the ratio of output to input (follower to base) angular speeds is 1/5. Such a gear has a high torque transfer ratio of 5, from base to follower. In an automobile, such low gears are used to accelerate the vehicle from a stop by transferring a large torque down the drivetrain from the engine.
- The high gear is the Gear Low block, coupled by engaging its own clutch, represented by the High gear clutch block. The gear ratio is 2:1, and the angular velocity ratio of follower to base is 1/2, or 5/2 times the ratio in the low gear. The torque transfer ratio is only 2 from base to follower. An automotive high gear is used for milder acceleration or coasting once a vehicle is moving at a significant speed. The vehicle acceleration generated by this gear is less than the acceleration that is generated by the low gear.

Switching on either the Neutral switch or the Brake switch disengages both gear clutches. In either case, the driver shaft continues to spin, approaching a steady velocity, subject to the competing driving and damping torques.

- Switching the transmission to neutral leaves the brake clutch disengaged and the driven shaft free to spin. But without a driving torque, damping gradually brings the driven shaft to a stop.
- Switching on the brake immediately locks the brake clutch and stops the driven shaft.

This simple transmission is based on mapping each transmission state one-to-one with an engaged clutch. You cannot engage more than one clutch at a time without creating conflicts between gear ratios or between the driver shaft and the rotational ground.

Controlling the Transmission State with a Clutch Schedule

The requirement to engage a certain clutch or set of clutches and disengage others, both to implement transmission functions and to avoid motion conflicts between gears, is the basis for all clutch schedules. Simulink provides a number of ways to implement clutch schedules, depending on the complexity of the transmission and how much realism you require for the clutch pressure signals.

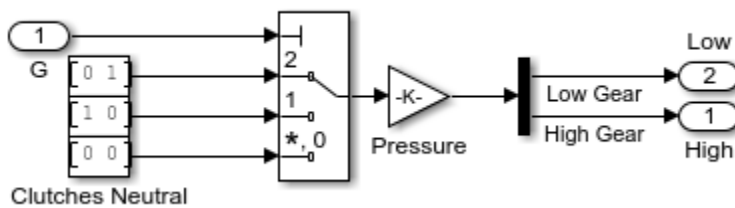
Caution To ensure that the transmission states are implemented correctly and to avoid motion conflicts among gear sets, check the clutch schedule for the transmission. To make sure that the clutches are engaged, locked, unlocked, and disengaged in a realistic and conflict-free manner, check the clutch pressure signal profiles. Unphysical or conflicting clutch schedules and clutch pressure signals lead to simulation errors in Simscape Driveline models.

For the `TwoSpeedTransmissionExample` model, avoiding such conflicts leads to a unique clutch schedule.

Clutch Schedule for the Simple Two-Speed Transmission

Transmission State	Brake Clutch State	Low Gear Clutch State	High Gear Clutch State
Neutral/Braked	Disengaged/Locked	Disengaged	Disengaged
Low Gear	Disengaged	Locked	Disengaged
High Gear	Disengaged	Disengaged	Locked

The model contains a simple Clutch Control subsystem to implement the clutch schedule and to output the clutch pressure signals to lock each clutch as needed.



Clutch Control Subsystem for Simple Transmission Model

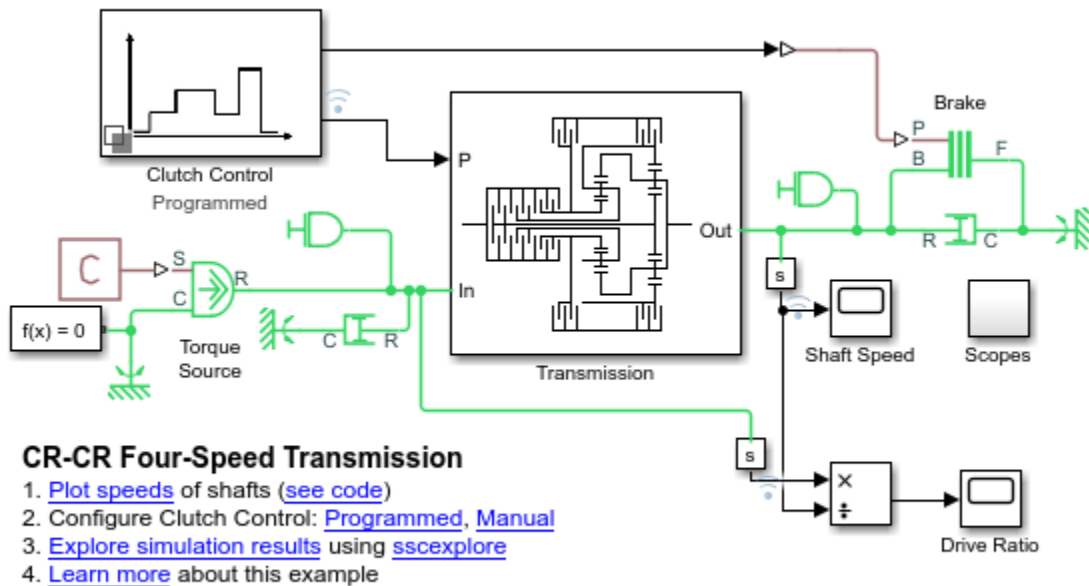
Adding Realistic Clutch Signals

The clutch control subsystem of this example is adequate for a simple model, but not realistic. It contains unrealistic clutch pressure signals that rise and fall sharply. A full clutch control model requires realistic clutch pressure signals that rise from and fall back to zero in a smooth way. Greater realism requires a potentially more complex model. During simulation, the Simscape and Simulink solvers can determine transmission motion only if exactly two clutches are locked, or if all four clutches are unlocked. This model is similar to a real transmission where improperly constrained clutches can lead to lockup or damage to the transmission components. Changing the gear settings for the transmission while maintaining this requirement is an example of the central problem of transmission design.

For transmission and car examples with smoothed clutch pressure signals, see “Model a CR-CR 4-Speed Transmission Driveline with Braking” on page 8-6 and “Complete Vehicle Model” on page 3-2.

Model a CR-CR 4-Speed Transmission Driveline with Braking

The CR-CR Four-Speed Transmission Example models a realistic transmission. It uses a CR-CR 4-Speed transmission subsystem to transfer motion and torque from one shaft and inertia to another.



CR-CR 4-Speed Transmission Model

There is a constant driving torque from a torque source to the driver shaft (Inertia block on the left). Two damping subsystems apply heavy and light viscous friction to the driver and driven shafts, respectively. The two scope subsystems measure the clutch pressures. The model workspace defines essential parameters for the blocks. For information on creating, accessing, and changing model workspace variables, see “Specify Source for Data in Model Workspace” and “Change Model Workspace Data”.

The CR-CR 4-Speed transmission subsystem couples the driver to the driven shaft (Inertia block on the right). If the transmission is disengaged, a brake clutch and fixed housing allow you to brake the driven shaft.

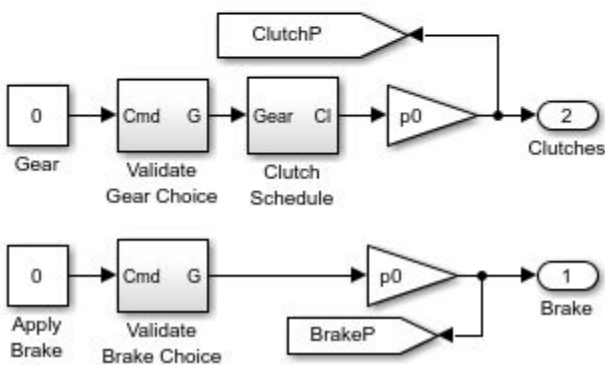
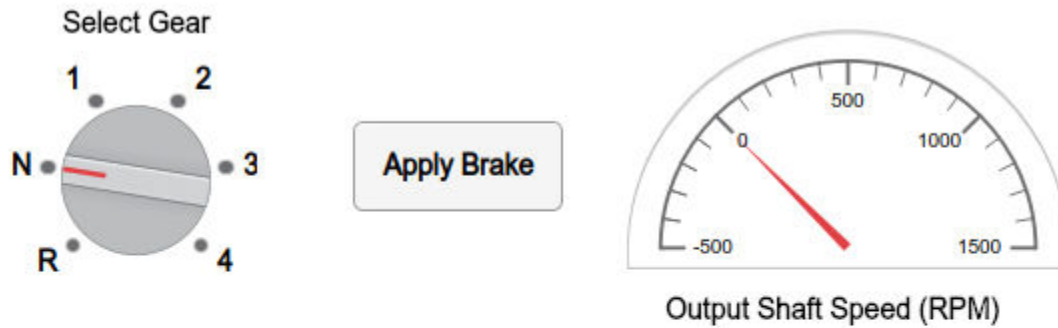
For clarity, the major signal buses of the model have been bundled as vectors and directed using Goto and From blocks. The Clutch Pressures are collected in the Scopes subsystem for convenience.

Replacing Programmed with Manually Controlled Clutch Pressures

The model represents the clutch control system using a Variant Subsystem block. To switch between **Programmed** and **Manual**, the two clutch control modes that the variant provides, click the links in

the model window. During simulation, the manual subsystem provides direct control over gear changes. To switch gears in manual control mode:

- 1 In the Simulink toolbar, change the simulation time to `inf`.
- 2 Run simulation.
- 3 Change gears during using the **Select Gear** widget.



Manually Switchable Clutch Control for the CR-CR Transmission

Use this subsystem as a manually switchable clutch control for the CR-CR transmission.

For complete control, change the simulation time to infinity.

Manual Clutch Control for CR-CR Transmission

Modeling Driveline Components

These sections introduce you to modeling specialized driveline components in the Simscape Driveline environment. Using predefined blocks from the Simscape Foundation and Simscape Driveline libraries and constructing custom components of your own are both emphasized.

- “Specialized and Customized Driveline Components” on page 9-2
- “Rotational-Translational Couplings” on page 9-4
- “Modeling Transmissions” on page 9-5

Specialized and Customized Driveline Components

In this section...

“Optimal Physical Modeling in the Simscape Environment” on page 9-2

“Reasons for Specialized Driveline Components” on page 9-2

“Greater Model Fidelity and Performance” on page 9-3

Optimal Physical Modeling in the Simscape Environment

Within the Simulink environment, you follow best practice if you model driveline systems by representing as much of the physical system as possible with Simscape Driveline and Simscape components. Major advantages include these features that make it easier to create accurate simulations of physical systems:

- Physical ports and connection lines supporting physical units
- Data logging
- Specialized solvers
- Consistent treatment of differential and algebraic constraint equations
- Customization with Simscape Foundation blocks and Simscape language

For custom block modeling with Simscape language, see “Custom Components”.

Reserve Simulink blocks and signals for nonphysical aspects of modeling, such as nonphysical signals, algorithmic control, and model-level input/output tasks.

Reasons for Specialized Driveline Components

Simscape Driveline and Simscape physical connections help you create model architectures with clear physical component boundaries. You can then increase the fidelity of the model overall, or make only certain components more accurate representations of the system.

In driveline modeling, there are several reasons for making a model more complex and accurate.

Complex Component Geometries

Driveline models abstract the motion of three-dimensional mechanical systems constrained to move in one dimension. Some driveline components require extra specification to capture underlying two- and three-dimensional geometry. An example in the Simscape Driveline library is the Differential gear.

Internal Compliance Dynamics

Many standard Simscape Driveline components allow you to enable or disable modeling of internal dynamics. For example, Generic Engine allows you to represent an engine with instantaneous response for a simple, idealized model. To access a more complex and accurate representation, enable the internal time lag for the Generic Engine block. You can also create your own custom components (with internal compliance dynamics, for example), to whatever degree of fidelity and complexity that you want.

Frictional Losses

Much of complex internal dynamic Simscape Driveline modeling comes from representing the effect of both viscous and Coulomb friction.

- Viscous friction is proportional to the relative velocity of two surfaces in contact.
- Coulomb, or “sliding-sticky,” friction is proportional to the force normal to surfaces in contact. For low relative velocities, Coulomb friction causes surfaces to lock and cease relative motion.

Thus, friction models involve specification of relative geometry and motion, friction coefficients, and normal forces, of surfaces in contact.

Components with internal friction models include gears, clutches, tires, and other couplings, at different levels of optional complexity.

Greater Model Fidelity and Performance

Typically, greater fidelity of model components results in reduced simulation performance and changes the tradeoff between simulation accuracy and speed. You can adapt your simulation methods to handle greater fidelity, or reduce model fidelity to enhance performance.

For a discussion of how model fidelity and performance are related to simulation settings, see “Adjust Model Fidelity” on page 16-2 in “Driveline Simulation Performance” on page 16-2.

Rotational-Translational Couplings

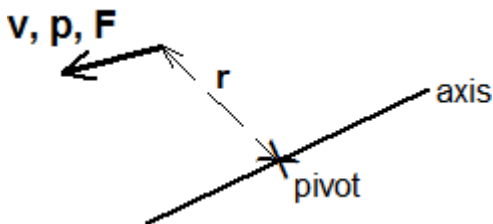
In this section...

“Convert Between Rotational and Translation Motion” on page 9-4

“Use Simscape and Simscape Driveline Elements to Couple Rotation and Translation” on page 9-4

Convert Between Rotational and Translation Motion

In general, mechanical systems mix rotational and translational motion. Rotational dynamics and motion about an axis couples to translational dynamics and motion (velocity \mathbf{v} , momentum \mathbf{p} , force \mathbf{F}) at some distance (moment arm \mathbf{r}) from the rotational pivot through a constraint that captures only motion normal to the moment arm.



While most driveline components involve rotational motion around fixed axes, certain key components transform translational and rotational motions from one to the other. Such components map motion along lines and motion around circles to one another.

The rack-and-pinion is an example of a mixed-motion gear constraint. Tires and propellers use contact friction to change driven rotational motion into forward or backward linear motion.

Use Simscape and Simscape Driveline Elements to Couple Rotation and Translation

Simscape and Simscape Driveline components that couple rotational and translational motion have mixed mechanical conserving ports of both rotational and translational type. Such blocks include wheels, tires, and certain gears:

- Leadscrew
- Rack & Pinion
- Tire (Magic Formula)
- Wheel and Axle

For an example of rotational-translational coupling with Tire (Magic Formula), see the model in “Complete Vehicle Model” on page 3-2.

Modeling Transmissions

In this section...

“Transmission Templates” on page 9-5
 “Transmission Ports” on page 9-5
 “Gear Input Signal” on page 9-5
 “Initial States” on page 9-6
 “Clutch Control” on page 9-6
 “Inertias and Friction Losses” on page 9-7
 “Real-Time Simulation” on page 9-7

Transmission Templates

The Transmissions library provides subsystem templates for modeling geared transmission systems with four to nine speed settings. The templates use Simscape Driveline and Simscape blocks to represent the transmission components—their gears, clutches, and brakes. An embedded Simulink subsystem defines the clutch schedule.

Use the templates as starting points and examples for your own transmission models. The template blocks are not library-linked, so you can modify them to suit your needs. Add, remove, or reconnect blocks to change the transmission structure. To capture transmission losses due to gear meshing or viscous damping, modify the block parameters.

Transmission Ports

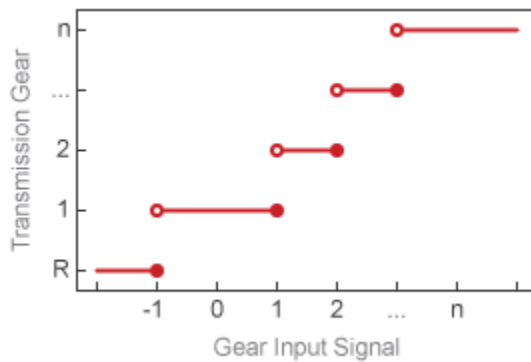
The transmission template blocks each have three ports. Ports **B** and **F** are rotational conserving ports. They represent, interchangeably, the input and output shafts. The **Gear** port is a Simulink input port that you can use to shift gears during simulation.

Gear Input Signal

The input signal to the **Gear** port sets the transmission gear according to the expression:

$$\text{Gear Setting} = \begin{cases} \text{Rev, if Gear} \leq -1 \\ 1^{\text{st}}, \text{ if } -1 < \text{Gear} \leq +1 \\ 2^{\text{nd}}, \text{ if } +1 < \text{Gear} \leq +2 \\ \dots \\ n^{\text{th}} \text{ if } (n-1) < \text{Gear} \leq n \end{cases} .$$

The figure shows the correspondence between the transmission gear and the Gear input signal.



Initial States

The transmission templates are configured to begin simulation in first gear. To change the default initial gear, you must set the clutch and brake initial states to the values shown in the transmission clutch schedules. You change the initial states in the **Initial Conditions** tab of the block property inspector.

Consider the Lepelletier 6-Speed Transmission template. The transmission clutch schedule shows the clutch and brake initial states for reverse gear to be [1 0 0 1 0] in the order A-E. To begin simulation in reverse gear, you must then set the initial states in the block property inspector as follows:

- Clutch A locked
- Clutch B unlocked
- Clutch C unlocked
- Clutch D locked
- Clutch E unlocked

The Gear input signal must agree with the clutch and brake initial states. If the initial states correspond to reverse gear, then the first value in the Gear input signal must also correspond to reverse gear ($Gear \leq -1$). Matching the two in this way helps to avoid inconsistent states known to cause simulation errors.

Clutch Control

The clutch schedule converts the Gear input signal into clutch and brake input signals. These signals drive the clutches and brakes, causing some to lock and others to unlock in an orchestrated fashion. The resulting configuration determines which gears power flows through—and therefore which gear the transmission is in.

A Gain block scales the input signals to the proper magnitudes for actuating the clutches and brakes. These magnitudes depend on the actuation inputs expected by the Clutch and Brake blocks. The actuation inputs can be:

- Shift linkage displacements in Dog Clutch blocks
- Normal forces in Cone Clutch and Loaded-Contact Rotational Friction blocks
- Plate pressures in Disk Friction Clutch blocks

Transfer Function blocks smooth the otherwise discrete input signals using first-order filters. The smoothing allows the clutch state transitions to occur gradually over short periods of time rather than instantaneously. The transfer function time constants determine the characteristic time periods over which the clutch transitions occur.

Inertias and Friction Losses

Simscape Inertia blocks represent the inertias of transmission gears. These blocks enhance the accuracy of the model. They also prevent simulation errors due to zero-inertia gear sets disconnected from input and output shafts due to clutch unlocking.

By default, all frictional losses are set to zero. Frictional losses include losses due to meshing in gears and viscous damping in gears, clutches, and brakes. To account for frictional losses in your model, you must specify gear efficiencies and friction coefficients in the block property inspector.

Real-Time Simulation

Transmission components such as gears and clutches can slow down simulation by introducing zero-crossing events and complex state-change calculations to your model. To minimize their impact on simulation speed, several blocks provide optional parameterizations suited for real-time simulation. These parameterizations include:

- `Friction clutch approximation` in dog clutches — Reduces model stiffness due to backlash-induced vibrations.
- `No meshing losses` in gears — Eliminates gear friction calculations and associated zero crossings due to motion reversals.

These block parameterizations are labeled `Suitable for HIL`. Use them if you intend to run any type of real-time simulation, including hardware-in-loop (HIL) and software-in-loop (SIL) simulation. By default, all gear and clutch blocks in the transmission templates are set to use these parameterizations.

See Also

4-Speed CR-CR | 4-Speed Ravigneaux | 6-Speed Lepelletier | 7-Speed Lepelletier | 8-Speed | 9-Speed

Related Examples

- “Transmission Testbed” on page 18-131

Effective Inertias and Driveshafts

Model Driveshafts with Loss

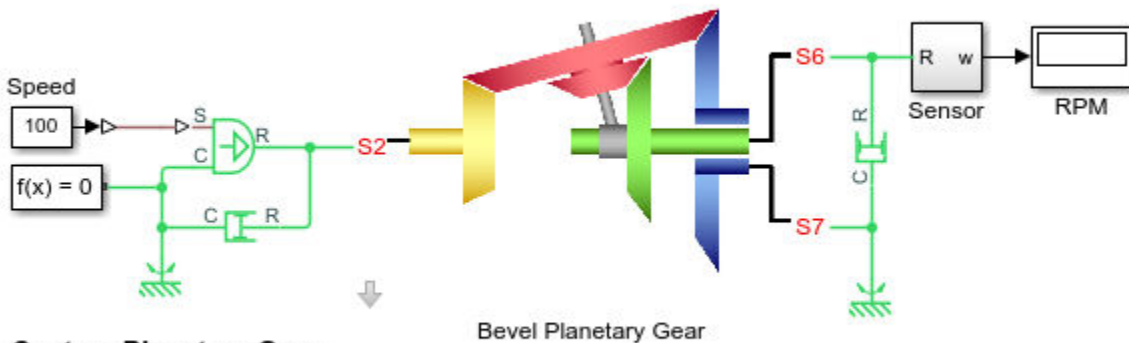
Realistic driveshafts experience damping from viscous friction, which is proportional to the driveshaft angular velocity. You can model such damping with the Rotational Damper and, if necessary, build complex damping subsystems from this block.

Specialized Gears

Make custom gears blocks the represent gear subcomponents. Create realistic models with gears that experience frictional, thermal, or load-dependent losses.

Custom Planetary Gear Model

While the Simscape Driveline library contains a Planetary Gear, you can create your own custom planetary gear using the **Planetary Subcomponents** sublibrary. The “Custom Planetary Gear” on page 18-27 example model combines three Sun-Planet Bevel subgears into a masked subsystem to model a coupled planetary gear train. The model uses an Ideal Angular Velocity Source to place a fixed velocity demand on the gearbox input, while damping the system on both the input and output driveshafts with Rotational Damper blocks.

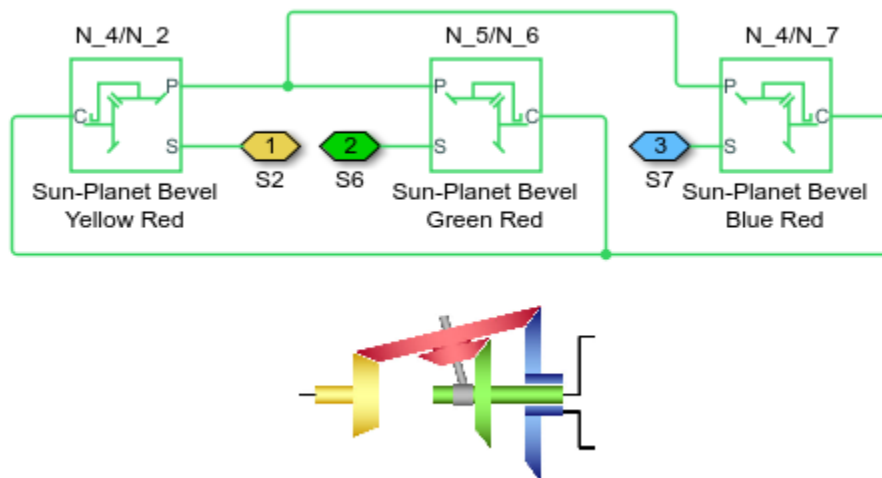


Custom Planetary Gear

1. [Explore simulation results](#) using [sscexplore](#)
2. [Learn more](#) about this example



Custom Planetary Gear System and Subsystem



See Also

Compound Planetary Gear | Ideal Angular Velocity Source | Rotational Damper | Sun-Planet Bevel

Model Gears with Losses

The blocks of the Simscape Driveline Gears Library contain optional built-in models of frictional losses, allowing you to represent nonideal gear couplings. In a nonideal gear pair (1,2), the angular velocity, gear radii, gear teeth constraints, and gear ratio $g_{12} = r_2/r_1 = \omega_1/\omega_2$ are unchanged. The transferred torque and power are reduced by:

- Coulomb friction between imperfectly meshing teeth surfaces on gears 1 and 2, parameterized by an efficiency η , $0 < \eta \leq 1$. This efficiency depends on the torque load on the teeth. But it is often approximated as constant.
- Viscous coupling of driveshafts with bearings, parameterized by viscous friction coefficients μ .

Constant Efficiency

In the simplest nonideal gear loss model, the efficiency η_{12} of meshing in gear pair (1,2) is constant, independent of load (torque or power transferred).

- The friction loss represented by η_{12} is effectively applied in full only if the transmitted power is greater than the power threshold p_{th} . Below this value, a hyperbolic tangent function smooths the efficiency factor, lowering the efficiency losses to zero when no power is transmitted.
- For gear sets with a carrier, η_{12} represents the ordinary efficiency, defined when the carrier is not moving.

For gears with different efficiencies for the forward and reverse power flow:

- *ForwardLoss* = $(1 - \eta_{FB})$, η_{FB} is the torque transfer efficiency from the follower shaft to the base shaft.
- *BackwardLoss* = $(1/\eta_{BF} - 1)$, where η_{BF} is the torque transfer efficiency from the base shaft to the follower shaft.

The frictional torque is calculated as:

$$T_f = T / 2((ForwardLoss + BackwardLoss)\tanh(4p / p_{th}) + ForwardLoss - BackwardLoss)$$

where:

- T is the transferred torque.
- p is the transferred power.
- p_{th} is the power threshold at the base shaft above which full efficiency losses are in effect.

For certain gear models, such as the Simple Gear, efficiency is assumed equal for both the forward and reverse power flow, $\eta_{BF} = \eta_{FB}$.

Load-Dependent Efficiency

Making η dependent on the load is a way to make the loss model more accurate. For an example of load-dependent efficiency, see the Simple Gear block reference page.

Geometry-Dependent Efficiency

Making η dependent on the geometry of gear meshing is another way to make the loss model more accurate. For an example of geometry-dependent efficiency, see the Leadscrew block reference page.

Viscous Friction

On a driveshaft mounted to a gear wheel by lubricated, nonideal bearings, the viscous friction experienced by the axis is controlled by the viscous friction coefficient μ . The viscous friction torque on a driveshaft "a" is $-\mu_a \cdot \omega_a$, where ω_a is the angular velocity of the driveshaft with respect to its mounting or carrier (if a carrier is present).

See Also

More About

- "Constant and Load-Dependent Gear Efficiencies" on page 11-5

Constant and Load-Dependent Gear Efficiencies

Here, you revisit the `TwoSpeedTransmissionExample` model in “Model a Two-Speed Transmission with Braking” on page 8-3. You reconfigure the Simple Gears to model power loss due to nonideal meshing. The effect of viscous bearing losses is ignored.

- 1 Open the “Two-Speed Transmission” on page 18-143 model and simulate to check the ideal gear behavior.
- 2 Open the Gear High and the Gear Low blocks. Under **Meshing Losses**, in the **Friction model** drop-down list, choose `Constant efficiency` for both. Enter efficiencies less than 1, but greater than 0. For example, for the Gear Low block, enter 0.7; and for the Gear High block, enter 0.95.
- 3 Leave the other settings as they are, including zero viscosity. Close the blocks.
- 4 Restart the model. The driveline runs at a lower efficiency and slightly smaller angular velocities, because of the power losses. If you enter different efficiency factors for the two gears, the effect of the loss is different if you switch between gears.

Experiment with load-dependent efficiency. In the **Friction model** drop-down menu, choose `Load-dependent efficiency` instead. In that case, you need more efficiency model details to specify.

See Also

More About

- “Model Gears with Losses” on page 11-3

Specialized Clutches

Create realistic models with clutches that experience frictional losses and smooth pressure signals.

Clutches, Clutch-Like Elements, and Coulomb Friction

Coulomb friction acts along the plane of contact between two solid surfaces, in opposition to their actual or potential relative motion, and in proportion to the normal force pushing the surfaces together. It encompasses both kinetic friction, applied when the surfaces are in relative motion, and static friction, applied when they are locked together. Coulomb friction is the basis for clutches and clutch-like elements that rely on normal forces to keep surfaces in contact. When the relative speed of the surfaces becomes small enough and a normal force is applied, these elements lock and move together.

Realistic friction models often include viscous friction. This type of frictional force or torque is proportional to the relative translational or rotational velocity of the two surfaces in contact.

The Clutches library contains various clutch types, including single- and multi-plate, friction, cone, and dog (positive) clutches. You can customize the fundamental clutch blocks to meet your requirements. The Brakes & Detents library provides brake, detent, and friction blocks. These clutch-like elements apply Coulomb friction forces or torques between pairs of translating or rotating axes in loaded contact. Many also allow inclusion of viscous friction. Once engaged, clutches and brakes act to decelerate the relative motion of surfaces in contact and can lock the surfaces together under certain conditions.

Clutches and clutch-like elements have a dual role in a driveline model. When engaged but not locked, they act as *dynamic elements*, generating torques and forces between driveline axes in relative motion. When locked, they act as *conditional* or *dynamic constraints*, locking driveline axes to move together. Such constraints are conditional, because they can unlock, unlike gears.

For more information on clutches and dynamic constraints, see “Driveline Degrees of Freedom” on page 16-21 and “Driveline States & Effect of Clutches” on page 16-32.

See Also

More About

- “Model Clutches with Viscous Friction Loss” on page 12-3

Model Clutches with Viscous Friction Loss

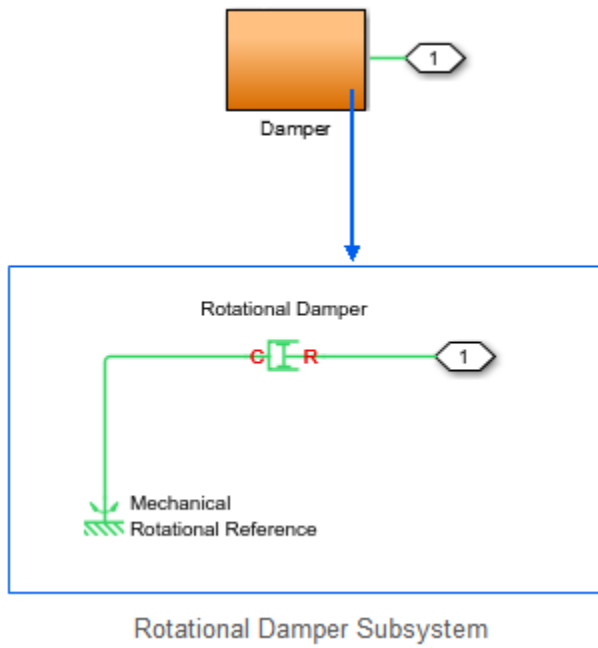
A source of loss in a clutch system coupling two driveshafts comes from viscous friction at the two shaft bearings. Consider the “Custom Clutch” on page 18-24 example model. Here you add a kinetic friction torque proportional to the angular velocity on both sides of the clutch (viscous friction). The Simscape Foundation library provides a Rotational Damper block that represents such a damper. The angular motion of the driveshafts is relative to another component. Here the angular velocities of the shafts are measured relative to rotational ground, represented by Mechanical Rotational Reference. You can make a friction subsystem that applies such a torque to any driveline axis connected to it. You can copy the subsystem and modify the existing clutch model by connecting the two copies on either side of the clutch.

Note The velocity used in this damping is the absolute velocity of a single shaft relative to rest. If you had *two* rotating driveline shafts and wanted to exert a relative damping between them as a function of their *relative* velocities, use the same Rotational Damper block connected between the two axes.

Creating a Torque Damping Subsystem

The viscous friction torque is $\tau_{\text{fric}} = -\mu\omega$, where μ is the viscous friction coefficient. To implement this torque:

- 1 You can start with your modified model from the “Engage and Disengage Gears Using a Clutch” on page 7-4tutorial or with the CustomClutch model. In either case, open the model.
- 2 From the Simscape library, copy Mechanical Rotational Reference, Rotational Damper, and Connection Port into your model window.
- 3 From the Simscape Driveline **Couplings & Drives > Springs & Dampers** library, copy the Rotational Damper block into your model window.
- 4 Connect the Mechanical Rotational Reference to the case (C) port of the Rotational Damper and the rod (R) port of the Rotational Damper to the Connection Port.
- 5 For the Rotational Damper block, for the **Damping coefficient**, enter 0.3. Leave the default units.
- 6 Select the whole connected three-block set, and create a subsystem. Name the subsystem Damper 1.
- 7 Create a second copy of Damper. Name the new subsystem Damper 2.

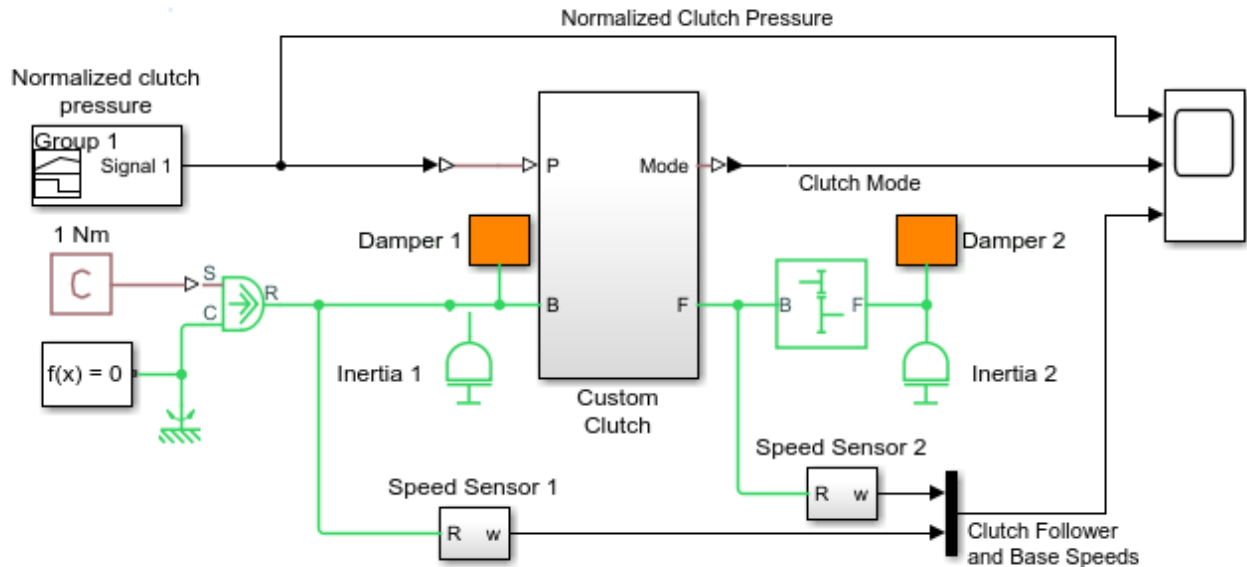


Rotational Damping Subsystem

Connecting and Simulating the Damped Clutch System

Complete and run the model.

- 1 Connect the two Damper subsystems to the driveline of the clutch model, as shown in the figure.

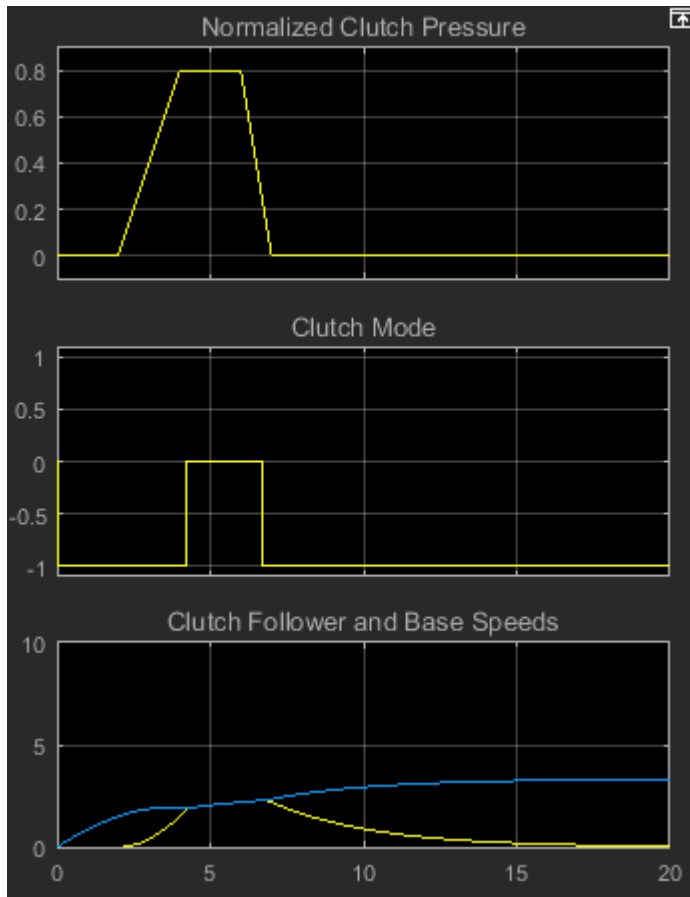


Custom Clutch

1. [Plot speeds](#) of clutch shafts ([see code](#))
2. [Explore simulation results](#) using [sscexplore](#)
3. [Learn more](#) about this example

Damped Custom Clutch Model

- 2 Change the simulation time to 20 seconds.
- 3 Open the Scope blocks and click **Start**. To see the full plots, readjust the horizontal axes of the Scope using **Autoscale**.



The clutch pressure and external torques are applied as before. The shafts now rotate slower because of the damping.

As before, Inertia2 begins to spin when the clutch starts to engage at 2 seconds. After the clutch locks at 4 seconds, the body continues to accelerate, at a slower rate than it did without damping. At about 6.7 seconds, the clutch begins to disengage and completely disengages at 7 seconds. Subject to friction, Inertia2 now starts to slow down, unlike in the friction-free case. Once the external torque is removed, its angular velocity drops exponentially with time.

The behavior of Inertia1 is more complex. It begins to spin up, at a lower rate than before, because of the damping. From 2-7 seconds in the simulation, Inertia1 shares the external torque with Inertia2 via the Clutch and the Simple Gear. After 7 seconds, the external torque applies to Inertia1 alone. It continues to accelerate, at an ever-slowing rate, because of the damping. If you let the simulation run without stopping, Inertia1 approaches its terminal angular velocity, a state where the frictional torque exactly balances the externally applied torque. This terminal velocity is $\omega_{\text{term}} = \tau_{\text{ext}}/\mu$ or $1/0.3 = 3.3333$ radians/second. The third Scope plot approaches this terminal value.

See Also

More About

- “Clutches, Clutch-Like Elements, and Coulomb Friction” on page 12-2

Model Realistic Clutch Pressure Signals

The most critical addition that you can make to clutch models for greater realism is to change the clutch pressure signals from step functions (0 to 1, or 1 to 0) to signals with a smooth rise and fall. This greater realism results in a more complex model. At any simulation time, it is critical for your model to determine transmission motion by locking exactly the correct number of clutches. If all clutches are unlocked, the transmission is in neutral. Changing the gear settings of a transmission while maintaining this requirement is one of the central problems of transmission design.

Such transmission and vehicle models as “CR-CR Four-Speed Transmission” on page 18-21 and “Vehicle with Four-Speed Transmission” on page 18-161 switch gear settings without placing their transmissions in neutral. Controlling an actual manual transmission requires moving the transmission out of gear and into neutral, picking a new gear setting, and then putting the transmission into the new gear.

In the `CRCRFourSpeedTransmissionExample` model, with manual transmission control, you can mimic these steps by turning on the Neutral Switch, changing the gear setting, then turning off slipping the Neutral Switch.

In a programmed transmission control model, you can filter clutch pressures with Transfer Fcn blocks, shaping the pressure signals from sharp steps to smooth rises or falls.

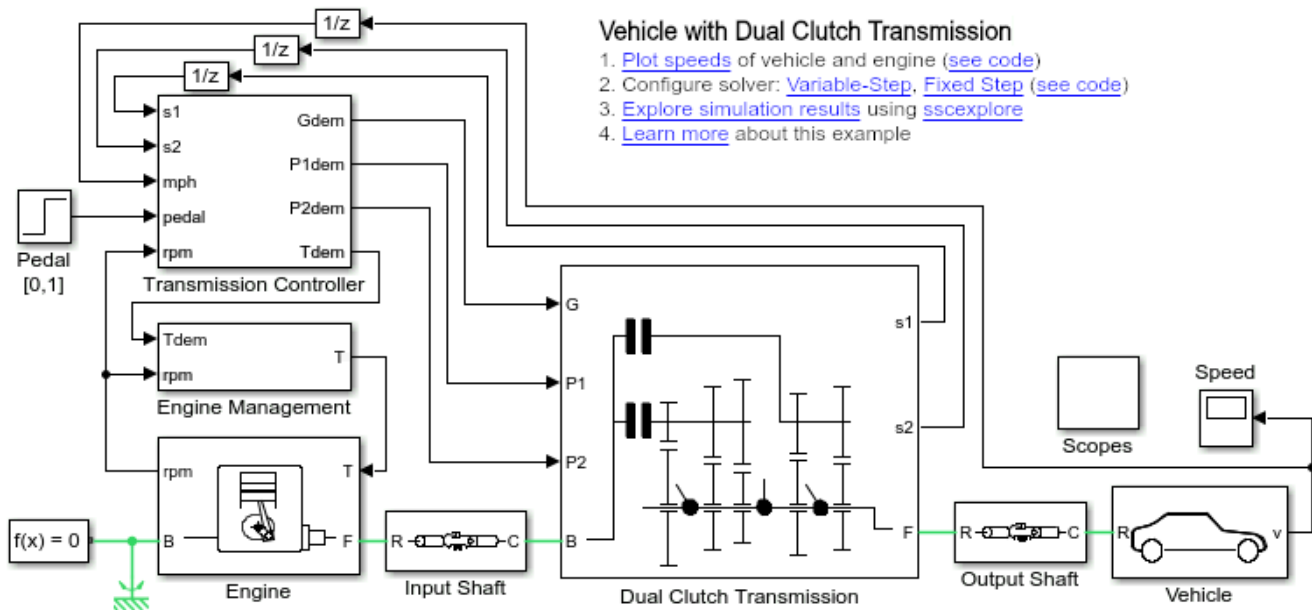
“Automatic Transmission with a Dual Clutch” on page 12-8

Automatic Transmission with a Dual Clutch

The “Vehicle with Dual Clutch Transmission” on page 18-155 example model illustrates important points about both physical and control design modeling using the Simscape and Simscape Driveline environment and libraries.

The model represents an automatic transmission with two clutches. In an automatic transmission, an engine management subsystem decides when to change gear ratio and what the next gear ratio is. The pedal deflection imposed by a vehicle driver is converted into a demanded engine torque. The torque demanded and the current forward vehicle speed together determine which gear ratio the transmission switches to before it actually switches (gear preselection). By gradually lowering the clutch pressure, the transmission control system smoothly unlocks and disengages the clutch configuration for the current gear ratio. At the same time, the control system gradually raises the clutch pressures to achieve the new gear ratio by engaging and locking the new clutch configuration.

- The physical components of the transmission, from the engine, gears, and clutches, to the vehicle body and tire, are modeled using Simscape Driveline blocks, with physical ports and connections.
- The algorithmic control of the transmission, including the gear-switching and transmission control, is modeled using normal Simulink blocks, with signal ports, signal lines, and enabled subsystems.



Vehicle with Dual Clutch Model

Predefined Simulation Options

The model is also set up to allow you to switch between two common simulation configurations. To configure the model to simulate with a variable-step global solver or a fixed-step local solver, click the links in the description.

You can directly adjust all the solver options by opening the model Configuration Parameters and the network Solver Configuration property inspector.

See Also

More About

- “Model Realistic Clutch Pressure Signals” on page 12-7

Control Vehicle Velocity

Control Vehicle Throttle Input Using a Powertrain Blockset Driver

In this section...

“Open-Loop Simulation Using the Test - Hill Subcomponent Block” on page 13-2

“Closed-Loop Simulation Using a Longitudinal Driver Block” on page 13-4

“Simulation Comparison” on page 13-8

This example shows how to control throttle input to a Simscape Driveline vehicle model using a Powertrain Blockset Longitudinal Driver block. You add the driver to an open-loop model that uses a Signal Builder block for feedforward control. Adding the driver allows you to model closed-loop control by supplying a reference velocity and a feedback loop.

Open-Loop Simulation Using the Test - Hill Subcomponent Block

In the open-loop simulation, a subcomponent block is used to ramp up the throttle. Simulate the model to see the open-loop response.

- 1 Open the model. At the MATLAB command prompt, enter this code.

See Code

```
%% Model information
modelName = 'VehicleWithFourWheelDriveExample';

%% Run the original model
open_system(modelName)
```

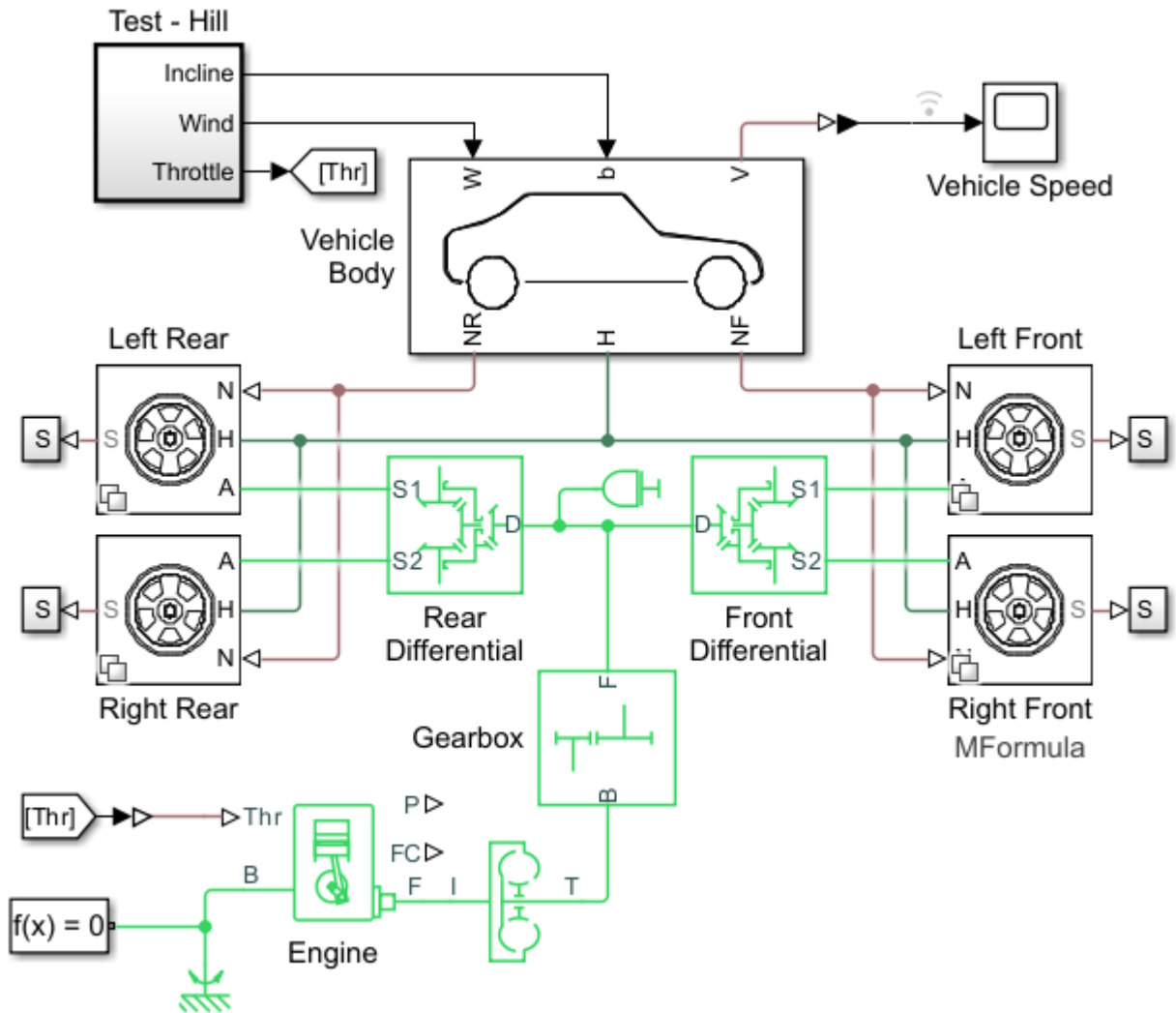
- 2 Enable the signal that goes to the **Motor RPM** scope block for Simulink data logging and viewing with the Simulation Data Inspector.

See Code

```
%% PS-S Converter1 information
pStoSConverter1Name = 'PS-Simulink Converter1';
pStoSConverter1Path = [modelName, '/', pStoSConverter1Name];
pStoSConverter1PortHandles = get_param(pStoSConverter1Path, 'PortHandles');
pStoSConverter1Inport = pStoSConverter1PortHandles.LConn(1,1);
pStoSConverter1Outport = pStoSConverter1PortHandles.Outport(1,1);

%% Enable the signal that goes to the Vehicle Speed scope block for
% Simulink(TM) data logging and viewing with the Simulation Data Inspector
set_param(pStoSConverter1Outport, 'DataLogging', 'on')
```

The logging badge  marks the signal in the model.



- 3 Increase simulation time to get steady-state results. Simulate the model.

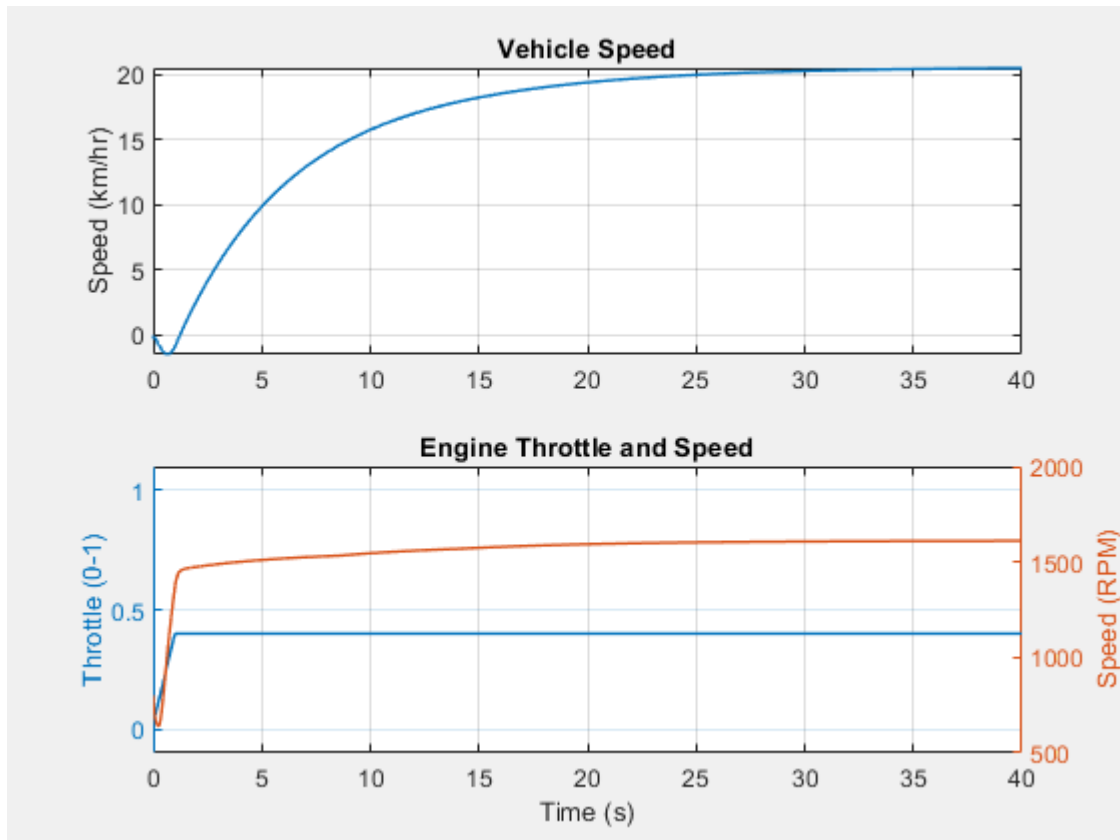
See Code

```

%% Increase simulation time
set_param(modelName, 'StopTime', '30')

%% Simulate the model
sim(modelName)
% Plot the vehicle speed, the engine speed, and the throttle input.
VehicleWithFourWheelDrivePlot1Speed

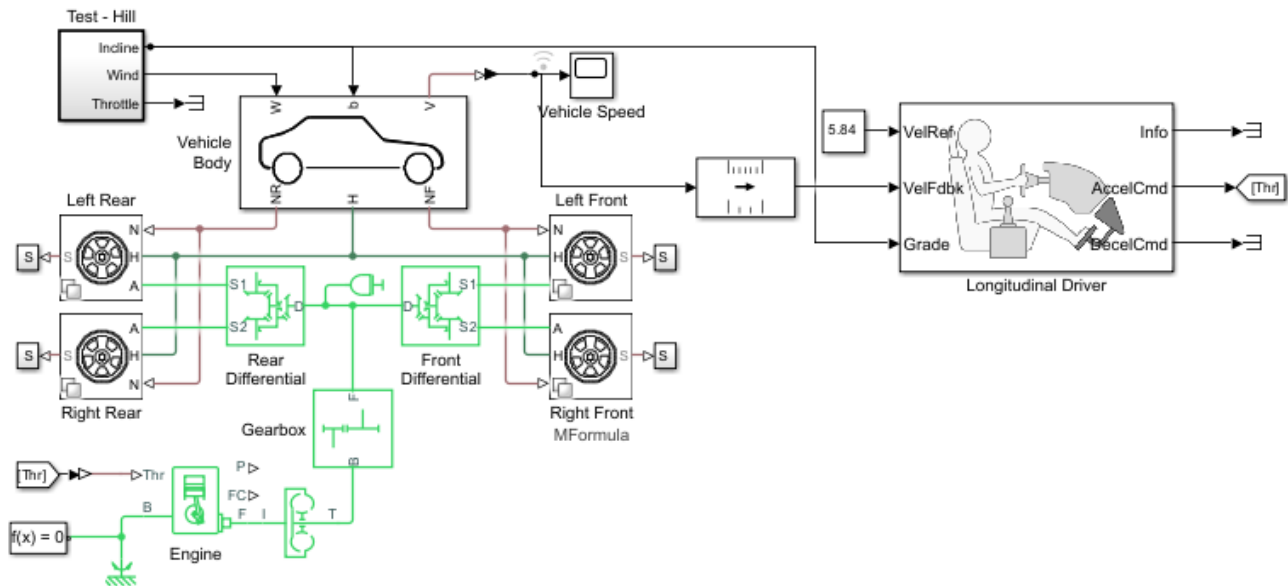
```



Even though the throttle input is nonzero at simulation time 0, the vehicle rolls down the hill at first because the input is too small to overcome the gravitational force of the incline. When the input is large enough, the vehicle accelerates up the hill and settles at a velocity of ~20 km/h.

Closed-Loop Simulation Using a Longitudinal Driver Block

To control throttle input, add a Powertrain Blockset Longitudinal Driver block.



- 1 Add a Longitudinal Driver block to the model.
 - a Expand the model window to fit the Longitudinal Driver block.

See Code

```
%% Update the model canvas
modelWindowNewLocation = [47 100 1072 723];

% Increase the size of the model window
set_param(modelName, 'location', modelWindowNewLocation);
```

- b Add the Longitudinal Driver block.

See Code

```
%% Longitudinal Driver block name, path, %
% and position information
longDriverName = 'Driver';
longDriverLibPath = 'autolibscenario/Longitudinal Driver';
longDriverPath = [modelName, '/Longitudinal Driver'];
longDriverPos = [710 15 905 135];

% Add a driver block
add_block(longDriverLibPath, longDriverPath, ...
          'position', longDriverPos)

% Longitudinal Driver block port information
longDriverPortHandles = get_param(longDriverPath, 'PortHandles');
longDriverPortVelRef = longDriverPortHandles.Inport(1,1);
longDriverPortVelFdbk = longDriverPortHandles.Inport(1,2);
longDriverPortGrade = longDriverPortHandles.Inport(1,3);
longDriverPortInfo = longDriverPortHandles.Outport(1,1);
```

- longDriverPortAccelCmd = longDriverPortHandles.Outport(1,2);
 longDriverPortDecelCmd = longDriverPortHandles.Outport(1,3);
- c Change the source of throttle input from the Test - Hill block **Throttle** port to the Longitudinal Driver block **AccelCmd** port and terminate the unconnected **Throttle** port with a Terminator block.

See Code

```
%% Test - Hill block information
signalTestName = 'Test - Hill';
signalTestPath = [modelName, '/', signalTestName];
signalTestPortHandles = get_param(signalTestPath, 'PortHandles');
signalTestPortIncline = signalTestPortHandles.Outport(1,1);
signalTestPortWind = signalTestPortHandles.Outport(1,2);
signalTestPortThrottle = signalTestPortHandles.Outport(1,3);

%% Throttle Goto block information
goToThrottleName = 'Goto';
goToThrottlePath = [modelName, '/', goToThrottleName];
goToThrottlePortHandle = get_param(goToThrottlePath, 'PortHandles');
goToThrottleInport = goToThrottlePortHandle.Inport(1,1);
goToThrottleNewPos = [950 66 985 84];

% Delete the connection line from the Signal Builder to the Throttle Goto
delete_line(modelName, signalTestPortThrottle, goToThrottleInport)

% Move the throttle Goto to the Longitudinal Driver
set_param(goToThrottlePath, 'position', goToThrottleNewPos)

% Add a connection line from the Longitudinal Driver AccelCmd
% port to the Throttle goto
add_line(modelName, longDriverPortAccelCmd, goToThrottleInport)

%% Terminator0 Block Information
% Terminator0 block name, path, and position information
terminator0Name = 'Terminator0';
terminator0LibPath = 'simulink/Sinks/Terminator';
terminator0Path = [modelName, '/', terminator0Name];
terminator0Pos = [200 5 220 25];

% Add Terminator0 block
add_block(terminator0LibPath, terminator0Path, ...
  'position', terminator0Pos)

% Terminator0 port information
terminator0PortHandle = get_param(terminator0Path, 'PortHandles');
terminator0Inport = terminator0PortHandle.Inport(1,1);

% Connect Longitudinal Driver AccelCmd Outport to the Thr Goto inport
add_line(modelName, signalTestPortThrottle, terminator0Inport)
```

- d Terminate the **Info** and **DeclCmd** outputs on the Longitudinal Driver block

See Code

```
%% T1
% Terminator1 block name, path, and position
% information
terminator1Name = 'Terminator1';
terminator1LibPath = 'simulink/Sinks/Terminator';
terminator1Path = [modelName, '/', terminator1Name];
terminator1Pos = [955 105 975 125];

% Add Terminator1 block
add_block(terminator1LibPath, terminator1Path, ...
  'position', terminator1Pos)

% Terminator1 port information
terminator1PortHandle = get_param(terminator1Path, 'PortHandles');
```

```

terminator1Inport = terminator1PortHandle.Inport(1,1);

% Connect Longitudinal Driver DecelCmd Output
%   to Terminator1 inport
add_line(modelName,longDriverPortDecelCmd,terminator1Inport)

%% T2
% Terminator2 block name, path, and position information
terminator2Name = 'Terminator2';
terminatorLibPath = 'simulink/Sinks/Terminator';
terminator2Path = [modelName,'/',terminator2Name];
terminator2Pos = [955 25 975 45];

% Add Terminator2 block
add_block(terminatorLibPath,terminator2Path,...
    'position',terminator2Pos)

% Terminator2 port information
terminator2PortHandle = get_param(terminator2Path,'PortHandles');
terminator2Inport = terminator2PortHandle.Inport(1,1);

% Connect Longitudinal Driver Info Output to Terminator2 inport
add_line(modelName,longDriverPortInfo,terminator2Inport)

```

- e Input a 5.84 reference velocity to the Longitudinal Driver block **VelRef** port using a Constant block.

See Code

```

%% Reference Velocity
ref_vel = '5.84';
constantLibPath = 'simulink/Commonly Used Blocks/Constant';
referenceVelocityName = 'Constant';
referenceVelocityPath = [modelName,'/',referenceVelocityName];
referenceVelocityPos = [655 20 685 50];
add_block(constantLibPath,referenceVelocityPath,...
    'position',referenceVelocityPos)
set_param(referenceVelocityPath,'Value',ref_vel)

% Reference Velocity port information
referenceVelocityPortHandle = get_param(referenceVelocityPath,'PortHandles');
referenceVelocityOutput = referenceVelocityPortHandle.Output(1,1);

add_line(modelName,referenceVelocityOutput,longDriverPortVelRef)

```

- f Input the incline angle signal from the Signal Builder block to the Longitudinal Driver block by connecting the **Incline** output to the **Grade** inport.

See Code

```

%% Connect incline output port to Grade input port
add_line(modelName,signalTestPortIncline,...
    longDriverPortGrade,'autorouting','on')

```

- g Input the velocity feedback signal to the Longitudinal Driver block using a Unit Conversion block to convert from km/hr to m/s.

See Code

```

%% Unit Conversion
% UnitConvert0 block name, path, and position information
unitConvert0Name = 'Unit Conversion';
unitConvert0LibPath = 'simulink/Signal Attributes/Unit Conversion';
unitConvert0Path = [modelName,'/',unitConvert0Name];
unitConvert0Pos = [565 54 635 96];

% Add block
add_block(unitConvert0LibPath,unitConvert0Path,...
    'position',unitConvert0Pos)

```

```

%% UnitConvert0 port information
unitConvert0PortHandle = get_param(unitConvert0Path,'PortHandles');
unitConvert0Inport = unitConvert0PortHandle.Inport(1,1);
unitConvert0Outport = unitConvert0PortHandle.Outport(1,1);

%% Connect Longitudinal Driver AccelCmd Outport to Goto Throttle inport
add_line(modelName,unitConvert0Outport,...
    longitudinalDriverPortVelFdbk,'autorouting','on')

%% Connect PS-S Converter Outport to Longitudinal Driver AccelCmd Inport
add_line(modelName,pStoSConverter1Outport,...
    unitConvert0Inport,'autorouting','on')

```

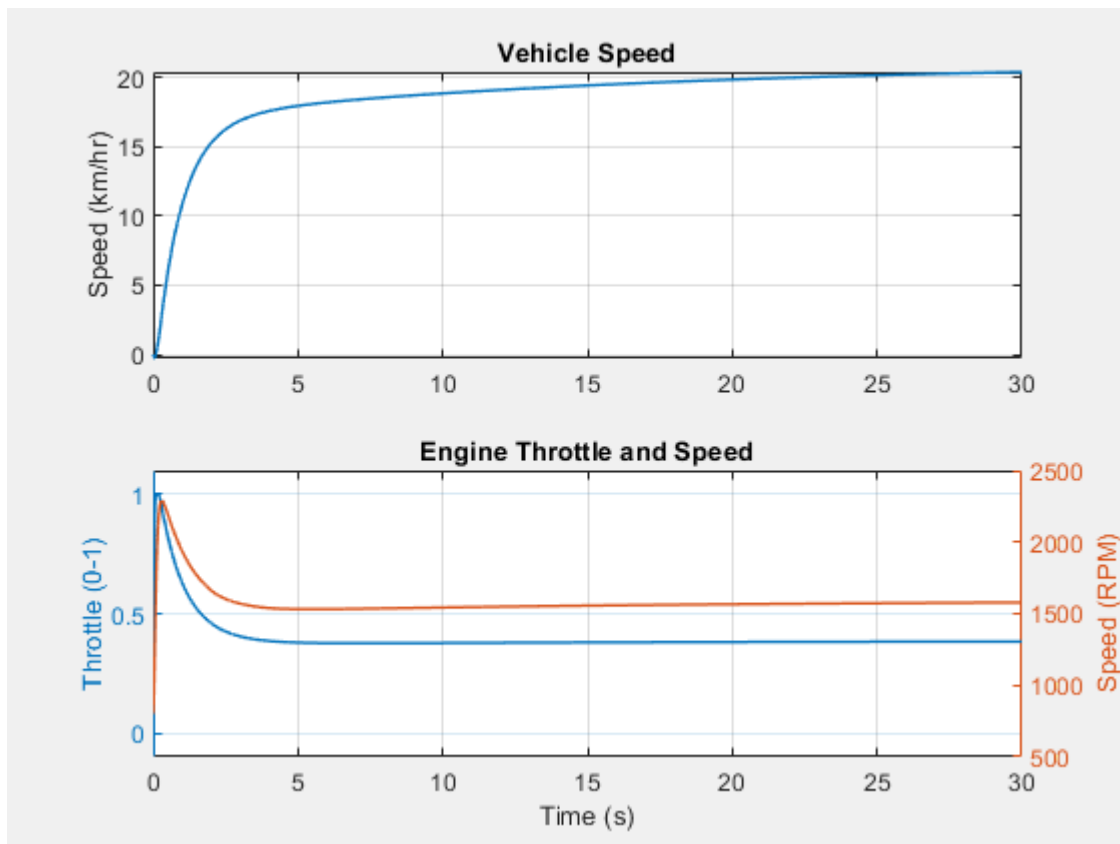
- 2 Simulate the closed-loop model using the Simple Tire blocks and plot the results.

See Code

```

%% Simple
VehicleWithFourWheelDriveSetTires(bdroot,'Simple');
sim(modelName)
% Plot the vehicle speed, the engine speed, and the Throttle input.
VehicleWithFourWheelDrivePlot1Speed

```



The drive block increases the throttle input rapidly at the beginning of simulation due to the difference between the velocity feedback and reference signals.

Simulation Comparison

Compare the open and closed-loop results using the Simulation Data Inspector.

See Code

```

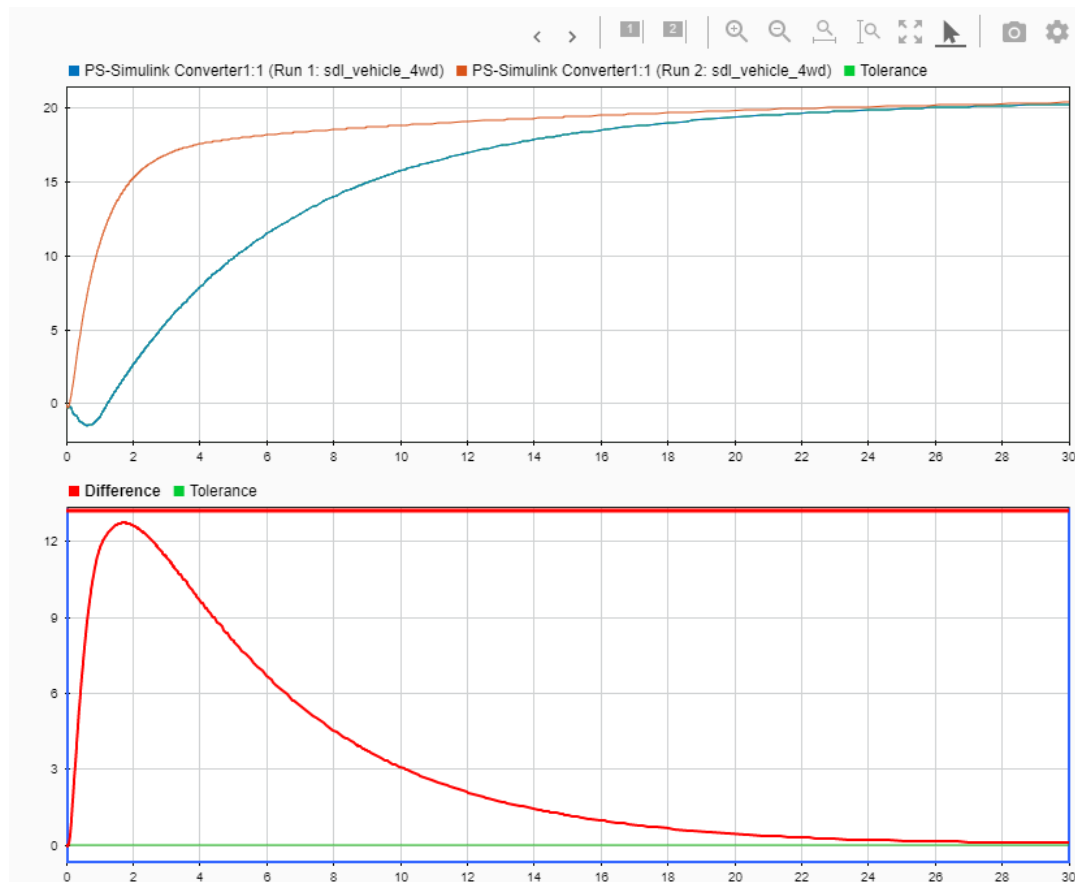
%% Get Simulation (TM) Data Inspector run IDs for
%   the last two runs
runIDs = Simulink.sdi.getAllRunIDs;
runBaseline = runIDs(end - 1);
runSimple = runIDs(end);

% Open the Simulation Data Inspector
Simulink.sdi.view

compBaselinePartition = Simulink.sdi.compareRuns(runBaseline,...
    runSimple);

```

To see the results in the Simulation Data Inspector, click the **Compare** icon and then, under **Filter Comparisons**, click **PS-Simulink Converter1:1**.



The first plot overlays the results from the open and closed-loop simulations. It shows how much faster the controlled vehicle goes to steady state.

The second plot shows the numerical difference in the results from the two simulations. It shows how much the two signals differ at the beginning of the simulation and how they eventually reach the same steady state.

You can also examine the results for other tire blocks.

See Code for Magic Formula Tire

```
%% Magic
VehicleWithFourWheelDriveSetTires(bdroot, 'Mformula');
sim(modelName)
% Plot the vehicle speed, the engine speed, and the throttle input.
VehicleWithFourWheelDrivePlot1Speed
```

See Code for Friction Parameterized Tire

```
%% Friction
VehicleWithFourWheelDriveSetTires(bdroot, 'Friction');
sim(modelName)
% Plot the vehicle speed, the engine speed, and the throttle input.
VehicleWithFourWheelDrivePlot1Speed
```

See Also

[Tire \(Friction Parameterized\)](#) | [Tire \(Magic Formula\)](#) | [Tire \(Simple\)](#) | [Constant](#) | [Terminator](#) | [Unit Conversion](#)

Related Examples

- “Vehicle with Four-Wheel Drive” on page 18-159

More About

- “Compare Simulation Data”

Drivetrain Disturbances

- “Model Drivetrain Noise” on page 14-2
- “Model and Detect Drivetrain Faults” on page 14-10

Model Drivetrain Noise

This example shows how to inject a fault into a drivetrain using a Torque Noise Source block. Injecting noise into your model allows you to predict how your actual physical system responds when it experiences environmental or internal disturbances. It also allows you to test the robustness and responsiveness of your control system.

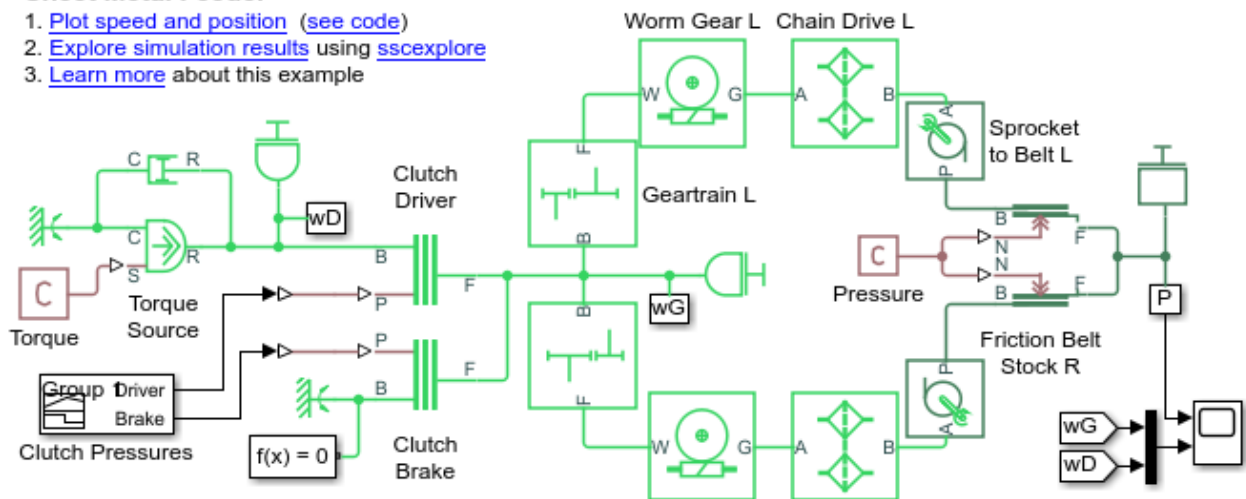
In the example, the noise is injected into one of the sprockets on a metal sheet feeder that is moving a piece of stock. Although you can perform most of the steps in this example using tools that the Simulink and Simscape Driveline user interfaces provide, programmatic commands are supplied. You can combine the programmatic commands to create a script for parameter sweeps.

- 1 Open the model. At the MATLAB command prompt, enter:

```
openExample('sdl/SheetMetalFeederExample')
model = 'SheetMetalFeeder';
```

Sheet Metal Feeder

1. [Plot speed and position](#) ([see code](#))
2. [Explore simulation results](#) using [sscxplore](#)
3. [Learn more](#) about this example



- 2 Simulate the model and plot the results.

Script for Generating and Plotting Simulation Results

```
%% Simulate
sim(model)

%% Define the data
simlog1 = simlog_SheetMetalFeeder;
time = simlog1.Sensor_Stock.Motion_Sensor.x.series.time;
sheetPosition = simlog1.Sensor_Stock.Motion_Sensor.x.series.values;
sprocketTorque = simlog1.Sprocket_to_Belt_R.t.series.values;

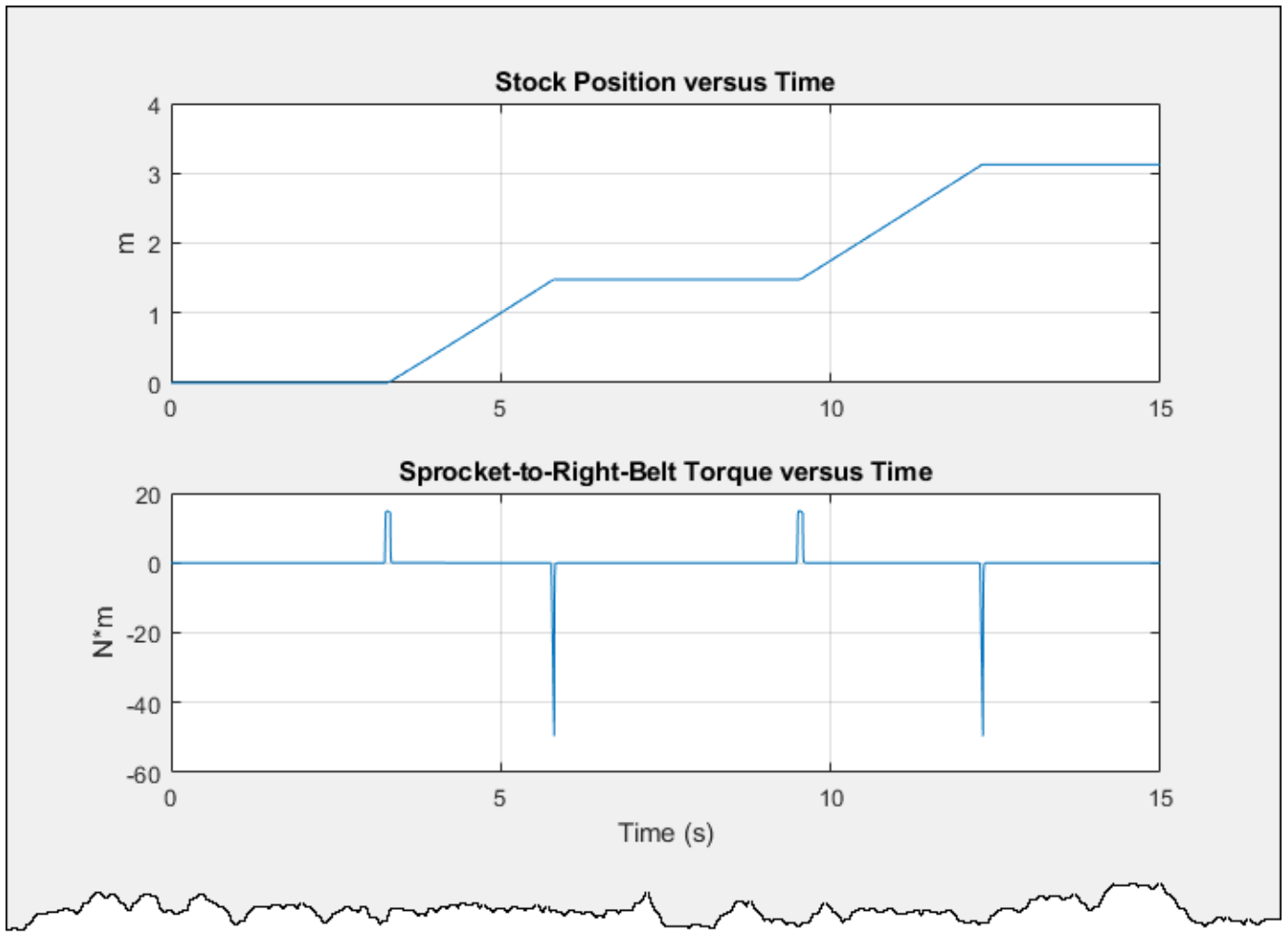
%% Create figure
figure1 = figure('NumberTitle','off',...
```



```
'Name','Simscape Results: SheetMetalFeeder_noise',...
'OuterPosition',[565 52 733 822]);

%% Create subplot 1: Sprocket to Belt L Torque versus Time
% Create axes for
axes1 = axes('Parent',figure1,...
'Position',[0.13 0.709 0.775 0.216]);
hold(axes1,'on')
box(axes1,'on')
grid(axes1,'on')
% Create title & axis labels
title('Stock Position versus Time')
% xlabel('Time (s)')
ylabel('m')
% Create plot
plot(time,sheetPosition,'Parent',axes1)

%% Create subplot 2: Sprocket to Belt R Torque versus Time
% Create axes
axes2 = axes('Parent',figure1,...
'Position',[0.13 0.409 0.775 0.2156]);
hold(axes2,'on')
box(axes2,'on')
grid(axes2,'on')
% Create title & axis labels
title('Sprocket-to-Right-Belt Torque versus Time')
xlabel('Time (s)')
ylabel('N*m')
% Create plot with linked axes
plot(time,sprocketTorque,'Parent',axes2)
linkaxes([axes1,axes2],'x')
```



3 Add, configure, connect, and arrange these blocks as shown:

- Step block — Specify 7.5 for the **Step time** parameter and 300 for the **Final value** parameter.
- Simulink-PS Converter block
- Mechanical Rotational Reference block
- Torque Noise Source block — For the **Sample time**, specify 1e-1, for **Repeatability**, select Specify seed, and for **Seed** specify 0.

Script for Adding, Configuring, Connecting, and Arranging Blocks

```
% Expand the model window to make room for new blocks
set_param(model,'Location',[108 73 881 682])

% Add and configure Step block
% Define block
stepPath = [model,'/Step'];
stepLib = 'simulink/Sources/Step';
stepPosition = [-195 205 -165 235];
stepTime = '7.5';
stepValue = '300';
% Add block
add_block(stepLib,stepPath,'Position',stepPosition)
% Configure Step block
set_param(stepPath,'Time',stepTime,...
```

```

    'After',stepValue)
% Check block configuration
open_system(stepPath)
pause(3);
close_system(stepPath)
% Get output port
stepPortHandle = get_param(stepPath,'PortHandles');
stepOutputport = stepPortHandle.Outputport;

%% Add and configure S-PS Converter block
% Define block
sPSConvPath = [model,'/Simulink-PS Converter'];
sPSConvLib = ...
    'nesl_utility/Simulink-PS Converter';
sPSConvPathPosition = [-115 205 -85 235];
% Add block
add_block(sPSConvLib,sPSConvPath,'Position',sPSConvPathPosition)
% Get output port
sPSConvPortHandle = get_param(sPSConvPath,'PortHandles');
sPSConvInport = sPSConvPortHandle.Inport;
sPSConvConPort = sPSConvPortHandle.RConn(1,1);

%% Add Mechanical Rotational Reference block
% Define block
mechRotRefPath = [model,'/Mechanical Rotational Reference'];
mechRotRefLib = ...
    'fl_lib/Mechanical/Rotational Elements/Mechanical Rotational Reference';
mechRotRefPosition = [-40 255 -20 275];
% Add block
add_block(mechRotRefLib,mechRotRefPath,'Position',mechRotRefPosition)
% Get output port
mechRotRefPortHandle = get_param(mechRotRefPath,'PortHandles');
mechRotRefConPort = mechRotRefPortHandle.LConn(1,1);

%% Add and configure Torque Noise Source block
% Define block
torqueNoisePath = [model,'/Torque Noise Source'];
torqueNoiseLib = 'sdl_lib/Sources/Torque Noise Source';
torqueNoisePosition = [25 210 65 250];
% Add block
add_block(torqueNoiseLib,torqueNoisePath,'Position',torqueNoisePosition)
% Get output port
torqueNoisePortHandle = get_param(torqueNoisePath,'PortHandles');
torqueNoiseConPort1 = torqueNoisePortHandle.LConn(1,1);
torqueNoiseConPort2 = torqueNoisePortHandle.LConn(1,2);
torqueNoiseConPort3 = torqueNoisePortHandle.RConn(1,1);
% Configure block
set_param(torqueNoisePath, 'sample_time','1e-1',...
    'repeatability','3')
% Check block configuration
open_system(torqueNoisePath)
pause(3);
close_system(torqueNoisePath)

%% Get block points for connecting as a branched line
torqueNoisePortPoints = get_param(torqueNoisePath,'PortConnectivity');
[torqueNoiseLConnPoints1,torqueNoiseLConnPoints2,torqueNoiseRConnPoints]...
    = torqueNoisePortPoints.Position;
sprocketPath = [model,'/Sprocket to Belt R'];
sprocketPortHandle = get_param(sprocketPath,'PortHandles');
sprocketConPort1 = sprocketPortHandle.RConn(1,1);
sprocketPortPoints = get_param(sprocketPath,'PortConnectivity');
[sprocketLConnPoints,sprocketRConnPoints] = sprocketPortPoints.Position;

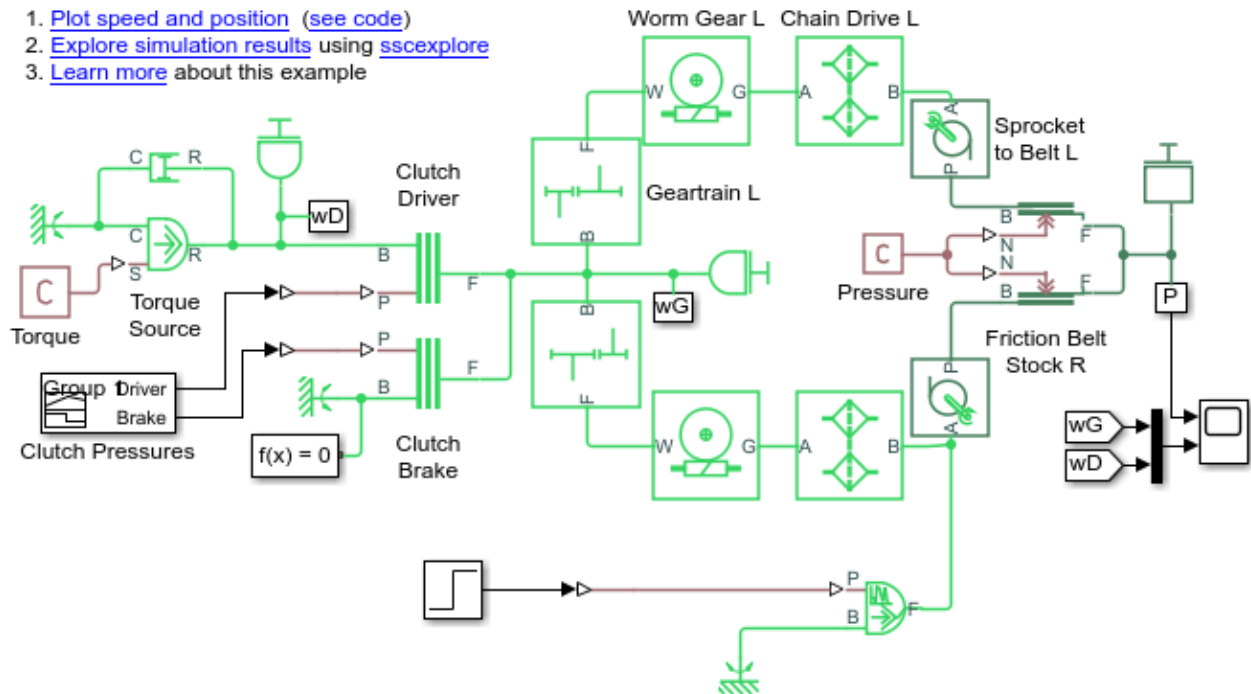
%% Connect blocks
add_line(model,stepOutputport,sPSConvInport)
add_line(model,sPSConvConPort,torqueNoiseConPort1)
add_line(model,mechRotRefConPort,torqueNoiseConPort2)
add_line(model,[sprocketLConnPoints;torqueNoiseRConnPoints])

%% Hold to check connections
pause(5);

```

Sheet Metal Feeder

1. [Plot speed and position](#) (see code)
2. [Explore simulation results](#) using [sscexplore](#)
3. [Learn more](#) about this example



4. Simulate the model and plot the results.

Script for Generating and Plotting Simulation Results

```

%% Simulate model
sim(model)

%% Define the datalog variable
simlog2 = simlog_SheetMetalFeeder;
time = simlog2.Sensor_Stock.Motion_Sensor.x.series.time;
sheetPosition = ...
    simlog2.Sensor_Stock.Motion_Sensor.x.series.values;
sprocketTorque = simlog2.Sprocket_to_Belt_R.t.series.values;
noiseTorque = simlog2.Torque_Noise_Source.t.series.values;

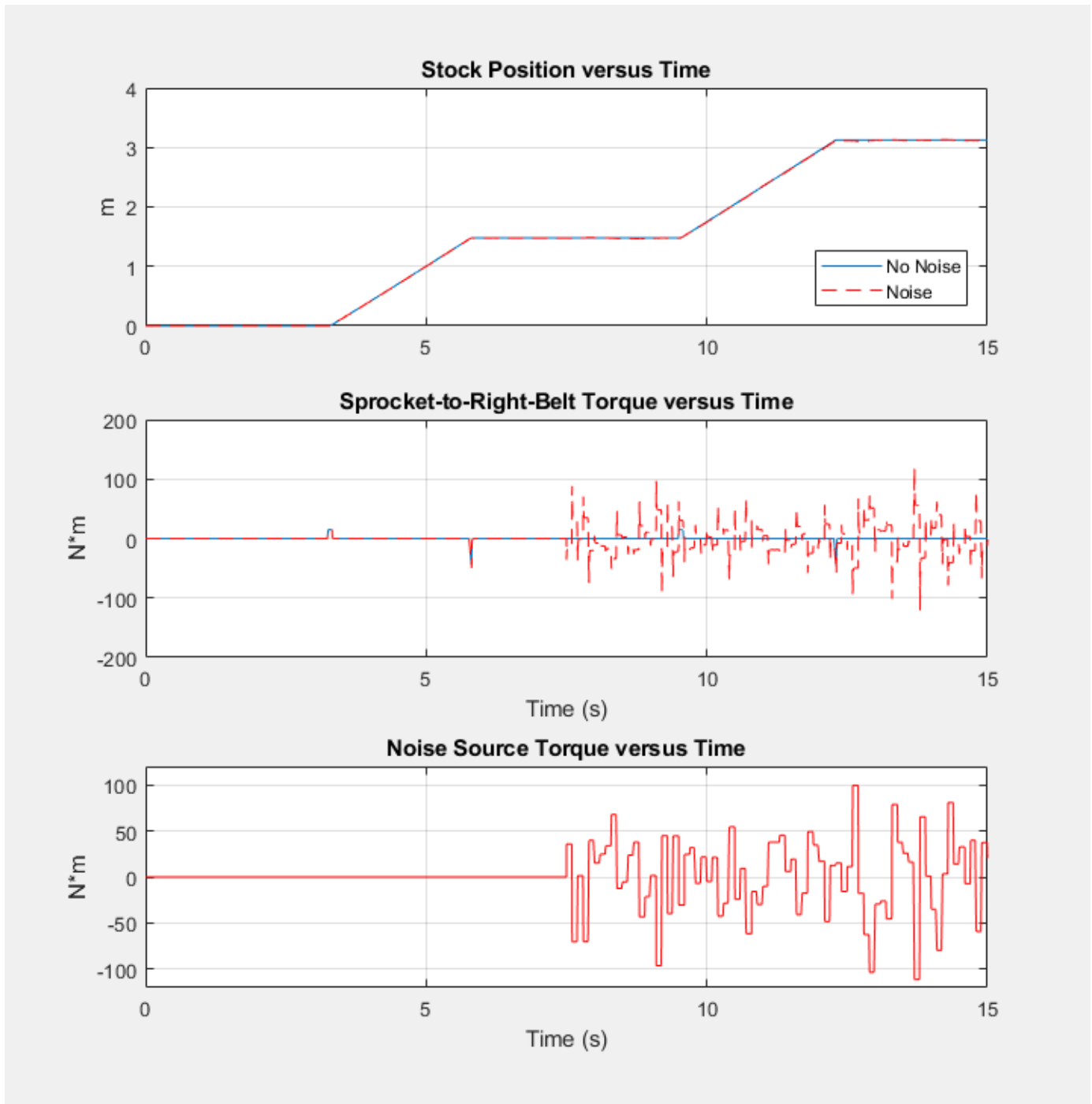
%% Update plots 1 & 2
figure1;
hold on
axes1;
hold on
plot(time, sheetPosition, 'Parent', axes1, 'LineStyle', '--', 'Color', 'r')
axes2;
hold on
plot(time, sprocketTorque, 'Parent', axes2, 'LineStyle', '--', 'Color', 'r')

%% Create subplot 3: Sprocket to Belt R Torque versus Time
% Create axes
axes3 = axes('Parent', figure1, ...
    'Position', [0.13 0.11 0.775 0.200]);
hold(axes3, 'on')
box(axes3, 'on')
grid(axes3, 'on')

% Create title and axis labels
title('Noise Source Torque versus Time')
xlabel('Time (s)')
ylabel('N*m')
linkaxes([axes1, axes2, axes3], 'x')

```

```
% Create plot
plot(time,noiseTorque,'Parent',axes3,'Color','r')
legend(axes1,'No Noise','Noise','Location','southeast');
ylim(axes3,[-120 120]);
```

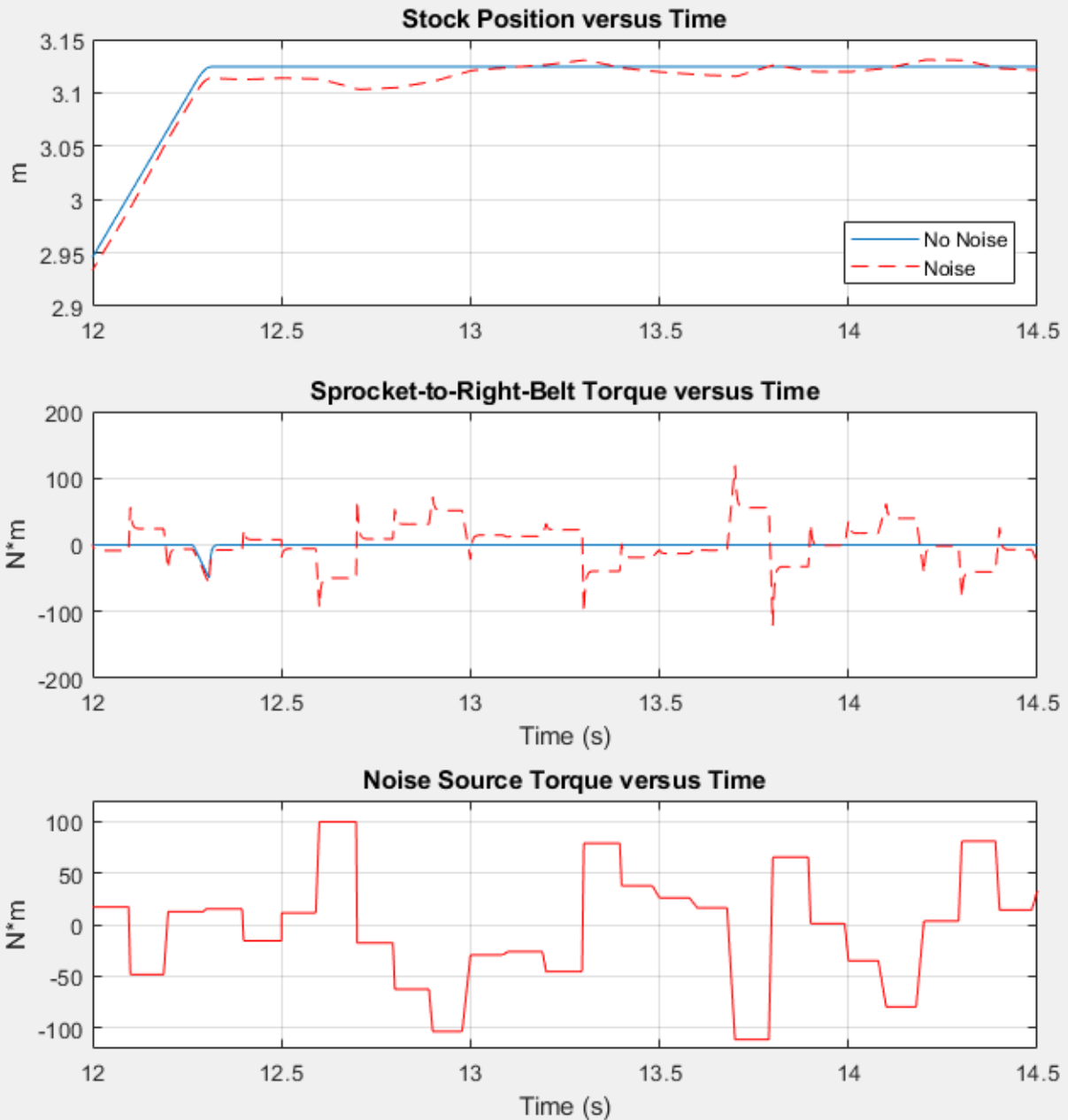


The noise injected at simulation time, $t = 7.5$ seconds introduces torque disturbances.

- 5 Zoom in to see the effects of the disturbance.

Script for Zooming In

```
% Zoom to examine data at peak noise
xlim(axes3,[12.0 14.50]);
ylim(axes3,[-120 120]);
```



When the noise source torque exceeds ± 50 N*m, it most significantly effects the torque applied by the sprocket and, therefore, the position of the stock.

See Also

[Torque Noise Source](#) | [Simulink-PS Converter](#) | [Step](#) | [Mechanical Rotational Reference](#) | [Sinusoidal Torque Source](#) | [Rotational Damper](#)

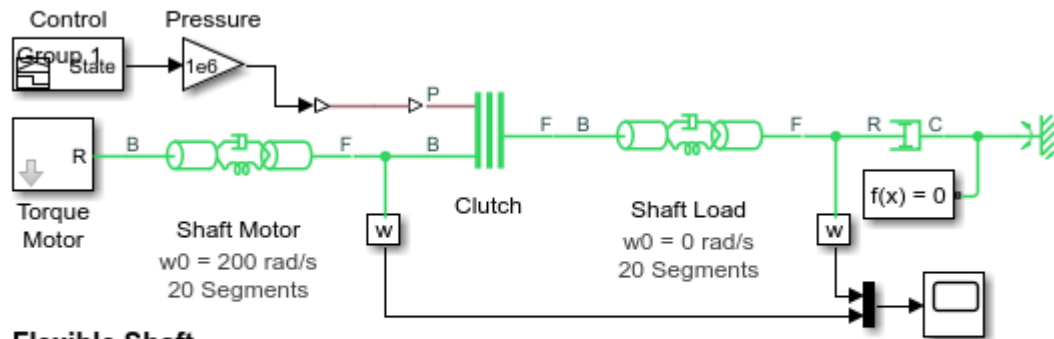
Model and Detect Drivetrain Faults

This example shows how to detect and respond to a fault in a drivetrain using a Rotational Damper block. The Rotational Damper block allows you to specify the damping coefficient as a function of temporal or behavioral triggers. You can program the damping coefficient to change at a particular time in the simulation or when the number of shocks for a given acceleration exceeds a limit to model fault behavior. Fault modeling allows you to predict how your actual physical system responds when it experiences real faults. It also allows you to test the robustness and responsiveness of your control system.

In the example, the fault is detected by a damper that is attached to a flexible shaft. Although you can perform most of the steps in this example using the tools that the Simulink and Simscape Driveline user interfaces provide, scripts are supplied. You can combine the scripts into a larger script for parameter sweeps.

- 1 Open the model. At the MATLAB command prompt, enter:

```
openExample('sdl/ShaftWithTorsionalFlexibilityExample')
model = 'ShaftWithTorsionalFlexibility';
```



Flexible Shaft

1. [Plot shaft speed](#) (see code)
2. [Plot twist](#) in shaft compliance elements (see code)
3. [Explore simulation results](#) using `sccxplorer`
4. [Learn more](#) about this example



This model contains two flexible aluminum shafts modeled using a lumped parameter approach. A motor drives the motor shaft. A viscous damper is connected to the load shaft. The viscous damper is represented by a Rotational Damper block from the **Simscape > Foundation Library > Mechanical > Rotational Elements** library. The **Foundation Library** Rotational Damper block is not able to detect or respond to faults.

- 2 Simulate the model and plot the results.

Script for Generating and Plotting Simulation Results

```
%% Simulate Model
sim(model)

%% Get simulation results
simlog01 = simlog_ShaftWithTorsionalFlexibility;
simlog_t = simlog01.Clutch.P.series.time;
simlog_pClutch = simlog01.Clutch.P.series.values('Pa');
simlog_wMotor = simlog01.Shaft_Motor.F.w.series.values('rad/s');
simlog_wLoad = simlog01.Shaft_Load.F.w.series.values('rad/s');
```



```

%% Plot results
X1 = simlog_t;
YMatrix1 = [simlog_wMotor simlog_wLoad];
Y1 = simlog_pClutch;

% Create figure
figure1 = figure('Name', 'ShaftWithTorsionalFlexibility', ...
    'OuterPosition', [107 336 733 822]);

% Create axes
axes1 = axes('Parent', figure1, ...
    'Position', [0.13 0.709 0.775 0.216]);
hold(axes1, 'on');

% Activate the left side of the axes
yyaxis(axes1, 'left');

% Create multiple lines using matrix input to plot
plot1 = plot(X1, YMatrix1, 'LineWidth', 2);
set(plot1(1), 'DisplayName', 'Motor Shaft', 'Color', 'k');
set(plot1(2), 'DisplayName', 'Load Shaft', 'Color', 'b');

% Create ylabel
ylabel('Speed (rad/s)');

% Comment/uncomment the following line to remove/
% preserve the Y-limits of the axes
ylim(axes1, [-100 250]);

% Set the remaining axes properties
set(axes1, 'YColor', [0 0.447 0.741]);

% Activate the right side of the axes
yyaxis(axes1, 'right');

% Create plot
plot(X1, Y1, 'DisplayName', 'Clutch Pressure', 'LineWidth', 2, 'Color', 'r');

% Create ylabel
ylabel('Pressure (Pa)');

% Comment/uncomment the following line to remove/
% preserve the Y-limits of the axes
ylim(axes1, [0 2000000]);

% Set the remaining axes properties
set(axes1, 'YColor', [0.85 0.325 0.098]);
% Create xlabel
xlabel('Time (s)');

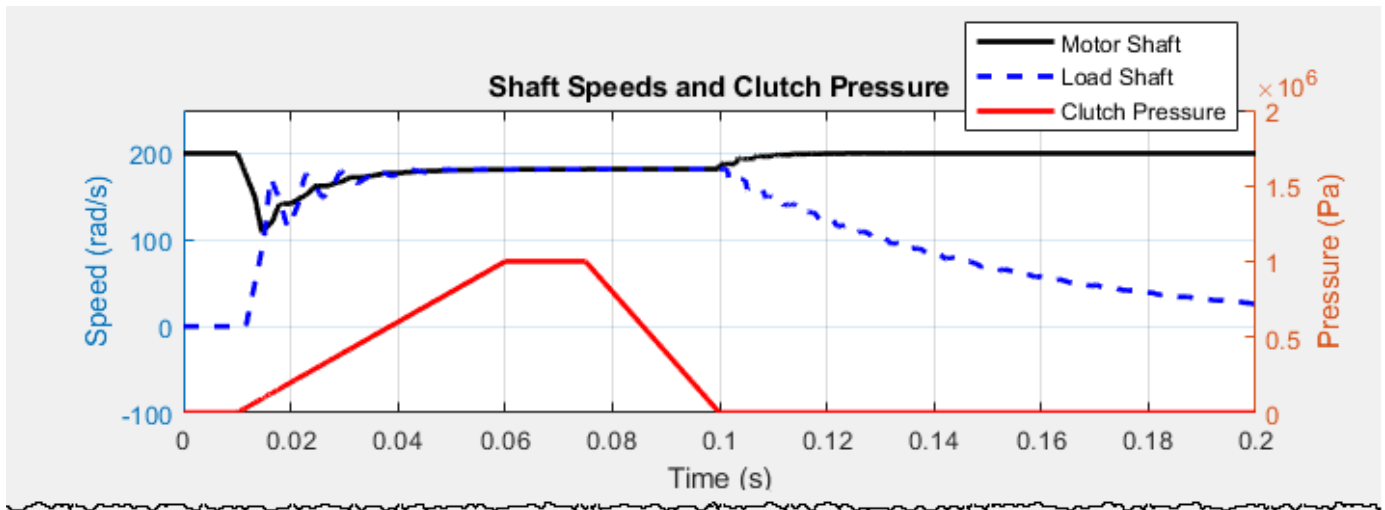
% Create title
title('Shaft Speeds and Clutch Pressure');

% Uncomment the following line to preserve the X-limits of the axes
% xlim(axes1, [0 0.2]);

% Set the remaining axes properties
set(axes1, 'LineStyleOrderIndex', 2);
box(axes1, 'on');
grid(axes1, 'on');

% Create legend
legend1 = legend(axes1, 'show');
% set(legend1, 'Location', 'best');
set(legend1, ...
    'Position', [0.6944 0.9125 0.1982 0.07270]);

```



At the start of the simulation, the clutch is unlocked and the driven shaft is free. The initial velocity of the motor shaft is the specified 200 rad/s and the system starts at steady state. The oscillations triggered by the engaging and disengaging of the clutch are due to the flexibility in the shafts.

- 3 Replace the Simscape damper with the Simscape Driveline Rotational Damper, which is in the **Simscape > Driveline > Couplings & Drives > Springs & Dampers** library. Label the new block **Faultable Damper**.

Script for Replacing the Rotational Damper Block

```

%% Replace Rotational Damper from Foundation Library with
% Faultable Rotational Damper from the Simscape Driveline Library

% Define Unfaultable Damper Block
foundationDamper = [model, '/Damping Bearing'];

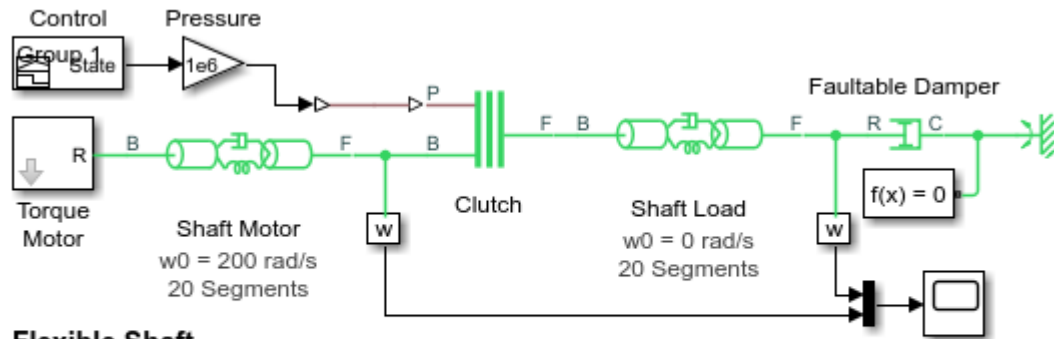
% Get Unfaultable Damper Block Damping Coefficient
dampingCoefficient = get_param(foundationDamper, 'D');

% Get Unfaultable Damper Block Position
damperPosition = get_param(foundationDamper, 'Position');

% Delete Unfaultable Damper Block
delete_block(foundationDamper)

% Add Faultable Damper Block
faultableDamperLib = ...
    'sd_lib/Couplings & Drives/Springs & Dampers/Rotational Damper';
faultableDamperPath = [model, '/Faultable Damper'];
add_block(faultableDamperLib, faultableDamperPath, ...
    'Position', damperPosition, ...
    'NamePlacement', 'alternate', ...
    'D', dampingCoefficient)

```



Flexible Shaft

1. [Plot shaft speed](#) (see code)
2. [Plot twist](#) in shaft compliance elements (see code)
3. [Explore simulation results](#) using [sscexplore](#)
4. [Learn more](#) about this example

- 4 Enable a time-based fault and specify a response that includes a change in the damping coefficient and the generation of a MATLAB warning. Use these values for the damper **Fault** parameters:

- **Enable faults** — Enabled
- **Faulted damping coefficient** — 10
- **Enable temporal fault trigger** — Enabled
- **Simulation time for fault event** — 0.06
- **Reporting when fault occurs** — Warning

Script for Configuring the Rotational Damper Block Using a Timed Fault

```

%% Define Faultable Damper Block Parameters
underDamp = '10';
overDamp = '150';

faultTime = '.06';

aMaxDef = '100';
aMaxHigh = '400';
aMaxLow = '50';

shockNMaxDef = '1';
shockNMaxHigh = '5';
shockNMaxLow = '2';

disabled = '0';
enabled = '1';

none = '1';
warning = '2';
error = '3';

%% Parameterize a Timed Fault
set_param(faultableDamperPath,...
    'enable_faults', enabled,...           % Enable faults
    'b_fault', underDamp,...             % Faulted damping coefficient
    'temporal_fault', enabled,...       % Enable temporal fault trigger
    'fault_time', faultTime,...         % Simulation time for fault event
    'shock_fault', disabled,...        % Disable behavioral fault trigger
    'acceleration_limit', aMaxDef,...   % Maximum permissible acceleration
    'shock_limit', shockNMaxDef, ...   % Maximum number of shocks
    'report_fault', warning)           % Reporting when fault occurs
    
```

- 5 Simulate the model and plot the results.

Script for Generating and Plotting Simulation Results

```

%% Simulate
sim(model)

%% Get Simulation Results
simlog02 = simlog_ShaftWithTorsionalFlexibility;
simlog_t02 = simlog02.Clutch.P.series.time;
simlog_pClutch02 = simlog02.Clutch.P.series.values('Pa');
simlog_wMotor02 = simlog02.Shaft_Motor.F.w.series.values('rad/s');
simlog_wLoad02 = simlog02.Shaft_Load.F.w.series.values('rad/s');

%% Plot Results
X2 = simlog_t02;
YMatrix2 = [simlog_wMotor02 simlog_wLoad02];
Y2 = simlog_pClutch02;

% Create axes
axes2 = axes('Parent',figure1,...
    'Position',[0.125 0.4 0.775 0.216]);
hold(axes2,'on');

% Activate the left side of the axes
yyaxis(axes2,'left');

% Create multiple lines using matrix input to plot
plot1 = plot(X2,YMatrix2,'LineWidth',2);
set(plot1(1),'DisplayName','Motor Shaft', 'Color', 'k');
set(plot1(2),'DisplayName','Load Shaft', 'Color', 'b');

% Create ylabel
ylabel('Speed (rad/s)');

% Comment/uncomment the following line to remove/
% preserve the Y-limits of the axes
ylim(axes2,[-100 250]);

% Set the remaining axes properties
set(axes2,'YColor',[0 0.447 0.741]);

% Activate the right side of the axes
yyaxis(axes2,'right');
% Create plot
plot(X2,Y2,'DisplayName','Clutch Pressure','LineWidth',2, 'Color', 'r');

% Create ylabel
ylabel('Pressure (Pa)');

% Comment/uncomment the following line to remove/
% preserve the Y-limits of the axes
ylim(axes2,[0 2000000]);

% Set the remaining axes properties
set(axes2,'YColor',[0.85 0.325 0.098]);

% Create title
title('Time-Faulted Speeds and Clutch Pressure');

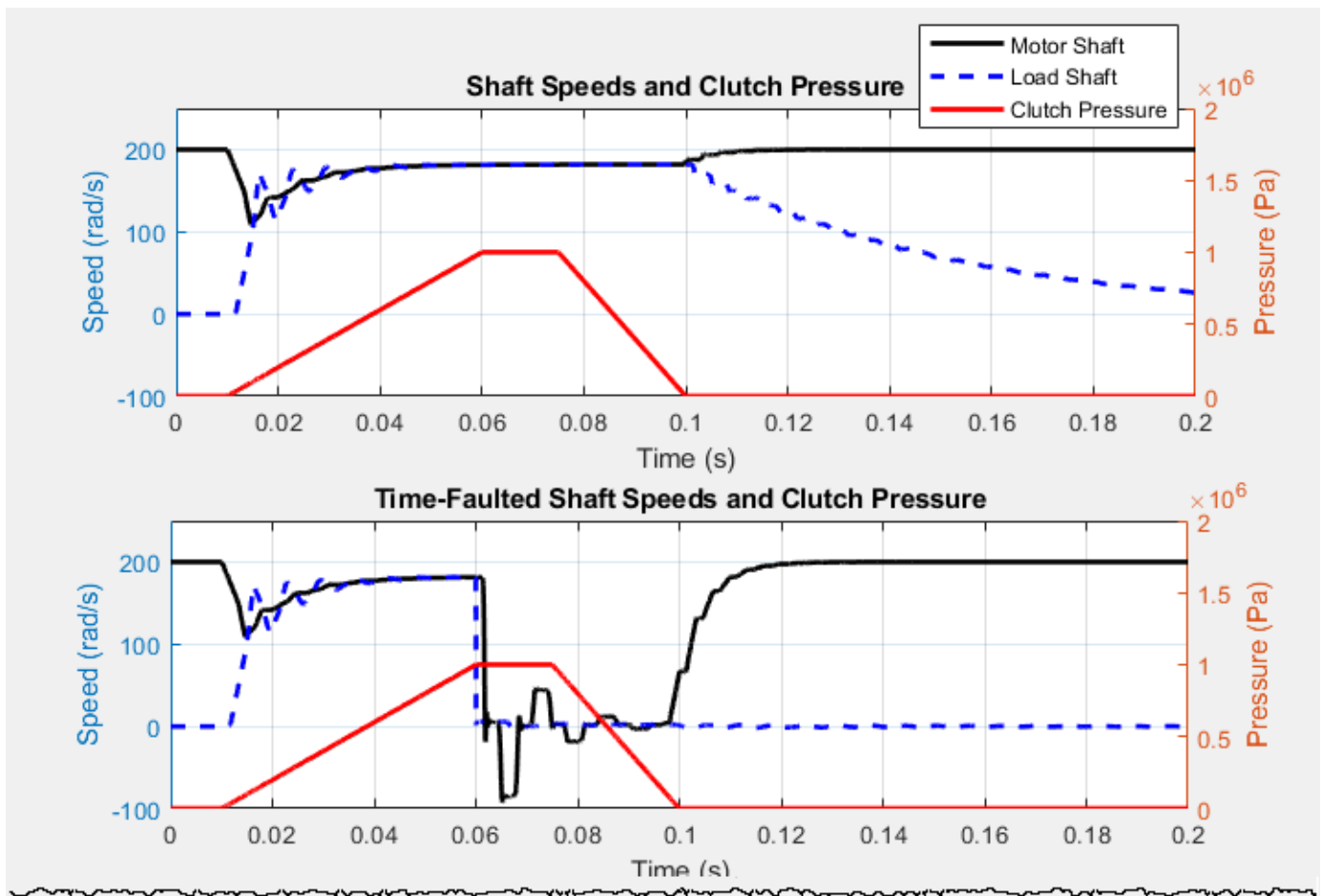
% Create xlabel
xlabel('Time (s)');

% Uncomment the following line to
% preserve the X-limits of the axes
% xlim(axes1,[0 0.2]);

% Set the remaining axes properties
box(axes2,'on');
grid(axes2,'on');
set(axes2,'LineStyleOrderIndex',2);

```

Warning: At time 0.060000, one or more assertions are triggered.
 A fault event has occurred The assertion comes from:
 Block path: ShaftWithTorsionalFlexibility/Faultable Damper
 Assert location: (location information is protected)



At simulation time $t = 0.06$ s, the time specified for the fault, a warning is reported. The damping coefficient drops and slows the speed of both shafts.

- 6 Enable a shock-based fault and specify a response that includes a change in the damping coefficient and the generation of a MATLAB warning. Then, simulate the model and plot the new results. Use these values for the damper **Fault** parameters:

- **Enable faults** — Yes
- **Faulted damping coefficient** — 150
- **Enable temporal fault trigger** — Disabled
- **Enable behavioral fault trigger** — Enabled
- **Maximum permissible acceleration** — 50
- **Maximum number of shocks** — 2
- **Reporting when fault occurs** — Warning

Script for Configuring the Rotational Damper Block Using a Timed Fault

```

%% Parameterize a Shock Fault
set_param(faultableDamperPath,...
    'enable_faults', enabled,...           % Enable faults
    'b_fault', overDamp,...               % Faulted damping coefficient
    'temporal_fault', disabled,...       % Disable temporal fault trigger
    'fault_time', faultTime,...         % Simulation time for fault event

```

```

    'shock_fault', enabled,...           % Enable behavioral fault trigger
    'acceleration_limit', aMaxHigh,...   % Maximum permissible acceleration
    'shock_limit', shockNMaxLow ,...    % Maximum number of shocks
    'report_fault', warning)           % Reporting when fault occurs

```

7 Simulate the model and plot the results.

Script for Generating and Plotting Simulation Results

```

%% Simulate
sim(model)

%% Get Simulation Results
simlog03 = simlog_ShaftWithTorsionalFlexibility;
simlog_t03 = simlog03.Clutch.P.series.time;
simlog_pClutch03 = simlog03.Clutch.P.series.values('Pa');
simlog_wMotor03 = simlog03.Shaft_Motor.F.w.series.values('rad/s');
simlog_wLoad03 = simlog03.Shaft_Load.F.w.series.values('rad/s');

%% Plot Results
X3 = simlog_t03;
YMatrix3 = [simlog_wMotor03 simlog_wLoad03];
Y3 = simlog_pClutch03;

% Create axes
axes3 = axes('Parent',figure1,...
    'Position',[0.125 0.08 0.775 0.216]);
hold(axes3,'on');

% Activate the left side of the axes
yyaxis(axes3,'left');

% Create multiple lines using matrix input to plot
plot1 = plot(X3,YMatrix3,'LineWidth',2);
set(plot1(1),'DisplayName','Motor Shaft','Color','k');
set(plot1(2),'DisplayName','Load Shaft','Color','b');

% Create ylabel
ylabel('Speed (rad/s)');

% Comment/uncomment the following line to
% remove/preserve the Y-limits of the axes
ylim(axes3,[-100 250]);

% Set the remaining axes properties
set(axes3,'YColor',[0 0.447 0.741]);

% Activate the right side of the axes
yyaxis(axes3,'right');

% Create plot
plot(X3,Y3,'DisplayName','Clutch Pressure','LineWidth',2, 'Color','r');

% Create ylabel
ylabel('Pressure (Pa)');

% Comment/uncomment the following line to
% remove/preserve the Y-limits of the axes
ylim(axes3,[0 2000000]);

% Set the remaining axes properties
set(axes3,'YColor',[0.85 0.325 0.098]);

% Create title
title('Shock-Faulted Shaft Speeds and Clutch Pressure');

% Create xlabel
xlabel('Time (s)');

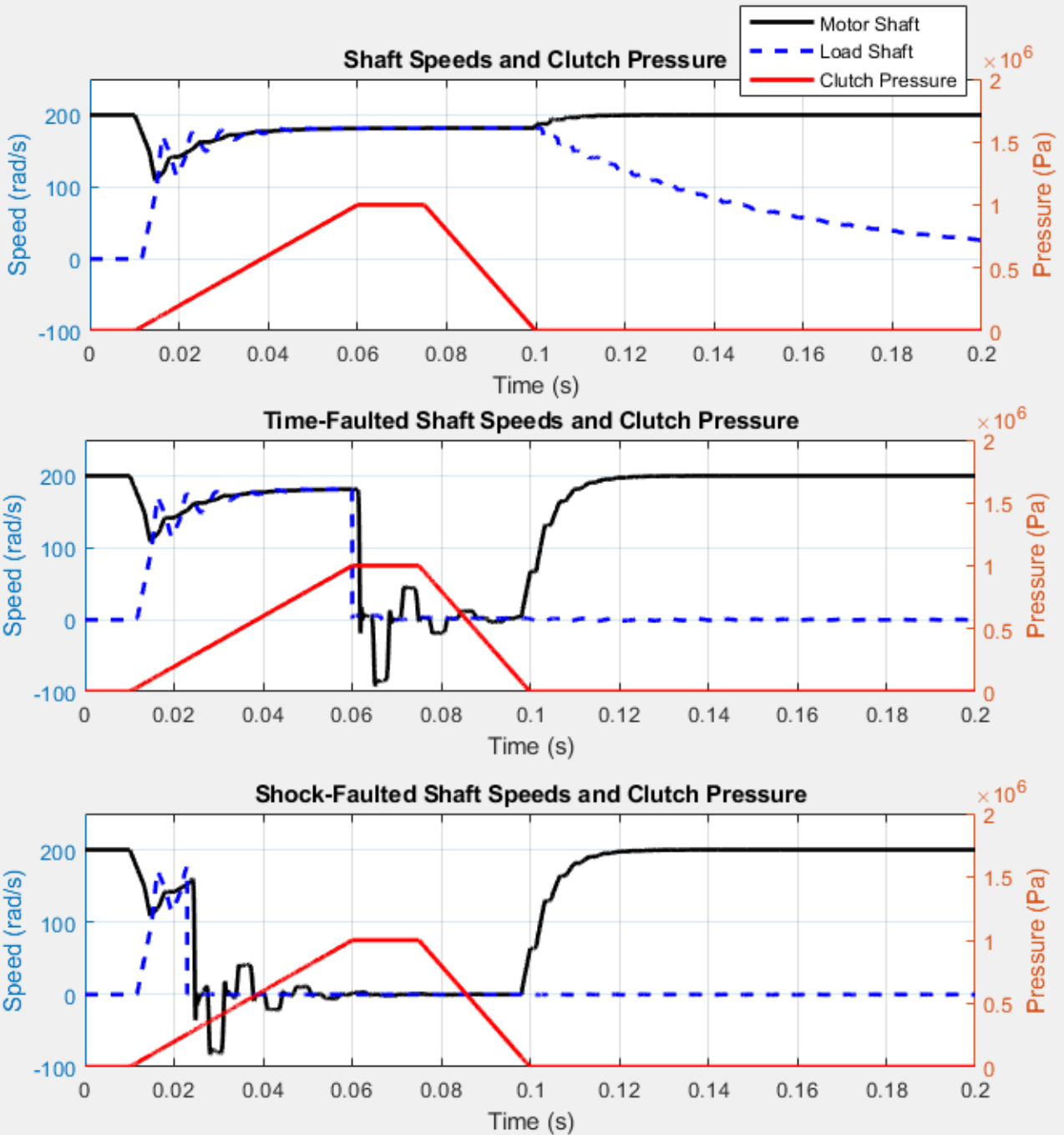
% Uncomment the following line to
% preserve the X-limits of the axes
% xlim(axes1,[0 0.2]);

% Set the remaining axes properties
box(axes3,'on');
grid(axes3,'on');
set(axes3,'LineStyleOrderIndex',2);

```

Warning: At time 0.026048, one or more assertions are triggered.
A fault event has occurred The assertion comes from:

Block path: ShaftWithTorsionalFlexibility/Faultable Damper
 Assert location: (location information is protected)



At simulation time $t = 0.026$ s, the maximum number of shocks for the specified acceleration is reached. A warning is reported and the damping coefficient increases and slows the speed of both shafts.

See Also

Rotational Damper | Translational Damper

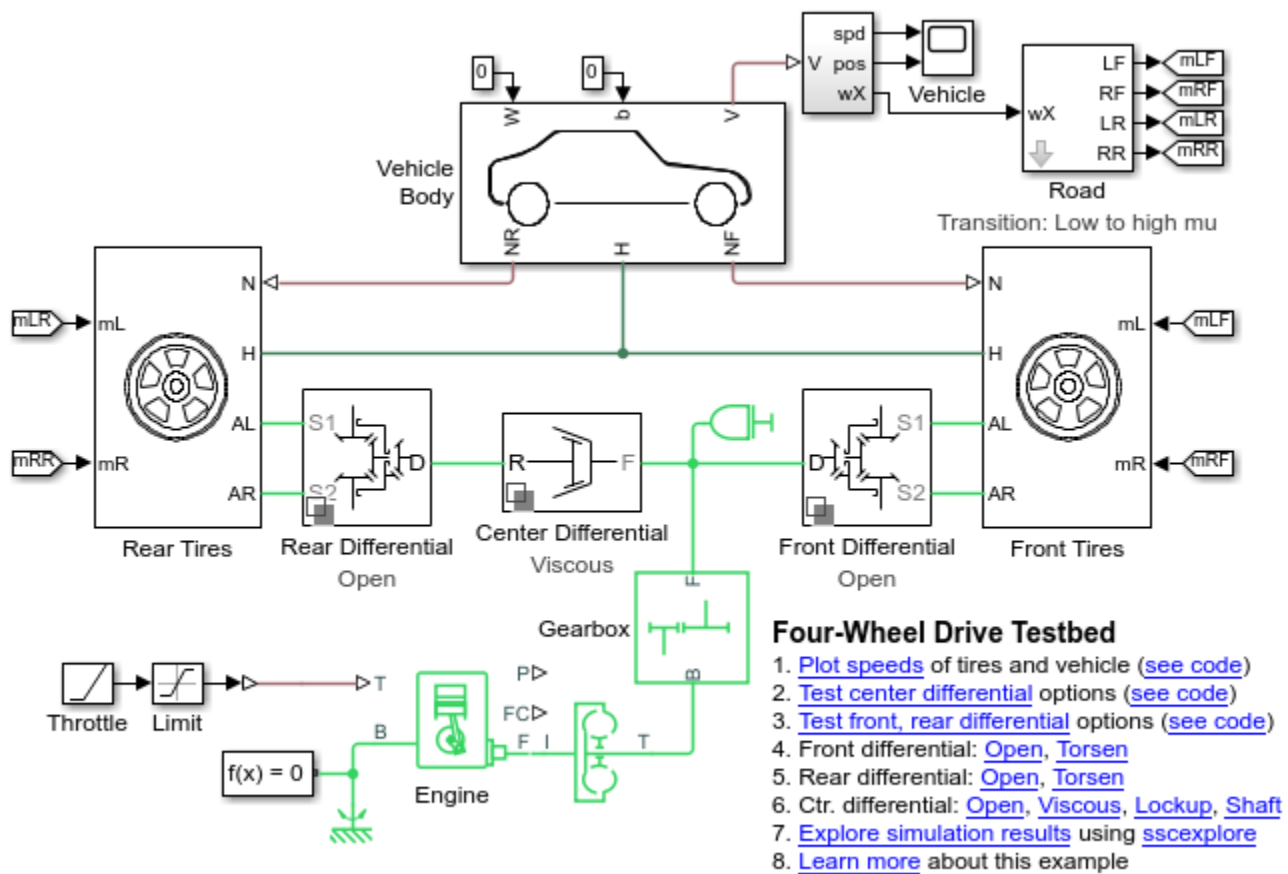
Modeling Driveline Environments

Model a Road Profile with Varying Elevation and Friction

This example shows how to vary road conditions throughout a simulation of a 4-wheel drive vehicle test-bed. The model is a version of the “Four-Wheel Drive Testbed” on page 18-53 that is updated to include Road Profile blocks for both the front and rear tires. As the vehicle travels, the axle parameters and the position of the center of gravity (CG) determine the position of the front and rear axles. The Road Profile blocks use the axle positions to determine vehicle angle and tire friction coefficients based on parameters that you specify.

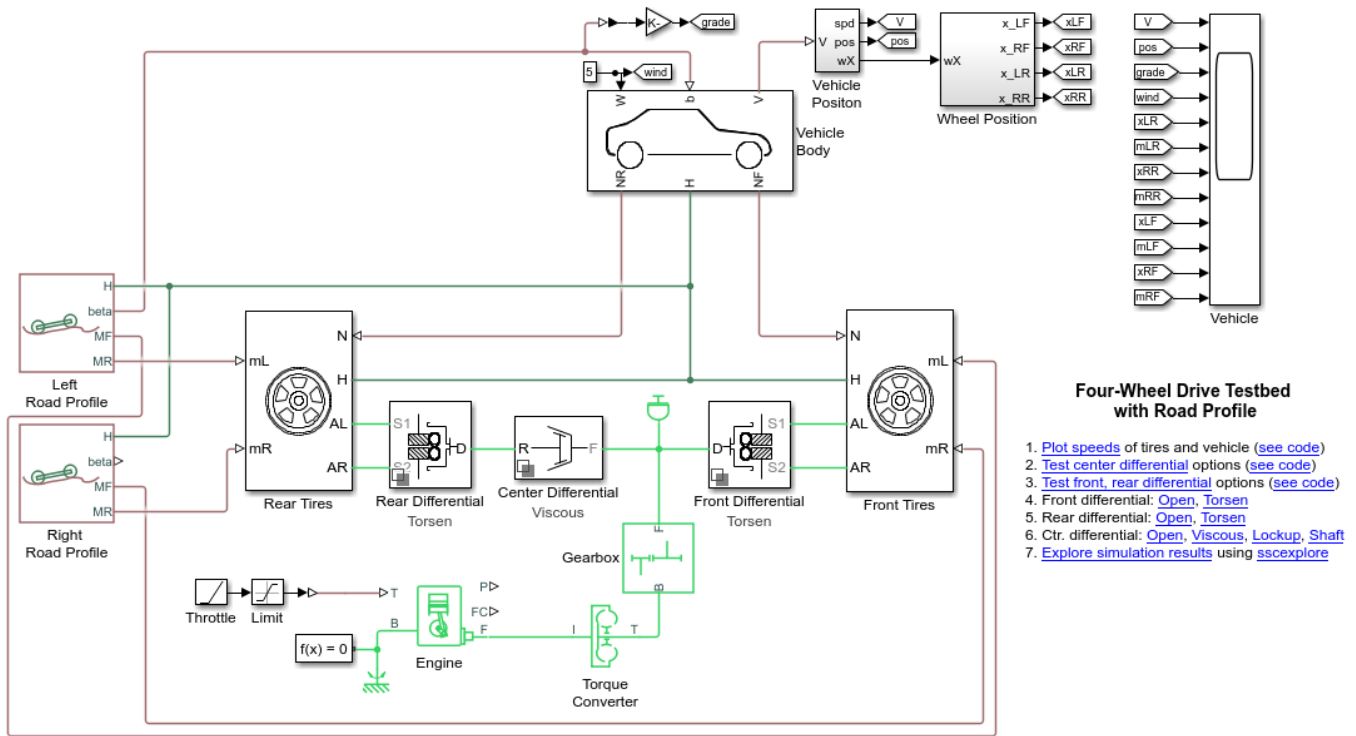
Updates to the Original Model

The original model determines the magic formula coefficients based on the position of the vehicle relative to its position at the simulation start. The figures show the original and the updated models.

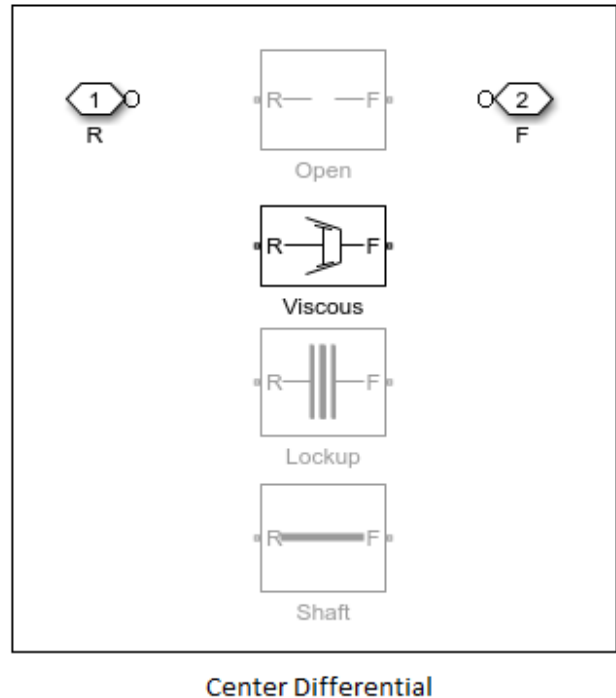
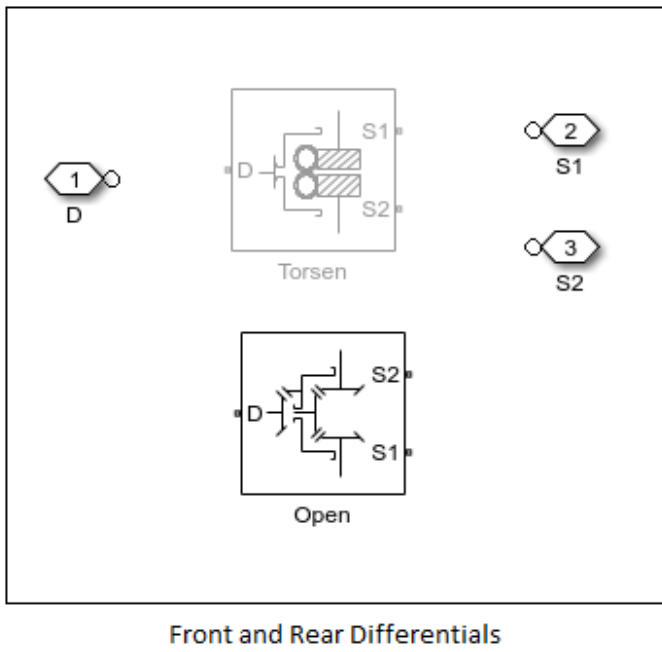


The new model includes Road Profile blocks for both the right- and left-side tires. To open the model, at the MATLAB command prompt, enter

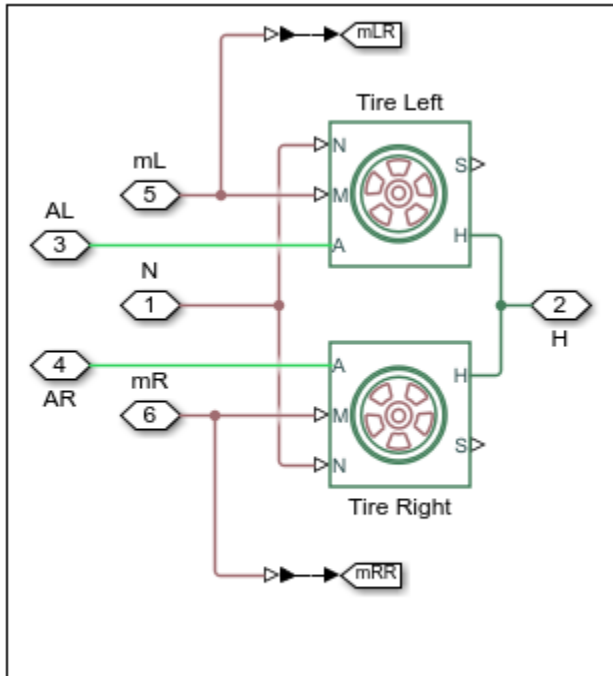
```
open_system('sdl_vehicle_road_4wd_testbed')
```



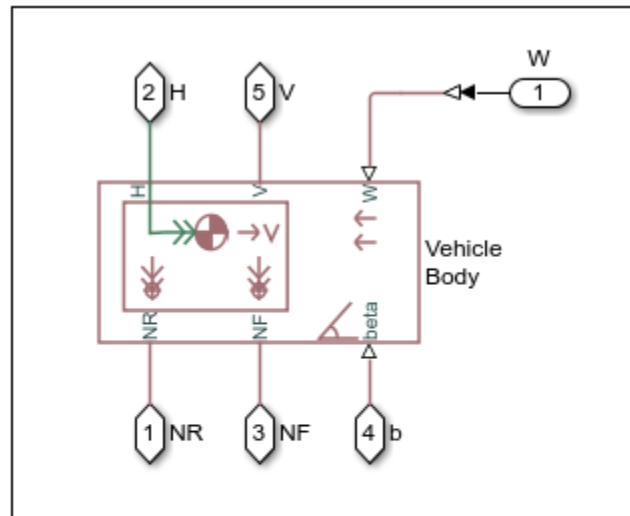
In both models, the front, rear, and center differentials are represented by variant subsystems.



The front and rear tire subsystems contain Tire (Magic Formula) blocks, while the **Vehicle Body** subsystem is a mask for a Vehicle Body block.



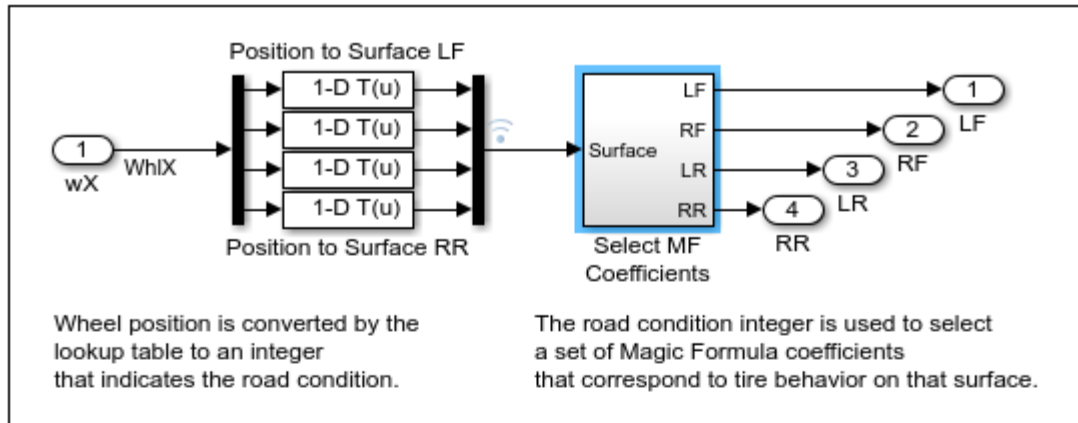
Front and Rear Tires



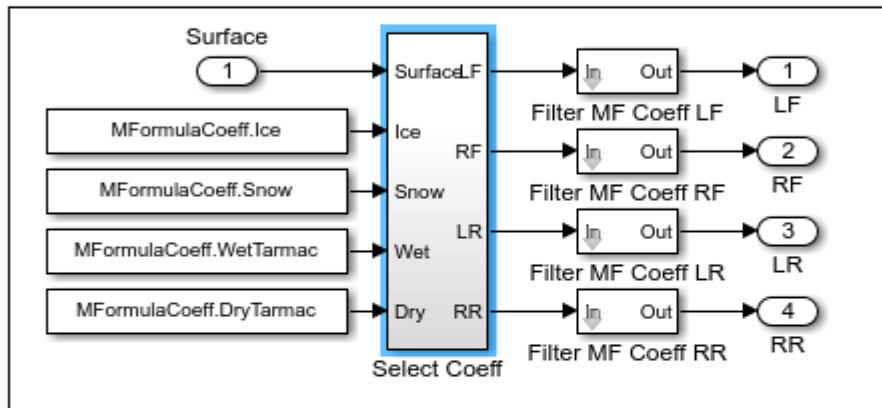
Vehicle Body

Updates that allow the model to determine road conditions using the Road Profile blocks are:

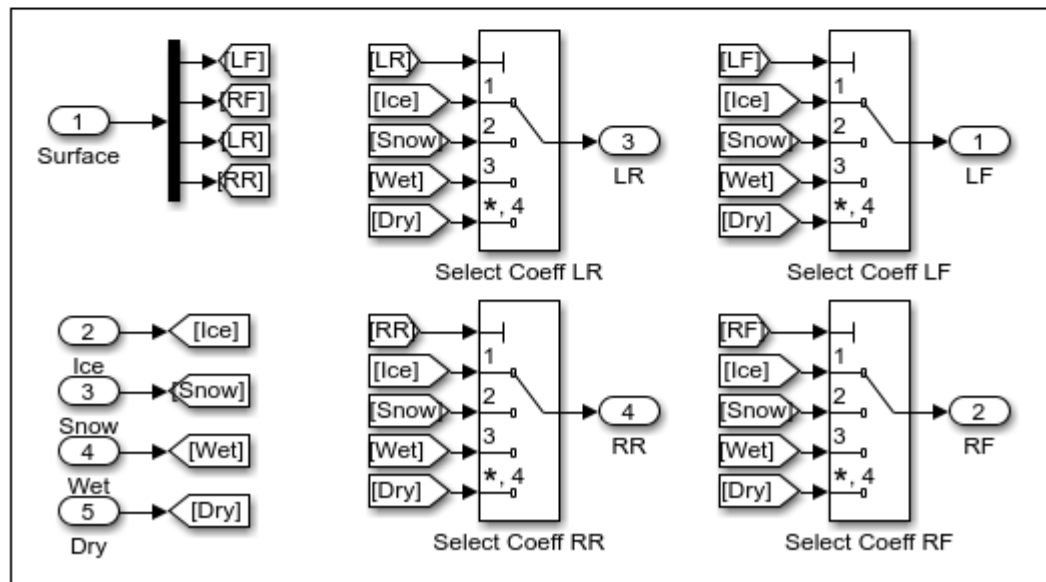
- 1 Replacement of the **Road** subsystem with the **Wheel Position** subsystem. The road subsystem contains three levels of subsystems that the model uses to determine the Magic Formula coefficients for the tires during simulation.



Road Subsystem

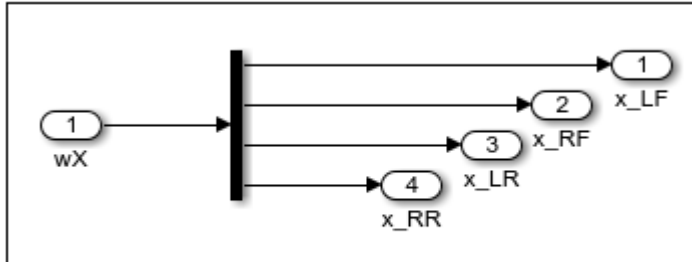


Select MF Coefficients Subsystem



Select Coeff Subsystem

The addition of the Road Profile blocks allows for the replacement of the **Wheel Position** system with the much simpler **Wheel Position** subsystem. The new subsystem demuxes the wheel position signals.



Wheel Position

- Parameterization for the added Road Profile blocks for the right and left tires:

Main

- **Horizontal distance from CG to front axle** — x_f
- **Horizontal distance from CG to rear axle** — x_r
- **Horizontal distance for vertical profile** — x_{height_vector}
- **Vertical profile** — $height_vector$

Friction

- **Friction output** — Physical signal Magic Formula coefficients
- **Horizontal distance for friction profile** — $x_{friction_vector}$
- **Magic Formula coefficients for front axle** — MF_M_matrix
- **Magic Formula coefficients for rear axle** — MR_M_matrix

Position Variable

- **Override** — select
- **Beginning Value** — x_0

- Vehicle Body block **Main** parameter updates:

- **Horizontal distance from CG to front axle** — x_f
- **Horizontal distance from CG to rear axle** — x_r

- Variable definitions for the model:

```
x_f=1.4;
x_r=1.6;
x_height_vector=[-10, 0, 10];
height_vector=[0, 0, 0.25];
x_friction_vector = [ -10, 5, 10, 15 ];
MF_M_matrix = [10  1.9  1  0.97;...
               4  2  0.1  1;...
               12  2.3  0.82  1;...
               10  1.9  1  0.97];
MR_M_matrix = [10  1.9  1  0.97;...
```

```

12 2.3 0.82 1;...
12 2.3 0.82 1;...
10 1.9 1 0.97];

```

$x_{\theta} = 0;$

5 Additional environmental updates:

- The left-tire Road Profile block introduces a variable road grade. The Gain block converts the grade variable, β from radians to degrees.
- A headwind is included by using a nonzero value for the Constant block.

6 Signal block updates:

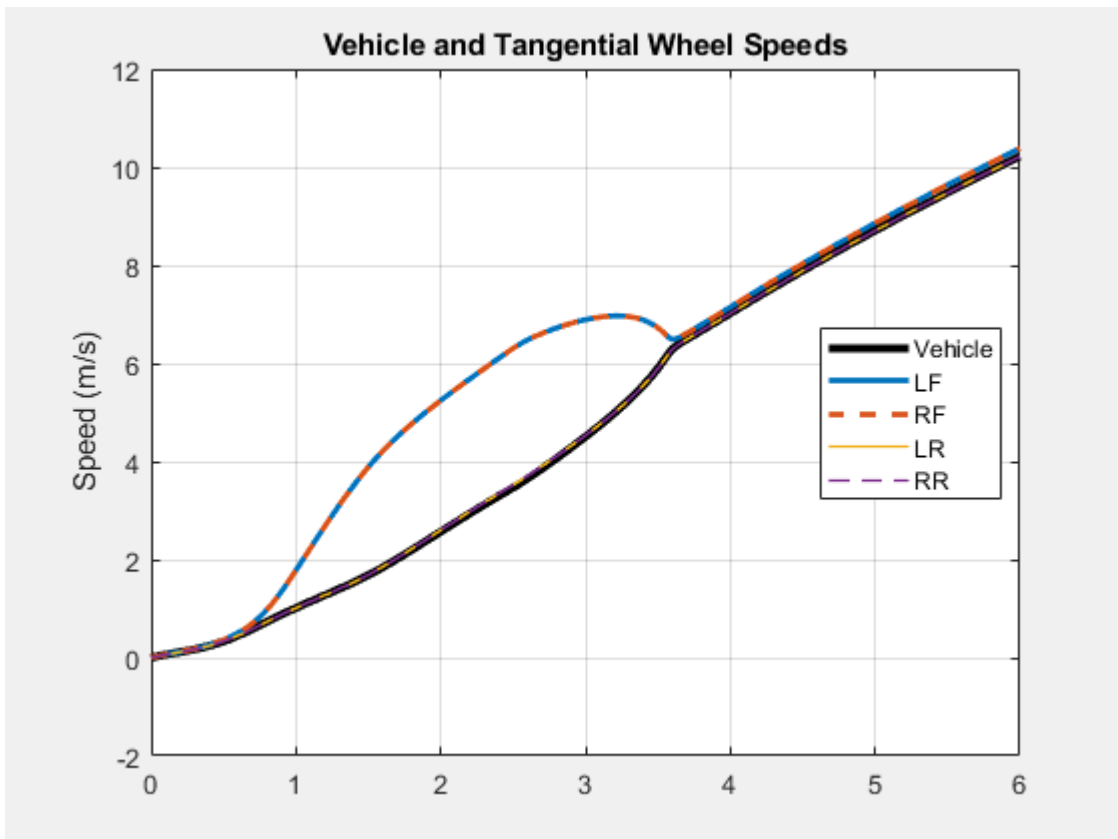
- Outports and Inports blocks are replaced with Connection Port blocks.
- Goto and From blocks are used to relay signals to the Scope.

7 Data visualization and logging updates:

- The Scope block is updated to show tire positions, Magic Formula Coefficients, headwind, and road elevation.
- The simlog name is updated to match the name of the updated model.
- The differential test and plot generation code is updated to use the new simlog name.

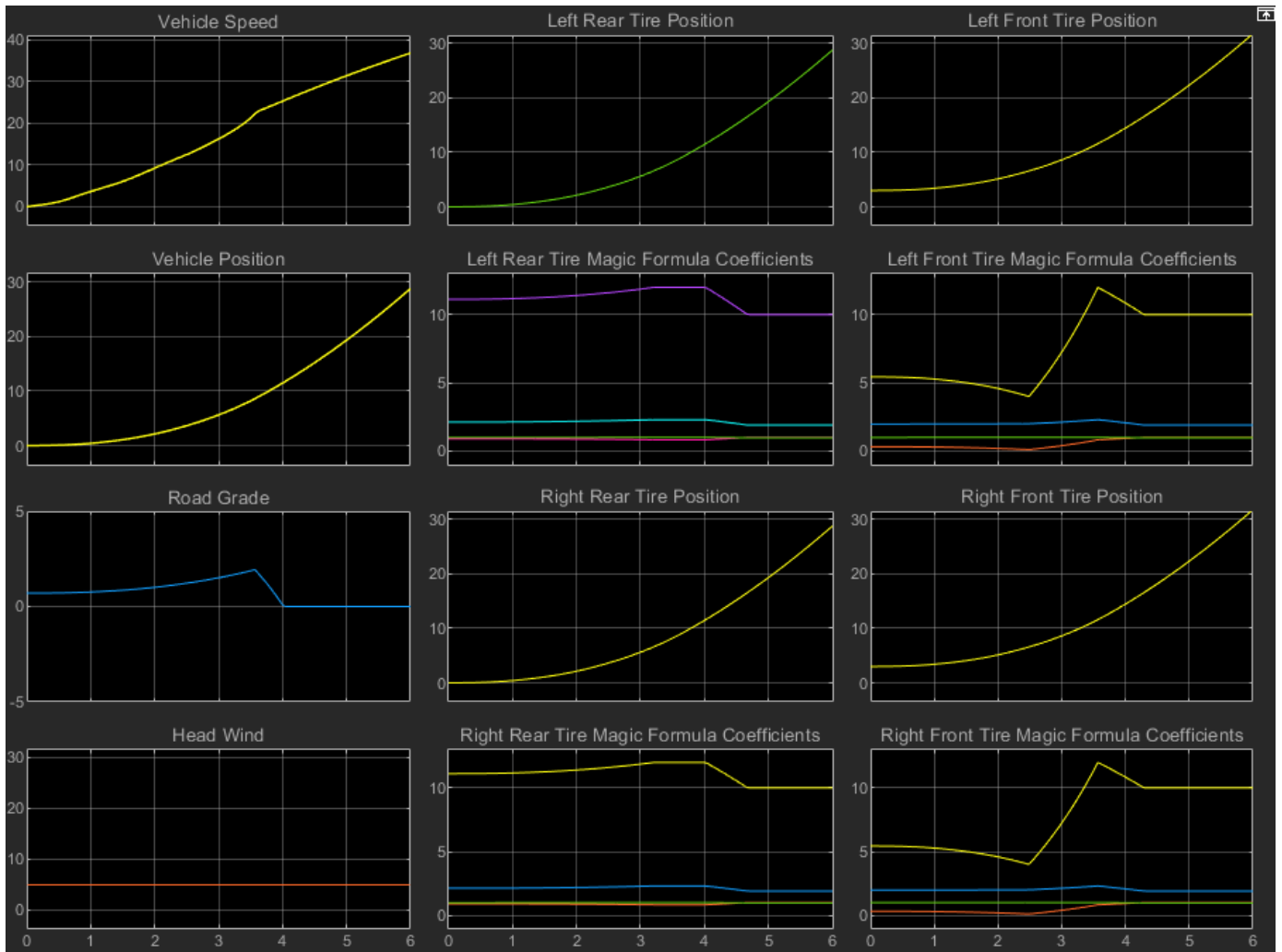
Run the Simulation

- 1 To run the simulation and generate a plot of the results, click **Plot speeds**.



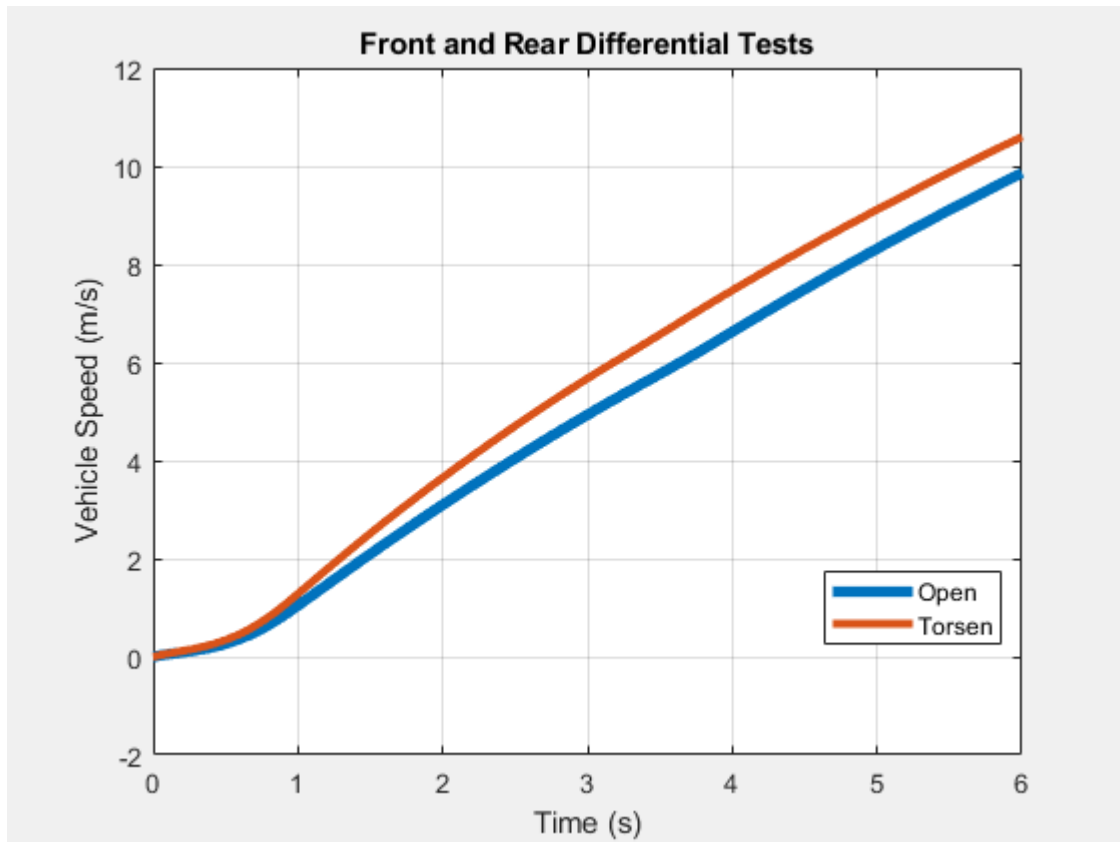
The front tires experience slip in the middle of the simulation due to the slippery conditions related to the $[4 \ 2 \ 0.1 \ 1]$ Magic Coefficients that the simulation uses when the positions of the front tires are 5 to 10 meters from the original positions.

- To see how the road incline, headwind, tire position, and vehicle speed and position relate to each other and to the Magic Coefficients, open the Scope block.



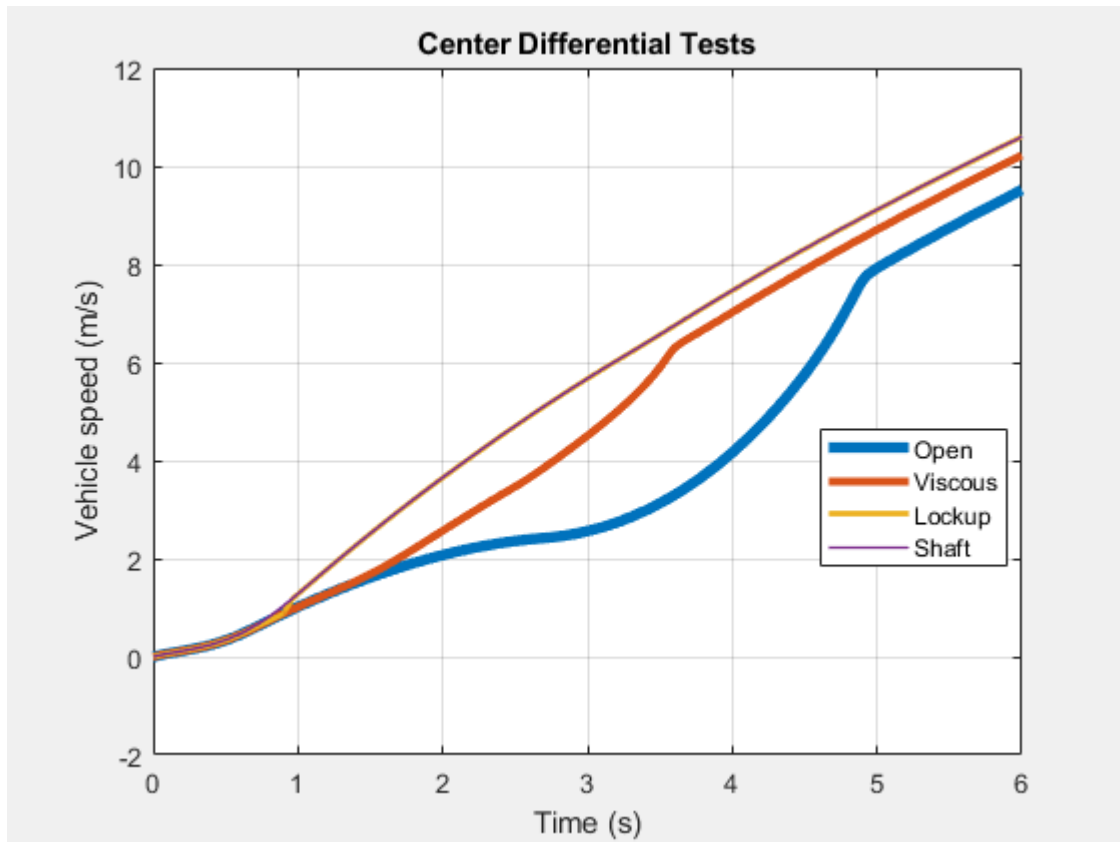
The vehicle velocity increases slightly as the road grade decreases.

- To test the front and rear differentials in both the Open and Torsen variants, click **Test front, rear differential**.



The Torsen differential configuration results in higher velocity throughout the simulation.

- 4 To test all variants of the center differential, click **Test center differential**.



The open and viscous differential configurations result in lower, more variable velocity when the grade changes during the simulation.

See Also

Road Profile | Tire (Friction Parameterized) | Tire-Road Interaction (Magic Formula) | Tire (Magic Formula) | Vehicle Body

Analyzing Driveline Models and Simulations

These sections explore some of the more complex issues in driveline modeling, and some powerful techniques that can extend your driveline simulations.

- “Driveline Simulation Performance” on page 16-2
- “Resolve Partitioning Solver Simulation Issues for Simscape Driveline Models” on page 16-6
- “Driveline Degrees of Freedom” on page 16-21
- “Driveline States & Effect of Clutches” on page 16-32
- “How Simscape Driveline Simulates a Drivetrain System” on page 16-34
- “Model Thermal Losses in Driveline Components” on page 16-35
- “Simscape Driveline Limitations” on page 16-42

For information about the nature of Simscape models, states, and simulation, see the documentation for “Simulink” and “Simscape”.

Driveline Simulation Performance

In this section...

“About Simulation Performance” on page 16-2

“Adjust Model Fidelity” on page 16-2

“Improve Simulation Performance by Using the Partitioning Solver” on page 16-3

“Optimize Simulation of Stiff Drivelines” on page 16-3

“Optimize Simulation of Clutches” on page 16-3

About Simulation Performance

Driveline simulation involves the tradeoff between accuracy and speed inherent in all numerical simulation. Accuracy bundles two distinct issues, the accuracy or *fidelity* of the model, versus of the accuracy of the simulation methods. This section describes the inherent complexity of driveline models, as distinct from general simulation issues.

About solvers and simulation methods, see “Setting Up Solvers for Physical Models” and “Making Optimal Solver Choices for Physical Simulation”.

Adjust Model Fidelity

Improving the fidelity of driveline models involves making blocks that are more accurate representations of the actual physical components. For example, you can make the internal dynamics of the components represented by blocks more or less accurate and realistic by:

- Turning physical effects on and off, such as nonideal gear meshing losses (gear efficiency)
- Including or omitting compliance (including damped spring reactions), hard stops, and time lags
- Including or omitting Coulomb friction from clutches and clutch-like elements
- Steepening or softening sharp gradients in physical thresholds, such as velocity thresholds in clutches and nonideal gears

Modeling these physical effects requires additional dynamics and algebraic constraints, generates computationally more intensive simulations, and can reduce simulation speed, often considerably.

Model Fidelity in Ordinary Desktop Simulation

- Very small velocity threshold values and short time lags can degrade numerical convergence or simulation performance. Consider whether you can make these values larger in your simulation.
- If your model includes gears with efficiency loss, select adaptive zero-crossing in the **Model Configuration Parameters** menu.

Model Fidelity in Fixed-Step, Real-Time, and Hardware-in-the-Loop Simulation

Apart from clutches, MathWorks® does not recommend including fidelity enhancements in fixed-step/ fixed cost, real-time, or hardware-in-the-loop (HIL) simulation.

To model compliance or efficiencies, consider reducing the number of such elements by:

- Deleting unnecessary lossy elements

- Combining lossy elements into as few elements as possible

If you simulate with a fixed-step solver, avoid:

- Very small velocity thresholds.
- Time lags that are short compared to the fixed time step.

Improve Simulation Performance by Using the Partitioning Solver

The Partitioning solver is a Simscape fixed-step local solver that improves performance for certain models. For more information about the Partitioning solver, including limitations for the types of models that it can solve, see “Increase Simulation Speed Using the Partitioning Solver”. For an example that shows how to simulate a Simscape Driveline model using the Partitioning solver, see “Resolve Partitioning Solver Simulation Issues for Simscape Driveline Models” on page 16-6.

Optimize Simulation of Stiff Drivelines

When modeling a driveline, consider not modeling all the compliances, depending on the purpose of your model. If there are specific compliances that are more dominant than others, then try modeling only the dominant compliances.

The coupling of drivelines to external loads — for an automobile, the wheel-tire-road load — is often stiff. Driving and road conditions typically change over seconds or tens of seconds. However, the internal changes of the drive system of an automobile can change over fractions of a second, especially if clutch changes and braking are at work. In addition, clutch locking and unlocking events create dynamic discontinuities.

For example, a tire is “stiff” in responding slowly to imposed forces and experiencing slip. A tire also has a broad range of frequency responses. Consider modeling tire compliance only when you model the automobile accelerating from rest.

Optimize Simulation of Clutches

Clutch locking and unlocking events generate discontinuous changes in driveline dynamics and can cause major inaccuracies, particularly if you are simulating with a large variable-step solver tolerance or a large fixed time step.

- Clutch discontinuities change the number and nature of the degrees of freedom of the driveline during the simulation.
- Because clutch discontinuities are idealized events, they cause the driveline torques to change abruptly, as the clutch switches abruptly between static and kinetic friction.

Smoothing and Offsetting Clutch Control Signals

You exert dynamic control on the locking and unlocking of clutches through their input pressure or other locking signals.

The simplest way to force a locking is to change a clutch pressure abruptly from zero to some predetermined value. You can then force an unlocking by abruptly changing the clutch pressure back to zero. Such abrupt clutch pressure changes are not realistic. The best solution is to model full clutch actuation. However, you can use simplified models to reduce model complexity.

You can improve your clutch modeling and make it more realistic by ensuring that the clutch pressure signals rise and fall smoothly, not suddenly. The Simulink Sources library provides many ways to create such signals. You can also reshape existing signals using blocks such as State-Space and Transfer Fcn.

These example models illustrate smoothed clutch pressure signals:

- “Custom Clutch” on page 18-24 ramps up and down the input clutch pressure.
- “Vehicle with Four-Speed Transmission” on page 18-161 uses Transfer Fcn blocks to reshape and smooth sharp clutch pressure signals.

For more information about smoothing clutch signals, see “Model Realistic Clutch Pressure Signals” on page 12-7.

Adjusting Clutch Parameters

You can adjust internal parameters within each clutch block to control when and how the clutch locks and unlocks.

Changing Pressure or Force Threshold

The locking signal coming into a clutch is physical, with units of force or pressure. With some clutches, you can specify a force or pressure threshold F_{th} or P_{th} . This threshold imposes a cutoff on the clutch pressure such that the effective controlling pressure is $P - P_{th}$ rather than P . If $P < P_{th}$, no pressure at all is applied. (Normal force between clutch surfaces can substitute for pressure.) Raising the pressure or force threshold of a clutch that has an adjustable threshold makes it harder for the clutch to engage.

Tip If a clutch in your simulation engages too easily, consider raising its pressure or force threshold. If the clutch has difficulty engaging, consider lowering this threshold.

Changing Velocity Tolerance

Most clutch blocks have a velocity tolerance parameter ω_{Tol} that controls when the clutch locks or unlocks.

- A clutch can lock only if the relative shaft velocity ω lies in the range $-\omega_{Tol} < \omega < +\omega_{Tol}$.
- A clutch unlocks if the torque across the clutch exceeds the static friction limit, which depends in turn on the normal force across the clutch.

You specify ω_{Tol} values through each clutch block.

Tip If a clutch switches between locked and unlocked too easily during simulation, consider increasing its velocity tolerance.

Adjusting Solvers for Clutch Discontinuities

If you use a solver tolerance or step size that is too large, clutch discontinuities can cause major inaccuracies.

- If the variable-step tolerances are too large, the solver finds it difficult or impossible to track the dynamic change associated with the change of friction torques acting on the driveline accurately.

- If the fixed step size is too large, the solver cannot accurately resolve abrupt changes such as clutch locking and unlocking events. A fixed-step solver cannot adaptively reduce its step size to compensate.

Tip If you encounter convergence failures or abrupt driveline state (velocity) changes at or around the instant of clutch state changes, consider reducing the solver tolerances (for a variable-step solver) or the step size (for a fixed-step solver). Set the variable-step solver tolerance or the fixed-step solver step size to the smallest value possible that produces an acceptable simulation speed (not too slow).

Adjustment	Solver Type and Setting	Effect on Accuracy	Effect on Speed	Effect on Clutch Simulation
Reduce	Variable-step: tolerances	Increases	Reduces	Improves resolution and simulation of abrupt locking and unlocking
	Fixed-step: step size			
Increase	Variable-step: tolerances	Reduces	Increases	Degrades resolution and simulation of abrupt locking and unlocking
	Fixed-step: step size			

Resolve Partitioning Solver Simulation Issues for Simscape Driveline Models

In this section...

“Resolving Issues for Blocks with Stiffness or Friction” on page 16-6

“Using the Partitioning Solver” on page 16-6

“Diagnose Unexpected Simulation Behavior Due to Nonlinearities” on page 16-7

“Resolve Chatter Due to Stiffness” on page 16-13

The Partitioning solver is a Simscape fixed-step, local solver that improves performance for certain models. However, when using the Partitioning solver, some Simscape Driveline models generate warnings, stop and generate errors, fail to initialize, or yield signal chatter due to numerical difficulties. These examples show how to eliminate errors, mitigate warnings, and reduce chatter by resolving numerical difficulties.

Resolving Issues for Blocks with Stiffness or Friction

Numerical difficulties that prevent models from simulating to completion, yield warnings, or introduce chatter are typically related to blocks that have high stiffness or friction. Simscape Driveline blocks with high stiffness or friction include clutches, belt pulleys, tires, and flexible shafts.

To resolve numerical difficulties in models that contain these blocks, use one or more of the methods:

- Adjust the solver settings.
- Remove high-priority variable redundancies.
- Unlock clutch initial conditions.
- Loosen friction-related tolerances.
- Eliminate high-stiffness blocks.
- Eliminate degrees of freedom.

Using the Partitioning Solver

When simulating models using the Partitioning solver:

- 1 Simulate the model by using a global variable-step solver to capture baseline results that agree with your mathematical model or empirical data.
- 2 Configure the local solver for a Partitioning solver simulation.
- 3 Run the Partitioning solver simulation. If the simulation:
 - Runs to completion — Compare the Partitioning solver simulation results to the baseline results. If the results do not agree, adjust the solver settings or model components and simulate again. For example, decrease the step size or simplify model dynamics. For more information, see “Reduce Numerical Stiffness”, “Choose Step Size and Number of Iterations”, and “Reduce Fast Dynamics”.

Repeat until the simulation returns results that agree with the baseline results.

- Fails to simulate to completion due to numerical issues — Resolve the issues by applying one or more of the methods listed in “Resolving Issues for Blocks with Stiffness or Friction” on

page 16-6. Rerun the simulation. Repeat until the simulation runs to completion, then compare the results to the baseline results. If the results do not agree, adjust the solver settings or model components and rerun the simulation. Repeat until the Partitioning solver simulation returns results that agree with the baseline results.

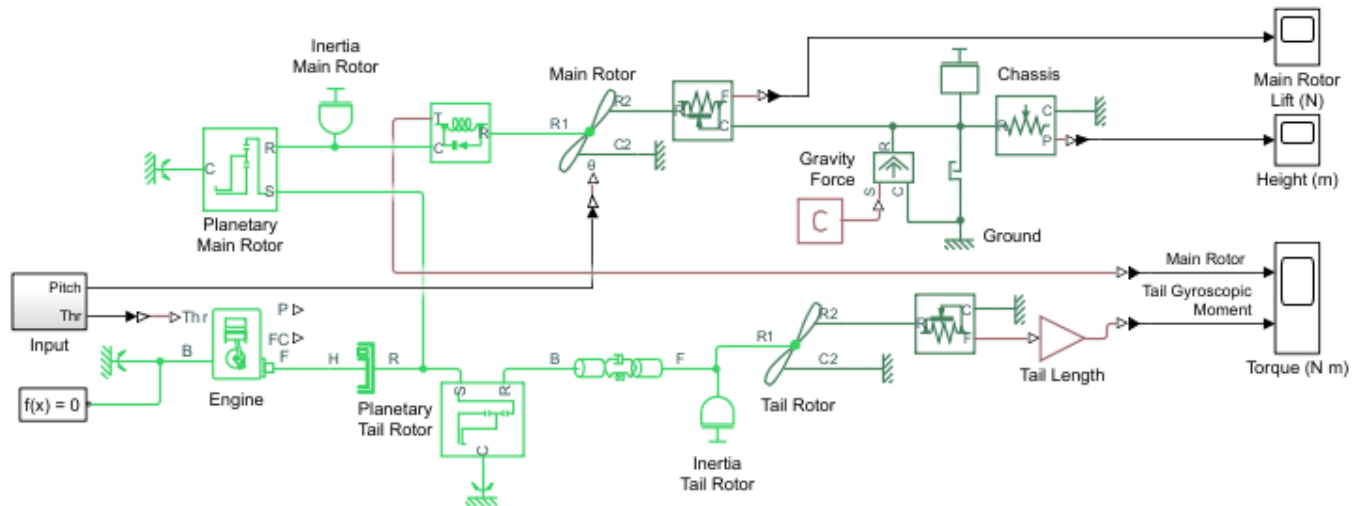
Diagnose Unexpected Simulation Behavior Due to Nonlinearities

This example uses a model of a helicopter taking off to show how to diagnose a numerical issue caused by a nonlinearity. The model uses a Translational Hard Stop block to represent the interaction of the helicopter chassis with the ground. When the helicopter takes off, the block implements smoothing to transition away from a position of zero. The partitioning solver produces unexpected results because of this smoothed transition.

- 1 Open the model. At the MATLAB command prompt, enter:

See Code

```
openExample('sdl/HelicopterTransmissionExample')
```



Note that the automatic solver selection is `ode23t`. This is a variable-step solver.

- 2 Inspect the Solver Configuration block settings

See Code

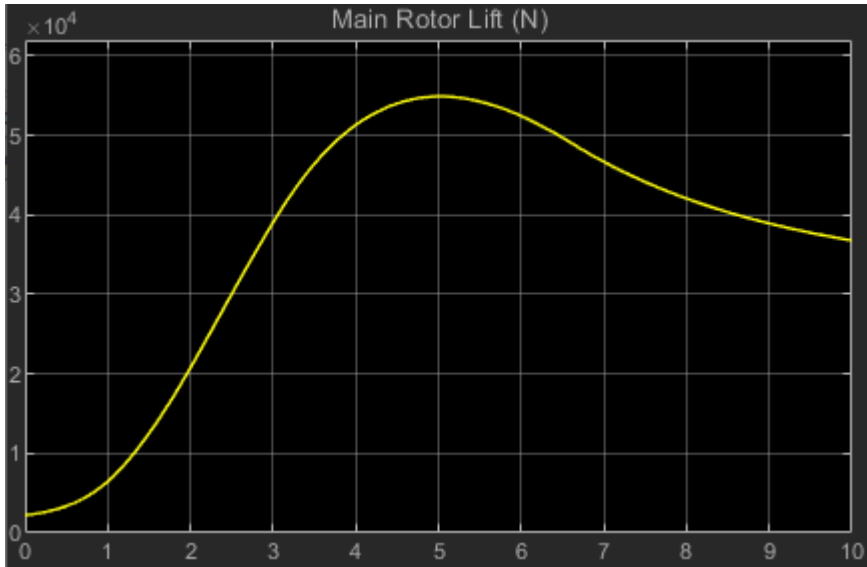
```
model = 'HelicopterTransmission';
solvConfig = 'Solver Configuration';
solvConfigPath = [model, '/', solvConfig];
open_system(solvConfigPath)
```

Note that **Use local solver** is off. This means that the simulation uses a global solver.

- 3 Generate baseline results from the variable-step, global solver and open the Scope block.

See Code

```
sim(model)
scope = 'Main Rotor Lift (N)';
scopePath = [model, '/', scope];
open_system(scopePath)
```



- 4 Configure the model to use the partitioning solver. In the Solver Configuration block settings, select **Use local solver**.

See Code

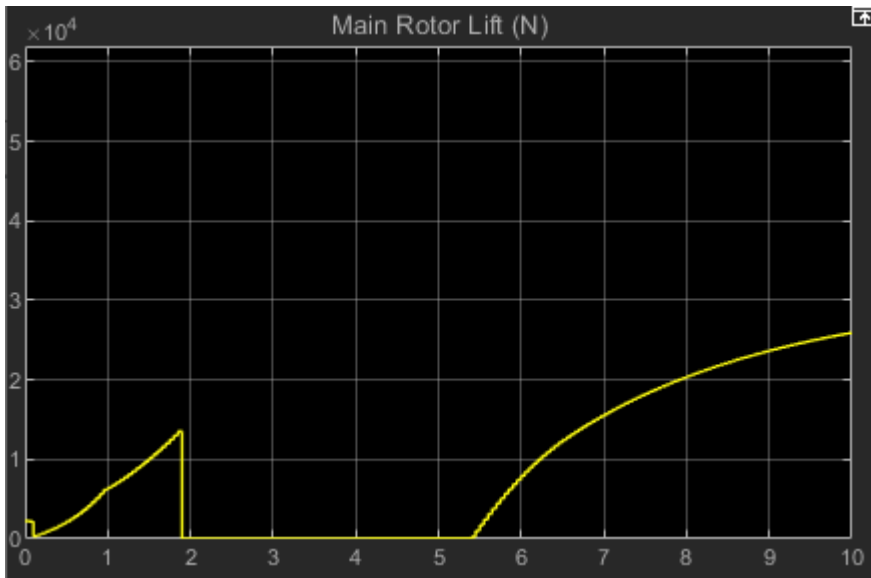
```
open_system(solvConfigPath)
set_param(solvConfigPath, ...
    'UseLocalSolver', 'on', ...
    'DoFixedCost', 'on')
```

When you select **Use local solver**, the block enables related parameters. The default parameter settings are:

- 1 **Solver type** — Partitioning
 - 2 **Sample time** — 0.05
 - 3 **Partition method** — Robust simulation
 - 4 **Partition storage method** — Exhaustive
 - 5 **Use fixed-cost runtime consistency iterations** — on
 - 6 **Nonlinear iterations** — 3
- 5 Simulate the model.

See Code

```
sim(model)
open_system(scopePath)
```

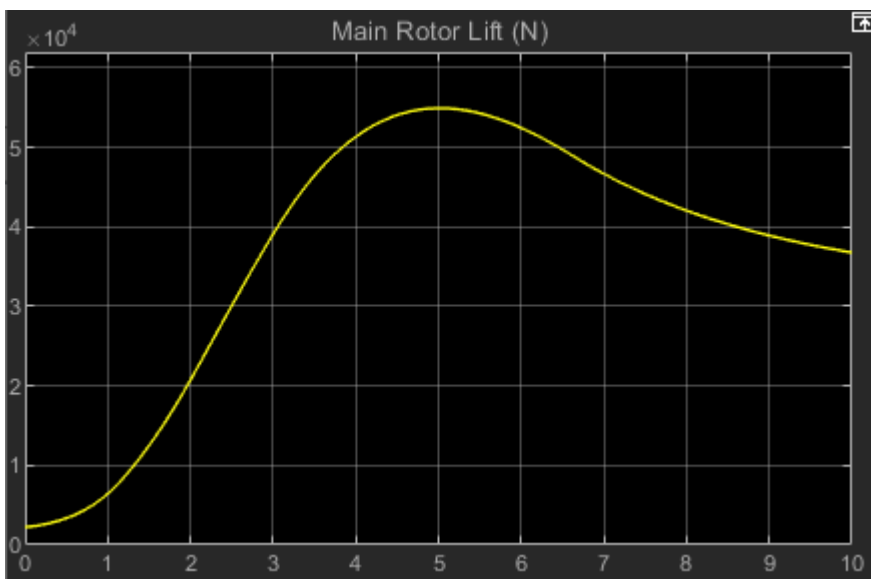


- The simulation runs to completion, but it generates unexpected results.
- To diagnose this behavior, reduce the simulation sample time. Set **Sample time** to 0.01 and run the simulation.

See Code

```
open_system(solvConfigPath)
set_param(solvConfigPath, 'LocalSolverSampleTime', '0.01')

sim(model)
open_system(scopePath)
```

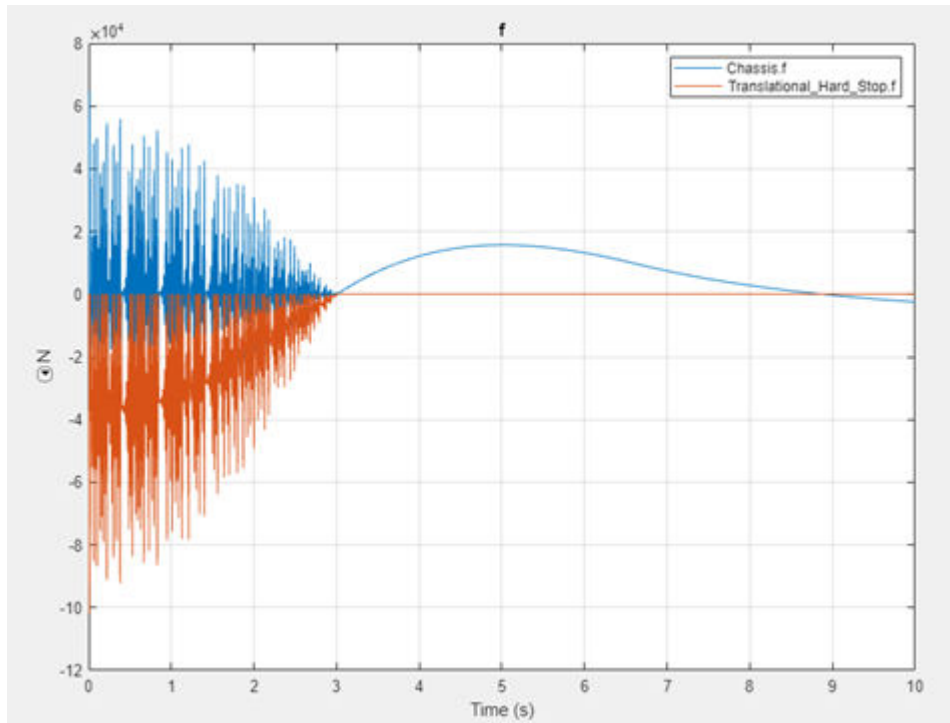


The partitioning solver results now reflect the baseline results. This confirms that the model is valid and is generated unexpected results for numerical reasons.

- 7 To diagnose the numerical issue, examine the results in the Simscape Results Explorer. Compare **Chassis.f** and **Translational_Hard_Stop.f** to observe the fast dynamics that affect the system.

See Code

```
sscexplore(simlog_sdl_transmission_helicopter_base);
```



The force variables both show excessive noise at the beginning of the simulation, when the helicopter takes off. You can use the **Statistics Viewer** tool to learn more about the nature of the problem. The figure shows **Translational_Hard_Stop.f** is in a partition by itself.

Q Type here to filter statistics	
Name	Value
▼ Partition 10	Backward ... ^
Equation Type	Switched I...
Number of variables	1
Number of equations	1
Number of modes	4
Number of configurations	16
Memory estimate	4
▼ Partition 11	Backward ... v
Sources ^	
Source	Value
Translational Hard Stop.f	Force

Translational_Hard_Stop.x is in a different partition because the force and position relationship is nonlinear in the current configuration of the Translational Hard Stop block. The dependency on **Translational_Hard_Stop.x** causes **Chassis.f** to also behave unexpectedly.

- 8 To generate accurate results using a 0.05 time step, you must remove the nonlinearity. Because the Translational Hard Stop block in the model uses the Stiffness and damping applied smoothly through transition region, damped rebound setting by default, the block implements smoothing near zero, when the helicopter takes off. This smoothing introduces a nonlinearity that the partitioning solver resolves by splitting the force variables and position variables into different partitions.

Choose a setting that avoids nonlinearities. Set **Hard stop model** to Full stiffness and damping applied at bounds, damped rebound and reset the solver sample time to its original value. Finally, run the simulation again and open the Scope block to view the results.

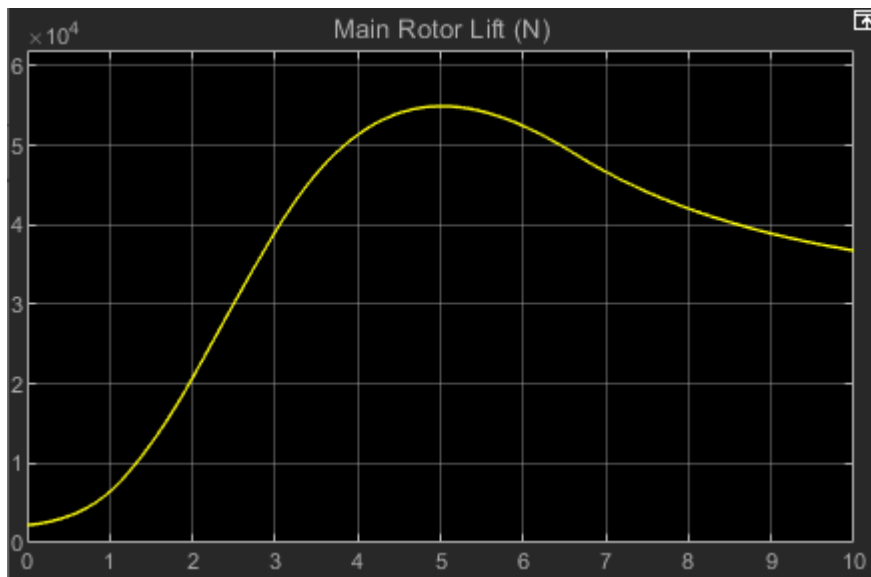
See Code

```
hardStop = 'Translational Hard Stop';
hardStopPath = [model, '/', hardStop];
open_system(hardStopPath)
set_param(hardStopPath, 'model', 'simscape.enum.hardstop.fulldamped')

open_system(solvConfigPath)
set_param(solvConfigPath, 'LocalSolverSampleTime', '0.05')

sim(model)
open_system(scopePath)
```

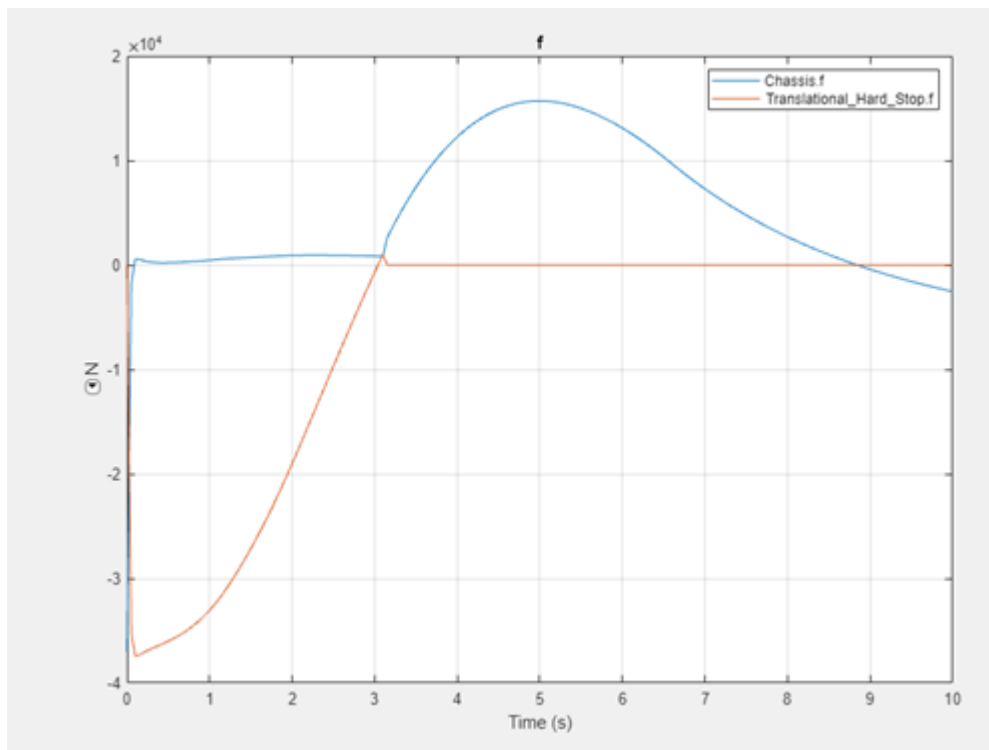
Removing the smoothing from the Translational Hard Stop block causes the simulation to generate the correct results using the original sample time.



- 9 Confirm that these actions remove the noise in the system. Open the Simscape Results Explorer and compare **Chassis.f** and **Translational_Hard_Stop.f** again.

See Code

```
sscexplore(simlog_sdl_transmission_helicopter_base);
```



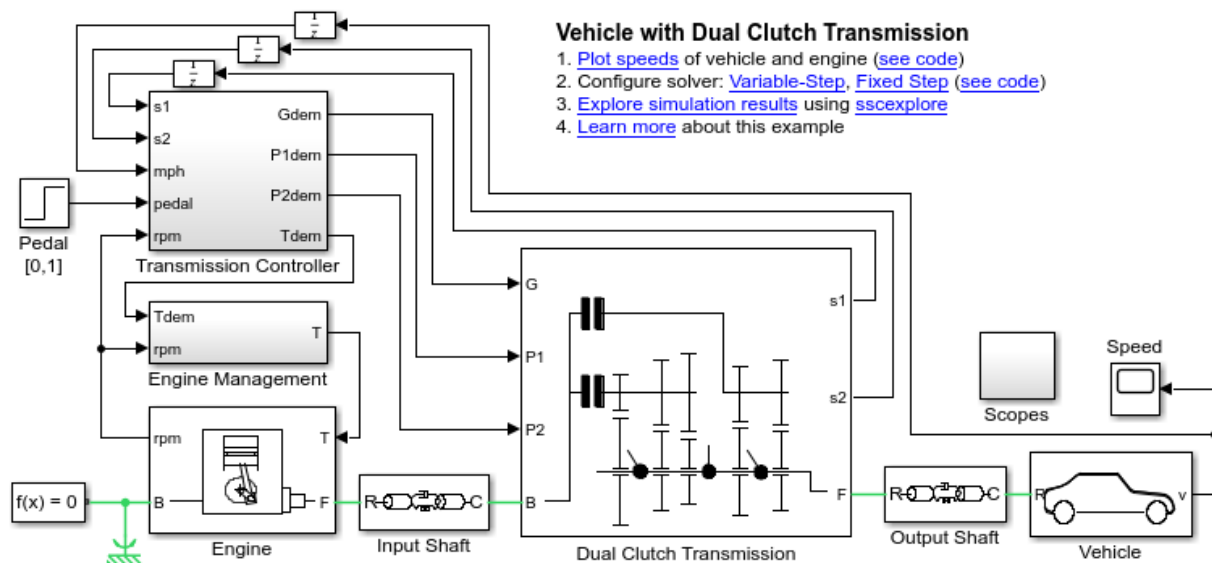
Noise is not present in the updated solver configuration for both **Chassis.f** and **Translational_Hard_Stop.f**.

Resolve Chatter Due to Stiffness

This example also shows how to resolve numerical difficulties that yield chatter in Simscape Driveline simulations that use the Partitioning solver. In this case, the chatter is caused by stiffness. A stiff model is one that contains both fast and slow dynamics.

- 1 Open the model with block. At the MATLAB command prompt, enter:

```
openExample('sdl/VehicleWithDualClutchTransmissionExample')
model = 'VehicleWithDualClutchTransmission';
```



The model is configured for a variable-step simulation that uses the global solver. Data logging is enabled only for the signal that contains the gear state data.

- 2 Configure the model for data logging. For this example, tire slip is the data of interest and logging additional data increases the computational cost of simulation. Enable data logging for tire slip and disable data logging for the gear state.
 - a In the Vehicle subsystem, the Tire LF Tire (Magic Formula) block represents the left-front tire of the vehicle. The Tire LF **S** port, which transmits the tire slip data, is a physical signal port. You can log physical signal data using Simulink data logging. The destination for the signal is the Tire slip Scope block, which is in the Scopes subsystem.
 - i Open the Scopes subsystem.
 - ii The tire slip data is in the signal that the FrontSlip From block transmits to the Tire slip Scope block. Select the signal, and in the Simulink toolstrip, on the **Signal** tab, click the arrow on the right side of the **Monitor** section. In the **Signal Monitoring** category, click **Log Signals**.

See Code

```

%% Enable Tire Slip Data Logging
% Enable Logging for Front Left Tire Slip Signal

% Define Scope Subsystem and path
scopesSS = 'Scopes';
scopesSSPath = [model, '/', scopesSS];

%% Define Front Slip From Tag Signal
fromTireLFLogSlip1 = 'From8';
fromTireLFLogSlip1Path = [scopesSSPath, '/', fromTireLFLogSlip1];

%% Enable the Front Slip From Tag Signal for
% Simulink(TM) data logging and viewing with the Simulation Data Inspector

fromTireLFLogSlip1PortHandles = get_param(fromTireLFLogSlip1Path, ...
    'PortHandles');
fromTireLFLogSlip1Outputport = fromTireLFLogSlip1PortHandles.Outputport;
set_param(fromTireLFLogSlip1Outputport, 'DataLogging', 'on')

```

- b** In the Transmission Controller subsystem, in the Shift state subsystem, the z3 Unit Delay block transmits the gear state.
 - i** Open the Transmission Controller subsystem.
 - ii** Open the Shift state subsystem.
 - iii** The Gear state G subsystem transmits the gear state to a Unit Delay block, which, in turn, transmits the data to the Gear state Outputport block. The signal that connects the Unit Delay block to the Gear state Outputport block is marked for data logging. Select the signal, and in the Simulink toolstrip, on the **Signal** tab, click the arrow on the right side of the **Monitor** section. In the **Signal Monitoring** category, click **Log Signals**.

See Code

```

%% Disable Gear State Data Logging

% Define Transmission Controller Subsystem and Path
transCntrl = 'Transmission Controller';
transCntrlPath = [model, '/', transCntrl];

% Define Shift State Subsystem and path
shiftState = 'Shift state';
shiftStatePath = [transCntrlPath, '/', shiftState];

% Define Unit Delay block and path
unitDelay = 'z3';
unitDelayPath = [shiftStatePath, '/', unitDelay];

% Disable Unit Delay block signal logging
unitDelayPortHandles = get_param(unitDelayPath, 'PortHandles');
unitDelayOutputport = unitDelayPortHandles.Outputport;
set_param(unitDelayOutputport, 'DataLogging', 'off')

```

- 3** Obtain and examine the baseline results. Simulate using the global variable-step solver and review the results in the Simulation Data Inspector.
 - a** Run the simulation. In the Simulink toolstrip, on the **Simulation** tab, in the **Simulate** section, click **Run**.

See Code

```

%% Simulate the Model
sim(model)

```

- b** Open the Simulation Data Inspector. In the Simulink toolstrip, on the **Simulation** tab, click the arrow on the right side of the **Review Results** section, and, in the **Signal Logging Results** category, click **Signal Logging Results**. To inspect the tire slip data, select the **From8:1** check box.

See Code

```

%% Examine Baseline Results in the Simulation Data Inspector

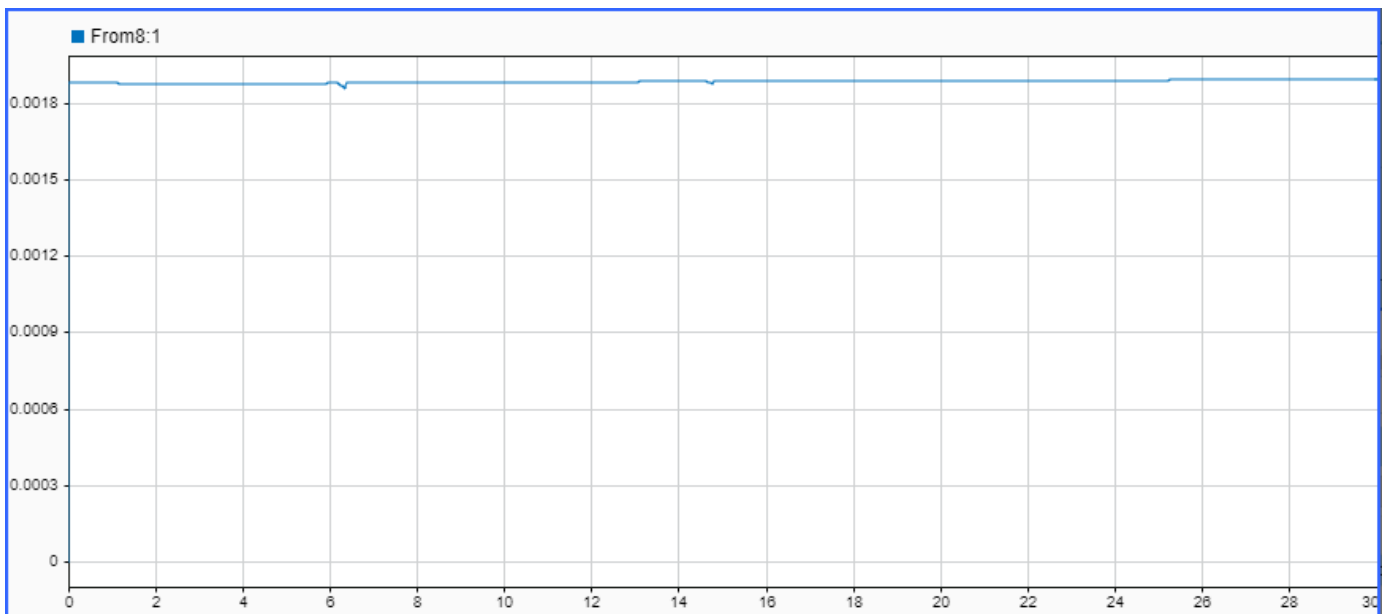
% Define Baseline Run
runIDs = Simulink.sdi.getAllRunIDs;
baselineRunID = runIDs(end);
baselineRun = Simulink.sdi.getRun(baselineRunID);
baselineRunTireLFSlip = baselineRun.getSignalByIndex(1);

% Set Line Color
baselineRunTireLFSlip.LineColor = [0 0.4470 0.7410];

% Select Signal (From8:1) Check Box
baselineRunTireLFSlip.Checked = true;

%% Open Simulation Data Inspector
Simulink.sdi.view

```



- 4 Configure the local solver for fixed-step simulation using the Partitioning solver.
 - a In the model, open the Solver Configuration block settings.
 - b Select the **Use local solver** check box.

See Code

```

%% Define the Solver Configuration Block and Path
solvConfig = 'Solver Configuration';
solvConfigPath = [model, '/', solvConfig];

%% Configure the Solver Configuration Block
set_param(solvConfigPath, ...
    'UseLocalSolver', 'on', ...
    'DoFixedCost', 'on')

```

- 5 Simulate using the Partitioning solver.

See Code

```

%% Simulate the Model
sim(model)

```

The simulation runs to completion, but generates a warning.

Warning: Simscape succeeded in finding consistent states with which to start the simulation, but the states found may deviate from requested initial conditions.

- 6 Compare the baseline and Partitioning solver results in the Simulation Data Inspector.
 - a Open the Simulation Data Inspector.
 - b In the top, left pane, select **Compare**.
 - c Configure the comparison. In the top, right pane:
 - i On the right side of the **Baseline** setting, click the down arrow and select Run 1: `sdl_vehicle_clutch`.
 - ii On the right side of the **Compare to** setting, click the down arrow and select Run 2: `sdl_vehicle_clutch`.
 - iii Click **Compare**.
 - d To change the Partitioning solver results line color, in the **Properties** pane, in the **Compare to** column, click the colored **Line**, select a different color, such as yellow, and then click **Set**.

See Code

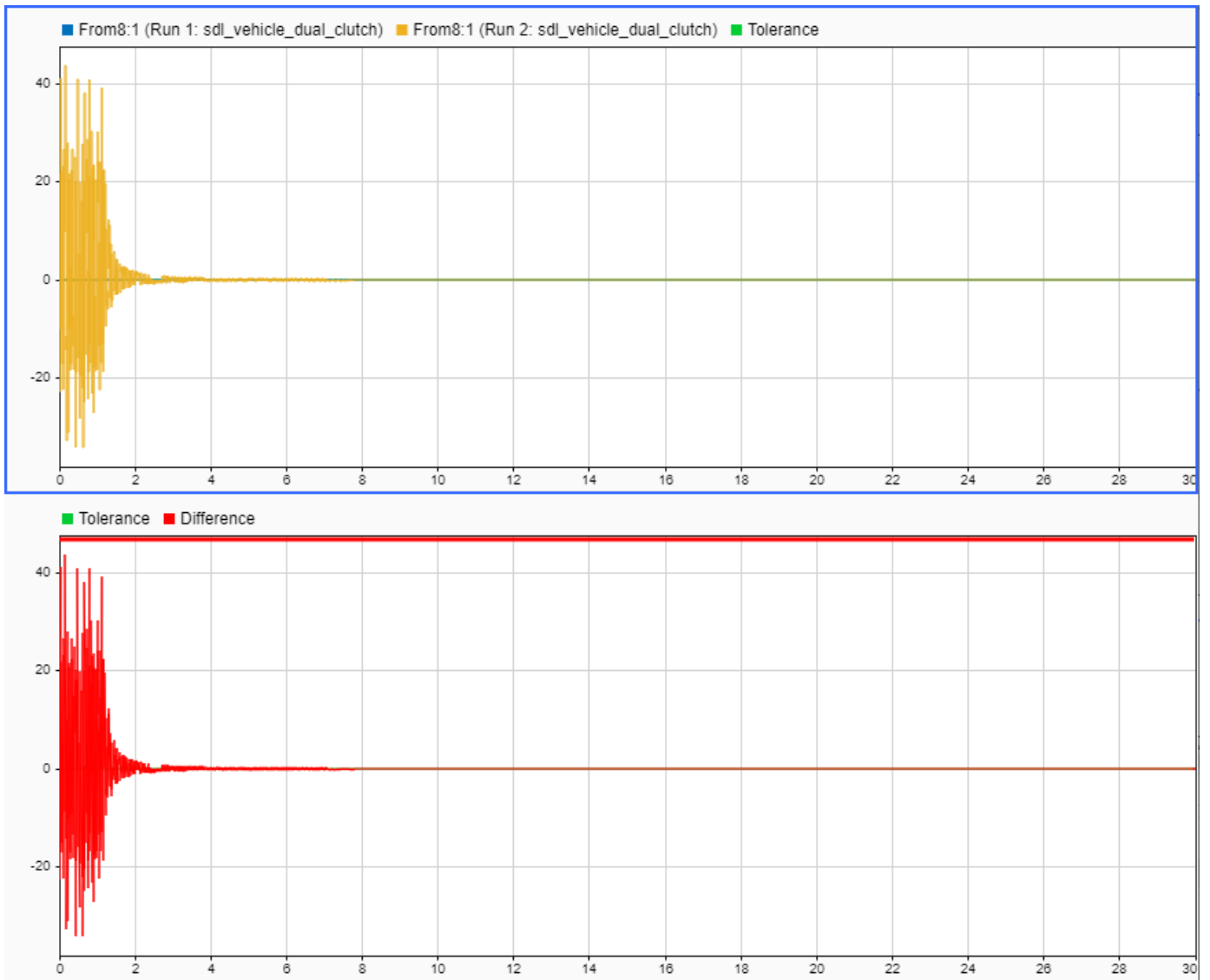
```
%% Compare Partitioning Solver Run 1 to Baseline Run
% Open Simulation Data Inspector
Simulink.sdi.view

% Define Partitioning Solver Run 1
runIDs = Simulink.sdi.getAllRunIDs;
pSolverRunIID = runIDs(end);
pSolverRun1 = Simulink.sdi.getRun(pSolverRunIID);
psolverRun1TireLFSlip = pSolverRun1.getSignalByIndex(1);

% Select From8:1 Check Box
% psolverRun1TireLFSlip.Checked = true;

% Set Line Color
psolverRun1TireLFSlip.LineColor = [0.9294    0.6941    .1255];

% Compare Runs
compBaselinePartition1 = Simulink.sdi.compareRuns(baselineRunID, ...
    pSolverRunIID);
```



- The results from the Partitioning solver simulation contain significant chatter.
- 7 Chatter in the front tire slip is due to the parameters specified for the Tire (Magic Formula) block. To resolve the signal chatter and the initial condition warning, simplify the tire dynamics:
 - a Open the Vehicle subsystem.
 - b Open the Tire LF block settings.
 - c In the **Dynamics** settings, set **Inertia** to No Inertia.
 - d Using the same process, omit the inertia for the Tire RF, Tire LR, and Tire RR blocks.

See Code

```

%% Resolve Chatter
% Simplify Tire Dynamics by Removing Inertia

% Define Vehicle Subsystem and Paths
vehicle = 'Vehicle';
vehiclePath = [model, '/', vehicle];

```

```

%% Define Tire Blocks and Paths and Omit Tire Inertia
%% Define Tire LF
tireLF = 'Tire LF';
tireLFPath = [vehiclePath, '/', tireLF];

%% Set Tire LF Settings Dynamics > Inertia to "No inertia"
set_param(tireLFPath, 'model_inertia', '0')

%% Omit Inertia from Tire RF, LR, RR Blocks

% Tire RF
tireRF = 'Tire RF';
tireRFPath = [vehiclePath, '/', tireRF];
set_param(tireRFPath, 'model_inertia', '0')

% Tire LR
tireLR = 'Tire LR';
tireLRPath = [vehiclePath, '/', tireLR];
set_param(tireLRPath, 'model_inertia', '0')

% Tire RR
tireRR = 'Tire RR';
tireRRPath = [vehiclePath, '/', tireRR];
set_param(tireRRPath, 'model_inertia', '0')

```

8 Simulate and then examine and compare the results in the Simulation Data Inspector.

- a Run the simulation.

See Code

```
sim(model)
```

The simulation runs to completion, and does not generate any initial condition warnings.

- b Open the Simulation Data Inspector and compare the results from the Partitioning solver fixed-step simulation to the baseline results from the variable-step simulation. To configure the comparison, in the top, right pane:
 - i On the right side of the **Baseline** setting, click the down arrow and select Run 1: `sd1_vehicle_clutch`.
 - ii On the right side of the **Compare to** setting, click the down arrow and select Run 3: `sd1_vehicle_clutch`.
 - iii Click **Compare**.

See Code

```

%% Open the Simulation Data Inspector
Simulink.sdi.view

%% Define the baseline run and select the From tag check box
runIDs = Simulink.sdi.getAllRunIDs;
pSolverRun2ID = runIDs(end);
pSolverRun2 = Simulink.sdi.getRun(pSolverRun1ID);
pSolverRun2TireLFSlip = pSolverRun1.getSignalByIndex(1);
pSolverRun2TireLFSlip.LineColor = [0.9294 0.6941 .1255];
pSolverRun2TireLFSlip.Checked = true;

% Compare the results
compBaselinePartition2 = Simulink.sdi.compareRuns(baselineRunID, ...
pSolverRun2ID);

```



The results from the Partitioning solver simulation no longer contain significant chatter and are much more similar to the baseline results.

See Also

Solver Configuration

Related Examples

- “Increase Simulation Speed Using the Partitioning Solver”
- “Reduce Numerical Stiffness”
- “Reduce Fast Dynamics”
- “Reduce Zero Crossings”

More About

- “Variable Viewer”
- “Partitioning Solver Statistics”

Driveline Degrees of Freedom

In this section...

“About Driveline Degrees of Freedom and Constraints” on page 16-21

“Identify Degrees of Freedom” on page 16-21

“Define Fundamental Degrees of Freedom” on page 16-22

“Define Connected Degrees of Freedom” on page 16-23

“Define Constrained Degrees of Freedom” on page 16-24

“Actuate, Sense, and Terminate Degrees of Freedom” on page 16-27

“Count Independent Degrees of Freedom” on page 16-27

“Count Degrees of Freedom in a Simple Driveline with a Clutch” on page 16-28

About Driveline Degrees of Freedom and Constraints

Identifying rotational degrees of freedom (DoFs) is important for building and analyzing a driveline, particularly a complex system with many constraints and external actuations. Simulink represents driveline DoFs and other Simscape system variables as *states*, among all states of a model, including the pure Simulink states.

This section explains how to identify driveline DoFs, handle constraints, and extract the true or *independent* DoFs from a complete driveline diagram.

- The basic elements of a driveline diagram:
 - Connection lines
 - Constraints, including branchings
 - Dynamic elements
- Sensors and sources
 - Actuating drivelines with motion sources and recording motions with motion sensors
 - Terminating DoFs

Identify Degrees of Freedom

In a Simscape Driveline model, mechanical motions can be rotational or translational: motion around or along one axis. The simplest way to identify a driveline *degree of freedom* (DoF) is from an angular or linear velocity. A DoF represents a single, distinct angular or linear velocity. Each DoF responds to the torques and forces acting on the inertias and masses making up the driveline. Integrating Newton's equations of motion determines the angular and linear motions. Mechanical DoFs are properties of rotating inertias and translating masses. It is nonetheless consistent and simpler to identify a single Simscape Driveline DoF as a driveline axis with its connected inertias and masses.

To identify and count DoFs in a driveline, look at a Simscape Driveline diagram starting with its mechanical connection lines first, before considering its blocks. Driveline blocks modify the DoFs represented by connection lines by:

- Generating torques and forces that act relatively between driveline axes

- Adding constraints between driveline axes
- Imposing externally actuated torques, forces, and motions


For the basic rules of connection lines and ports, see “Build a Drivetrain Model” on page 2-3.

Define Fundamental Degrees of Freedom

The basic unit of driveline motion is the DoF represented by an unbroken mechanical connection line. Such lines represent idealized massless and perfectly rigid driveline axes.

Represented by Inertia blocks, rotating bodies with inertias are rigidly attached to and rotate with their axes. Represented by Mass blocks, translating bodies with masses are rigidly attached to and translate along their axes. A single connection line or a set of branched connection lines represents either rotational or translational motion and must be connected to either rotational or translational ports.

Driveline Axes as Fundamental Degrees of Freedom — Mechanical Ports

A connection line anchored by physical network connector ports  represents an idealized driveline axis. The connection line enforces the constraint that the two connected driveline components rotate or translate at the same angular or linear velocity, respectively.



You measure the angular or linear velocity of an axis with an Ideal Rotational Motion Sensor or Ideal Translational Motion Sensor block.

Defining Relative and Absolute Angles and Positions

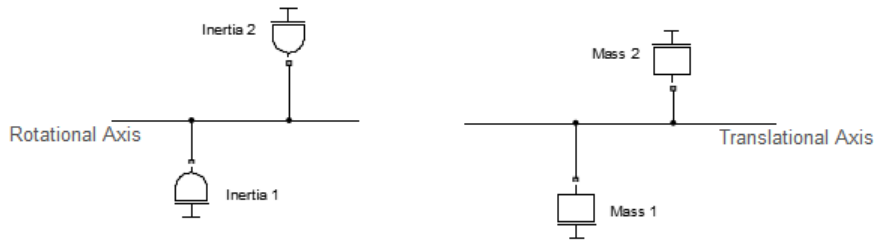
Relative angle or position is sometimes necessary to compute internally generated torques or forces between pairs of axes (see “Define Connected Degrees of Freedom” on page 16-23). To determine a relative angle or position, a motion sensor block integrates the relative angular or linear velocity of the pair of axes and adds the result to the initial relative angle or position specified in the block property inspector.

You can define an absolute rotation angle or translation position for a single axis when you measure its motion with a motion sensor, connecting the other physical connection port of the sensor to a Mechanical Rotational Reference or Mechanical Translational Reference. The sensor defines an absolute angle or position by integrating the velocity of the axis and adding the absolute reference angle or position that you provide in the motion sensor property inspector.

Rotating Inertias and Translating Masses Attached to Driveline Axes

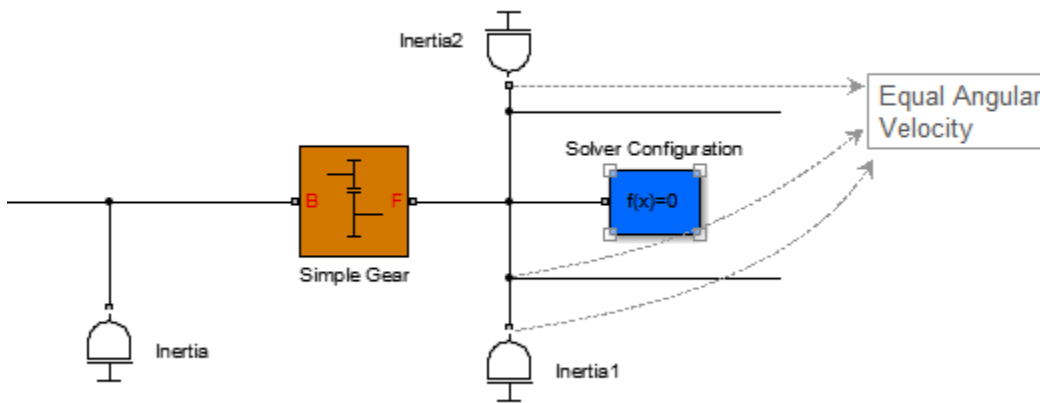
You cannot subject a driveline connection line, by itself, to any torques or forces, because it lacks inertia or mass. The other basic element to construct a functioning driveline model is one or more Inertia blocks, one or more Mass blocks, or both. In a real mechanical system, the spinning (or sliding) bodies carry both inertia (or mass) and DoFs.

You attach Inertias and Masses to mechanical connection lines by branching the lines. The attached inertias or masses are subject to whatever torque or force is transmitted by the connection line. The connection line imposes the constraint that everything attached to a single line must be spinning or sliding at the same velocity.



Driveline Axis Branching Rules and Constraints

You can branch connection lines. You can connect the end of any branch of a driveline connection line to a mechanical conserving connection port \square only. A set of unbroken, branched connection lines represents a single DoF.



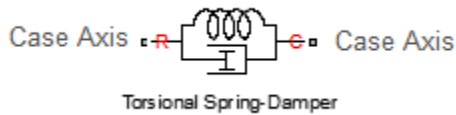
Branched Connection Lines and Angular Velocity Constraints

Define Connected Degrees of Freedom

You can connect two independent driveline axes, representing two independent degrees of freedom (DoFs), by an internal *dynamic element*. A dynamic element generates a torque or force from the relative angle, position, or motion of the two axes. This torque or force acts between the two axes, which remain independent DoFs, and which transmit the torque or force to their respective attached inertias or masses.

Dynamic Elements – Internal Torque and Force Generation

Apart from gears, most of the Simscape Driveline library blocks are dynamic elements, as are the mechanical rotational and translational blocks of the Simscape Foundation library. These blocks generate internal torques and forces. On a block with two mechanical conserving ports, a single torque or force is applied with positive sign to one axis and negative sign to the other axis. In this figure, torque is applied equally and oppositely to the rod and case axes of the Torsional Spring-Damper.



On blocks with more than two mechanical conserving ports, the total torques or forces flowing in and out of the block still sum to zero, but the torque or force is divided among the ports in a more complex way that depends on the driveline dynamics.

Clutch and Clutch-Like Elements — Conditional Connections

A clutch or clutch-like element is a *conditional* or *dynamic* constraint.

If unlocked, a clutch connects two driveline axes and can impose a relative torque between them, leaving the two axes independent. The unlocked clutch is either unengaged, imposing no torque at all; or engaged, imposing kinetic friction as a function of the relative velocity of the two connected axes.

If a clutch locks and applies only static friction between the two connected axes, the two axes are no longer independent. Instead, they act as a single axis, spinning at the same rate. See “Define Constrained Degrees of Freedom” on page 16-24.

Several other, clutch-like blocks also have locking and unlocking Coulomb friction:

- Torsional Spring-Damper
- Loaded-Contact Rotational Friction and Loaded-Contact Translational Friction

Define Constrained Degrees of Freedom

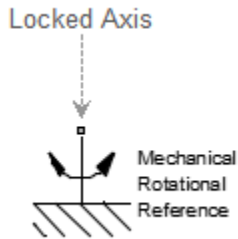
Certain driveline elements couple driveline axes in a way that eliminates their freedom to move independently. Such elements impose constraints on the motions of the connected axes. A constrained axis is no longer independent of other axes and does not count toward the total net or independent motions of the driveline. Such constraints remove independent degrees of freedom (DoFs) from the system.

Not all constraints are independent. Closing branched connection lines into loops makes some of the constraints within the loops redundant. The number of effective or independent constraints is the number of constraints arising from blocks minus the number of independent closed driveline connection line loops.

Except for clutches and clutch-like elements, driveline constraints are *unconditional* or *static* constraints; that is, unchanging over the simulation.

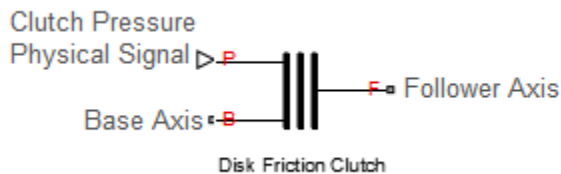
Locking a Driveline Axis

Connecting a driveline connection line to a Mechanical Rotational Reference or Mechanical Translational Reference block freezes the motion of the corresponding driveline axis. It cannot move, and its angular or linear velocity is constrained to be zero during a simulation. Such an axis has no associated independent DoF.



Locking Two Driveline Axes Together with a Clutch or Clutch-Like Element

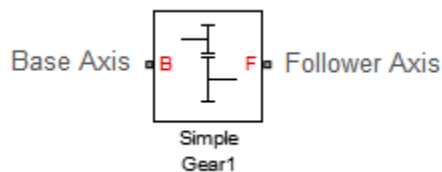
As long as the conditions for locking are valid, a locked clutch or clutch-like element constrains the two connected driveline axes to spin or slide together. The two axes remain distinct, but only one represents an independent DoF. The other is dependent.



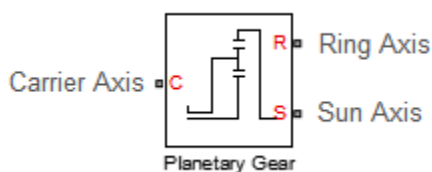
Even if it continues to apply kinetic friction between the axes, an unlocked clutch or clutch-like element no longer imposes a constraint. Instead, it acts as a dynamic element. See “Define Connected Degrees of Freedom” on page 16-23.

Coupling Driveline Axes with Gears

A gear coupling between two or more driveline axes reduces the independent DoFs of the driveline by imposing constraints. The nature of those constraints depends on the gear that you use. Gear blocks with two connected axes impose one such constraint and reduce the two axes to a single independent DoF.



Multiaxis gears impose more than one constraint. For example, a planetary gear imposes two constraints on three axes, reducing the axes to one independent DoF. (This count does not include the fourth, internal DoF, the planetary wheel, which is not connected to an axis with a mechanical port.)



Closed Loops, Effective Constraints, and Constraint Consistency

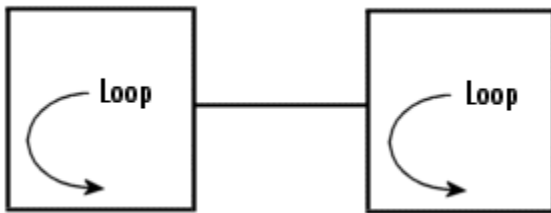
The actual constraint count to determine the number of DoFs is the number of effective or *independent* constraints. When connection lines form closed loops, take extra care in counting constraints in a driveline diagram. The presence of closed loops in a diagram reduces the effective constraint count by rendering some of the constraints redundant:

$$N_{\text{constr}} = N_{\text{bconstr}} - N_{\text{loop}}$$

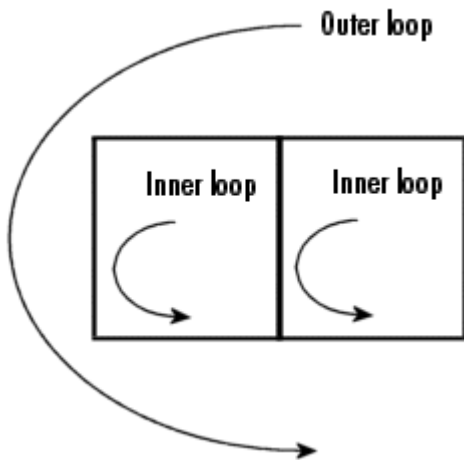
N_{constr}	Number of independent constraints
N_{bconstr}	Number of constraints from blocks
N_{loop}	Number of independent loops

You can reliably count the number of independent loops by counting the fundamental loops. Fundamental loops have no subloops. You can trace a fundamental loop with only one path. By counting only fundamental loops, you avoid overcounting loops that overlap.

For example, this diagram has two independent loops.



In this diagram, you can draw three loops: two inner loops, left and right, and the outer loop. The outer loop encompasses both inner loops.



There are two independent loops in this diagram, because only two are fundamental. The outer loop is not fundamental.

Consistency of Constraints

As long as all the velocities constrained by line branch points are equal over the whole loop, a closed loop renders redundant one of the constraints contained within it. (See “Driveline Axis Branching

Rules and Constraints” on page 16-23.) The velocities not directly connected by lines must also be consistent if, for example, they are transferred through gears.

If the velocities along a closed loop cannot be made consistent, the driveline is overconstrained and cannot move.

Actuate, Sense, and Terminate Degrees of Freedom

You can use Simscape Driveline and related blocks with only one driveline connector port \square to originate or terminate a physical connection line. Terminating a connection line limits the DoF.

Such blocks include:

- Inertia and Mass, which accept torque and force and respond with acceleration.
- Mechanical Rotational Reference and Mechanical Translational Reference, which ground DoFs to zero velocity.
- Vehicle Body, which implicitly connects the driveline to ground.

These blocks do not have to end a connection line, but can instead be branched off a connection line.

Directionality of Degrees of Freedom

Driveline connection lines have no inherent directionality. The direction of motion and torque flow is determined by the driveline dynamics when you simulate a model.

Effect of Torque and Force Actuation on Degrees of Freedom

Connecting an Ideal Torque Source or Ideal Force Source into a driveline connection line adds the torque or force specified by a physical signal input into that driveline axis. Such an actuation has no effect on the number of system DoFs. The driveline axes transmit torques and forces to their connected Inertias and Masses. The driveline is free to respond to these imposed torques or forces. The motion is simulated by integrating the driveline accelerations (a result of the imposed torques and forces) to obtain the driveline velocities.

Effect of Motion Actuation on Degrees of Freedom

Connecting an Ideal Angular Velocity Source or Ideal Translational Velocity Source to a driveline axis removes the freedom of that axis to respond to torques or forces. Instead, it specifies the axis motion during the simulation from the actuating physical signal input. Unlike torque actuation, motion actuation removes an independent DoF from the system.

For more information about driveline actuation with torques, forces, and motions, see “Driveline Actuation”.

Count Independent Degrees of Freedom

To determine the number of independent degrees of freedom (DoFs) in your driveline:

- 1 Count all the continuous, unbroken driveline connection lines (grouping connected sets of branched lines) in the Simscape Driveline portion of your model diagram. Call the total of such lines N_{CL} .

These lines connect two driveline connector ports or terminate on one mechanical connector port □. For details, see “Define Fundamental Degrees of Freedom” on page 16-22 and “Actuate, Sense, and Terminate Degrees of Freedom” on page 16-27.

- 2 Count all the constraints arising from blocks that impose constraints on their connected driveline axes. Call the total of such constraints N_{bconstr} .

Usually, each such block imposes one constraint, but complex gears impose more than one. For details, see “Define Constrained Degrees of Freedom” on page 16-24.

- 3 Count the number of independent loops, N_{loop} . The effective number of constraints is $N_{\text{constr}} = N_{\text{bconstr}} - N_{\text{loop}}$. For details, see “Closed Loops, Effective Constraints, and Constraint Consistency” on page 16-26.
- 4 Count all the motion actuations in your driveline, by counting each motion source block. Call the total of such motion actuations N_{mact} . For details, see “Actuate, Sense, and Terminate Degrees of Freedom” on page 16-27.

The number N_{DoF} of independent DoFs in your driveline is:

$$N_{\text{DoF}} = N_{\text{CL}} - N_{\text{constr}} - N_{\text{mact}} = N_{\text{CL}} - [N_{\text{bconstr}} - N_{\text{loop}}] - N_{\text{mact}}$$

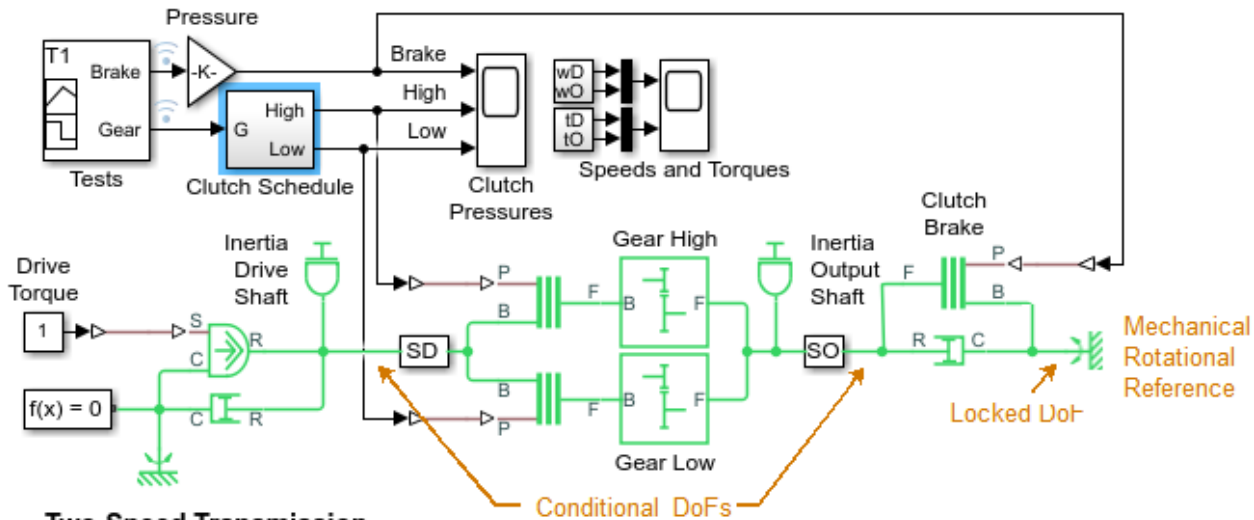
A necessary (although not sufficient) condition for driveline motion and successful driveline simulation is that N_{DoF} is positive. Count rotational and translational DoFs separately.

Conditional Degrees of Freedom with Clutches and Clutch-Like Elements

Unlike other driveline components, clutches, and clutch-like elements can undergo a discontinuous state change during a simulation. In general, the number of independent DoFs of a driveline is not constant during its motion. Each state change of one or more clutches changes the independent DoF count. Taken as a whole, different collective states of the clutches of a driveline can have different total net DoFs. To understand a driveline completely, examine each possible collective state of its clutch states to identify its independent DoFs and possibly invalid configurations.

Count Degrees of Freedom in a Simple Driveline with a Clutch

Consider the “Two-Speed Transmission” on page 18-143 example model.



Two-Speed Transmission

1. [Plot shaft speeds](#) (see code)
2. [Explore simulation results](#) using [sscexplore](#)
3. [Learn more](#) about this example

Simple Transmission

This system has five apparent DoFs, represented by these driveline axes:

- Branched axis with the Inertia Drive Shaft block
- Branched axis with the Inertia Output Shaft block
- Axis connecting the high gear clutch block (the clutch for the high clutch schedule) to the Gear High block
- Axis connecting the low gear clutch block (the clutch for the low clutch schedule) to Gear Low block
- Axis connecting the Clutch Brake block to the Mechanical Rotational Reference block (rotational ground)

There is an apparent closed loop formed by the gear blocks and gear clutch blocks. This loop is real only if both gear clutch blocks are locked.

The actual number of independent DoFs depends on the state of the clutches. The model has no motion sources, so we need consider only gears and clutches as constraints:

- The two gears blocks are always acting, therefore yielding two ever-present constraints.
- The fifth axis is always connected to the housing (rotational ground).

These three constraints reduce five DoFs to two DoFs.

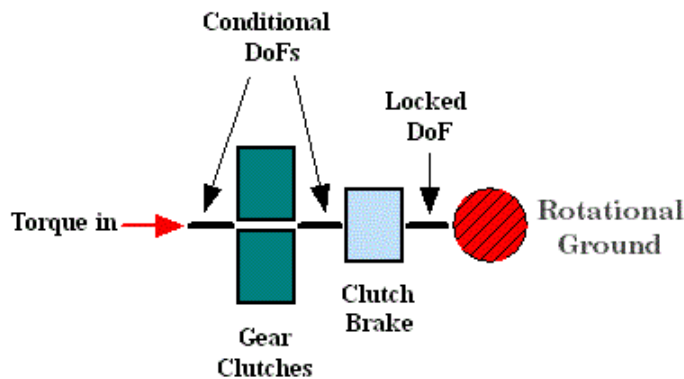
Now consider the clutches.

- Consider first the case where the Clutch Brake block is disabled (free).

- If both the high gear clutch and low gear clutch blocks are unlocked, the system has two independent DoFs, one on the left of the gear clutch blocks and the other between the gear clutch blocks and the Clutch Brake block.
- If one of these gear clutch blocks is locked, the additional constraint reduces the system to one independent DoF, everything to the left of the Clutch Brake block. (The clutch control schedule is set up to prevent both of these clutch blocks from being locked at the same time.)
- If the Clutch Brake block is enabled, the clutch control schedule keeps the two gear clutch blocks disabled.
 - If the Clutch Brake block is unlocked, the driveline has two independent DoFs: to the left of the gear clutch blocks and between the gear clutch blocks and the Clutch Brake block.
 - If the Clutch Brake block is locked, the system is reduced to one DoF, to the left of the gear clutch blocks. Everything to the right of the gear clutch blocks is locked to the housing.

This table and abstract diagram summarize the possibilities available in this model.

Brake Enabling	Clutch Locking	Independent DoFs
Brake disabled	Both gear clutch blocks unlocked	Two: On the left and on the right of the gear clutch blocks
	One gear clutch block locked	One: On the left of the Clutch Brake block
Brake enabled	Clutch Brake block unlocked	Two: On the left and on the right of the gear clutch blocks
	Clutch Brake block locked	One: On the left of the gear clutch blocks



Degrees of Freedom in the Simple Transmission

Nonphysical Configurations

The clutch schedule design implemented in the Clutch Schedule subsystem excludes nonphysical configurations. It is worth considering them anyway, for the sake of a complete understanding of driveline design. For more information about clutch issues, see “Troubleshoot Driveline Modeling and Simulation Issues” on page 17-2 and “Modeling Transmissions” on page 9-5.

Both Gear Clutches Locked, Clutch Brake Unlocked

This configuration creates a conflict of DoFs and reduces the independent DoFs to one. The driveline axis to the right of the gear clutch blocks tries to spin at two different rates, as required by two

different gear ratios. Two locked clutches enforce two additional constraints on the two remaining DoFs, but form a closed loop, nominally leaving one freedom in the mechanism. Because of the DoF conflict, attempting to simulate such a configuration leads to a Simscape Driveline error.

If the two Gears had identical gear ratios, the DoFs would not conflict, and the simulation would run without error.

One Gear Clutch Locked, Clutch Brake Locked

This configuration also creates a conflict of DoFs and yields zero DoFs. The two locked clutches enforce two additional constraints on the two remaining DoFs and leave no freedom in the mechanism. Driven by the driveline axis to the left, the driveline axis between the gear clutch blocks tries to spin but finds itself locked to the Mechanical Rotational Reference. Attempting to simulate such a configuration leads to a Simscape Driveline error.

Both Gear Clutches Locked, Clutch Brake Locked

This configuration is also overconstrained. Three locked clutches enforce two effective constraints on the remaining two DoFs (after taking into account the closed loop) and yield $N_{\text{DoF}} = 0$. In addition, the driveline axis to the right of the gear clutch blocks tries to spin at two different nonzero rates, while remaining locked to the Mechanical Rotational Reference, creating two distinct DoF conflicts.

Driveline States & Effect of Clutches

In this section...

“Driveline States and Degrees of Freedom” on page 16-32

“Find and Use Driveline States” on page 16-32

Driveline States and Degrees of Freedom

It is best to have some familiarity with advanced Simulink modeling techniques before using this section. For more information on driveline degrees of freedom, see “Driveline Degrees of Freedom” on page 16-21.

Simulink and Simscape represent driveline degrees of freedom (DoFs) and other information about the dynamics of a model with *states*. The driveline states are a subset of the total states of the model. Although the number of independent driveline states in a model is equal to the number of independent DoFs (with all clutches unlocked), the driveline states in general are linear combinations of the velocities, not the velocities of particular driveline axes. Before you simulate a model, this DoF-to-state transformation is not known.

You can extract state and model output data from your simulation. In the **Model Configuration Parameters** property inspector, select the appropriate check boxes in the **Data Import/Export** pane. The default state and output vectors are `xout` and `yout`, respectively.

Discontinuous Clutch State Changes

In part, the overall state of the driveline is the set of all its clutch states. Because clutches are dynamic constraints, the nature of the driveline states in a model with clutches and clutch-like elements can change during simulation. When a clutch locks, two independent driveline states become dependent on one another.

For software to design and analyze transitions among discontinuous states such as those found in clutches and transmissions, see .

Inverse Dynamics

State information is also useful for analyzing the *inverse dynamics* of a driveline. Often, you apply torques and forces to a driveline in *forward dynamics* and then determine the motions. Inverse dynamics means specifying motions to determine what torques and forces produce those motions.

If you motion-actuate some parts of your driveline instead, those axes and the equivalent states are no longer independent. If you want outputs from these axes, measure the torques and forces flowing along them. Knowing these torques and forces is the starting point of inverse dynamic analysis.

Find and Use Driveline States

This section explains how you locate and use Simscape Driveline states.

Locating Driveline States in Simulink

Your driveline model consists of a mixture of Simscape Driveline, Simscape, and ordinary Simulink blocks. In general, a model has Simulink states associated with the Simulink blocks. The Simscape

Driveline and Simscape states of a single driveline system are associated with the Solver Configuration block of that driveline.

You can list all model states with the Simulink `Simulink.BlockDiagram.getInitialState` method:

- 1 Open the “Simple Gear” on page 18-107 model as an example.
- 2 At the command line, enter:

```
sigt = Simulink.BlockDiagram.getInitialState('SimpleGear');  
sigt.time  
sigt.signals
```

The `Simulink.BlockDiagram.getInitialState` method initializes the model at zero time and captures the model states within the `.signals` structure. This list is the total set of states, not just the independent states. The Simscape and driveline states are a subset of the total states.

Trimming and Linearization — Clutch States

An important part of analyzing a driveline system is finding stable steady states of motion and understanding how the driveline responds to small changes in inputs, such as changes to initial conditions or to the applied forces and torques. Trimming and linearization are the formal steps of such an analysis.

If you implement clutch state changes in your simulation, trimming requires that you start by specifying which clutches are locked and unlocked. The trimming procedure then determines the state of continuous motion. During linearization, simulation starts with the clutch states that you specify and iterates to find a consistent state of all clutches. It then implements the perturbation of continuous states, holding the clutch states fixed.

For more information about trimming and linearizing Simscape models, see “Finding an Operating Point” and “Linearizing at an Operating Point”.

How Simscape Driveline Simulates a Drivetrain System

In this section...
“About Simscape Driveline and Simscape Simulation” on page 16-34
“Clutch State Determination” on page 16-34

About Simscape Driveline and Simscape Simulation

Apart from clutches and clutch-like elements, Simscape Driveline simulation is a special case of Simscape simulation.

- For information on clutches, degrees of freedom, and states, see “Driveline Degrees of Freedom” on page 16-21 and “Driveline States & Effect of Clutches” on page 16-32.
- For information on fixing simulation errors, see “Troubleshoot Driveline Modeling and Simulation Issues” on page 17-2.
- On how Simscape models work, see:
 - “How Simscape Models Represent Physical Systems”
 - “How Simscape Simulation Works”

Clutch State Determination

During simulation, Simscape Driveline software checks the clutch and clutch-like blocks in your model for locking and unlocking events. If a locked clutch meets the criteria for unlocking, or an unlocked clutch the criteria for locking, the respective clutch states change.

If one or more clutch and clutch-like constraints change, the driveline states are repartitioned into new sets of dependent and independent states. The repartitioning requires a partial reinitialization of the driveline that preserves the state of the driveline before the clutch changes, except for the subset of constraints and states affected by the clutch transitions.

Model Thermal Losses in Driveline Components

In this section...

“Thermal Ports” on page 16-35

“Thermal-Modeling Parameters” on page 16-35

“Model Thermal Losses for a Simple Gear” on page 16-36

Thermal modeling provides data that helps you to design efficiency and thermal protection into your system. Certain blocks in the Simscape Driveline Brakes & Detents, Clutches, and Gears libraries have thermal variants that allow you to determine how heat generation affects the efficiency and temperature of driveline components. For example, the Simple Gear block, which models a gear of base and follower wheels, has a thermal variant that can simulate the heat generated by meshing losses. Selecting a thermal variant for a block adds a thermal port to the block and enables the associated thermal-modeling parameters.

Thermal Ports

Thermal ports are physical conserving ports in the Simscape thermal domain. You can model thermal effects like heat exchange and insulation by connecting blocks, from other Simscape products, that use the thermal domain to the thermal ports on Simscape Driveline thermal variants.

Thermal ports are associated with temperature and heat flow which are the Across and Through variables of the Simscape thermal domain. To measure thermal variables, you can use one or both of these methods:

- 1 Log simulation data using a Simscape logging node. View the data using the `sscexplore` function.
- 2 Add a sensor from the **Simscape > Foundation Library > Thermal > Thermal Sensors** library to your model. To measure temperature, use a parallel-connected Ideal Temperature Sensor block. To measure heat flow, use a series-connected Ideal Heat Flow Sensor block.

There are several advantages to using data logging for desktop simulation. Data logging is less computationally costly than using a gauge block and it allows you to:

- View post-simulation results easily using the Simscape Results Explorer.
- Output data easily to the MATLAB Workspace for post-processing analysis.

However, if you use only data logging to measure a variable, you cannot output a feedback signal for that variable to a control system during simulation as you can when you use only a sensor to measure the variable. Also, because data logging is not supported for code generation, you cannot use Simscape data logging when you perform real-time simulation on target hardware.

Thermal-Modeling Parameters

Thermal-modeling parameters are device-specific characteristics that determine how thermal dynamics affect device temperature and performance during simulation.

For some blocks, the default variant includes requisite parameters for simulating thermal dynamics. For such blocks, parameter dimensions change when you select a thermal variant. For example, to parameterize meshing losses based on a constant efficiency friction model for the default variant of

the Simple Gear block, you specify the **Efficiency** parameter using a scalar value. If you select a thermal variant for the Simple Gear block, you must use a vector quantity to specify the **Efficiency** parameter.

Selecting a thermal variant enables additional thermal-modeling parameters. For example, selecting the thermal variant of the Simple Gear block enables the **Temperature** parameter. To determine the extent of thermal losses, the block performs a table lookup based on the values that specified for the **Efficiency** and **Temperature** parameters.

Model Thermal Losses for a Simple Gear

This example shows how to enable a thermal port, parameterize a thermal variant, and analyze the results of a simulation that models thermal losses.

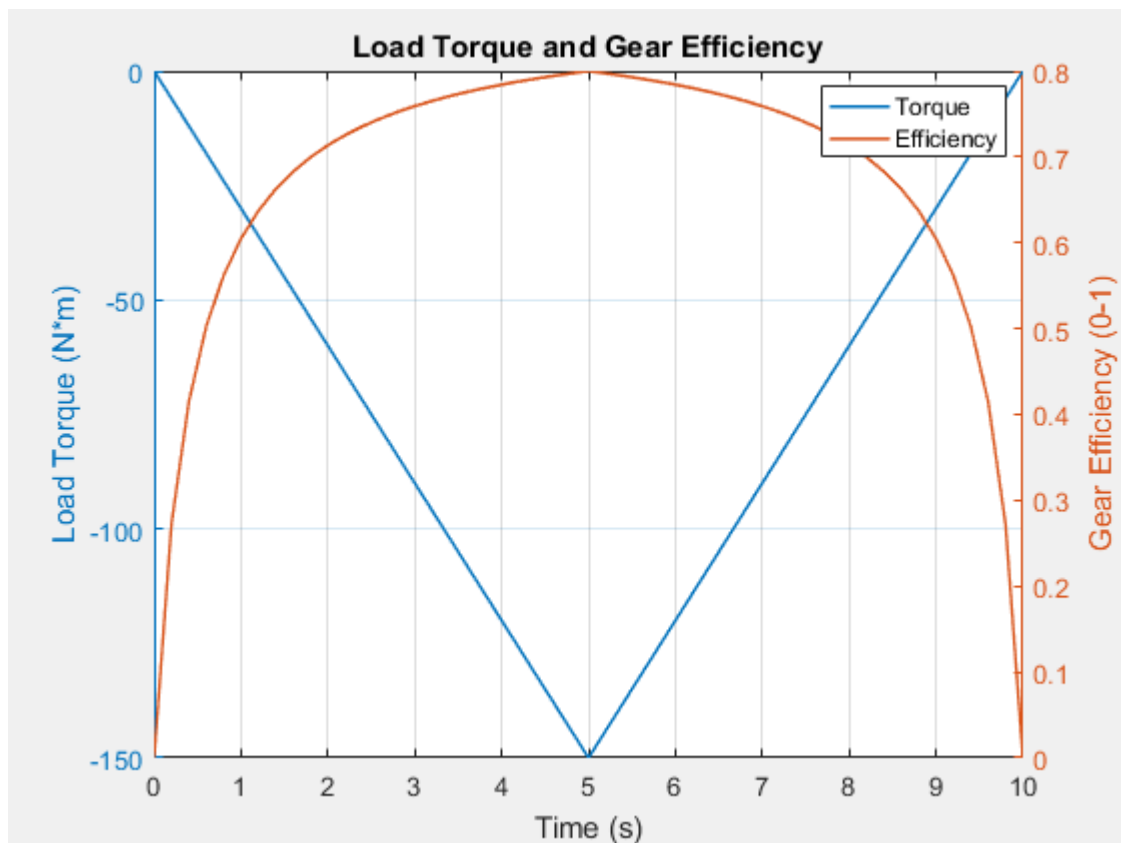
Measure Load-Dependent Efficiency

- 1 Open the model. At the MATLAB command prompt, enter

```
openExample('sdl/GearboxEfficiencyMeasurementExample')
```
- 2 Examine the parameters for the **Gear** block.

For **Meshing Losses**, the **Friction model** is set to Load-dependent efficiency. The **Nominal output torque** is 150 N*m and the **Efficiency at nominal output torque** is 0.8.

- 3 To simulate the model and plot gearbox efficiency, in the model window, click **Plot efficiency**.



The efficiency at the nominal point exactly matches the parameter values in the block. However, the efficiency is dependent only on the torque. Temperature does not factor into the efficiency calculation.

Use a Thermal Variant for the Gear Block

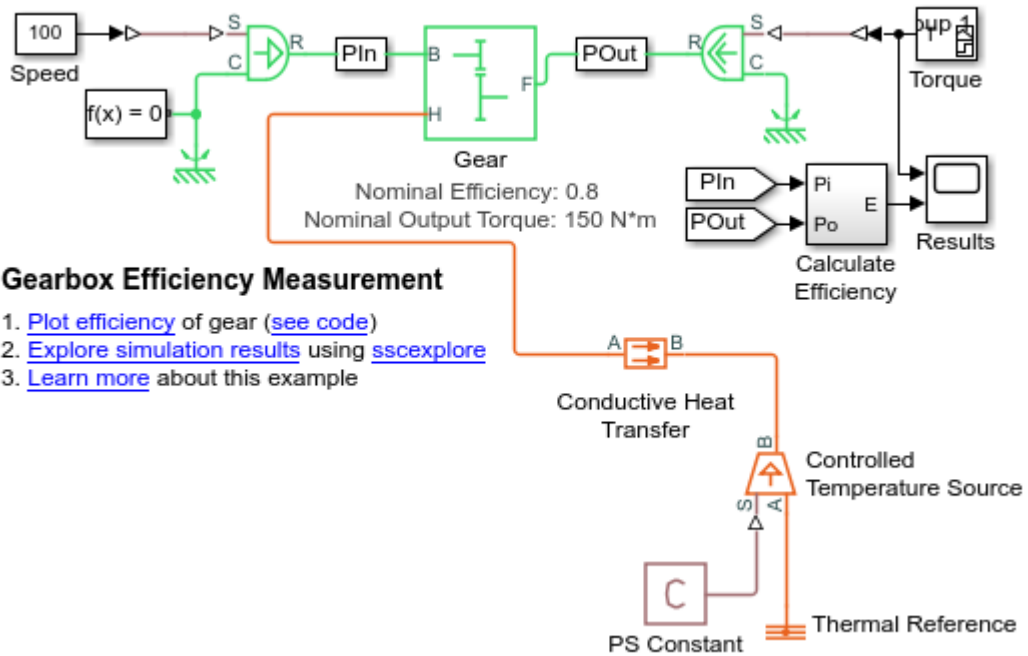
To include temperature as a factor in the efficiency calculation, select a thermal variant for the gear block.

- 1 Right-click the **Gear** block and, from the context menu, select **Simscape > Modeling option**. Select **Show thermal port**.
- 2 Parameterize the thermal variant. If the block property inspector is open, close and then open it to make the thermal parameters visible. For the **Meshing Losses** parameters:
 - a Set **Friction Model** to Temperature and load-dependent efficiency.
 - b For **Temperature**, specify [280 400 500].
 - c For **Efficiency matrix**, specify [0.65 0.65 0.7; 0.7 0.7 0.75; 0.75 0.75 0.8].
- 3 For the **Thermal Port > Initial Temperature** parameter, specify 320.

Add Thermal Library Blocks

To model heat transfer, add blocks from the Simscape Foundation Thermal library.

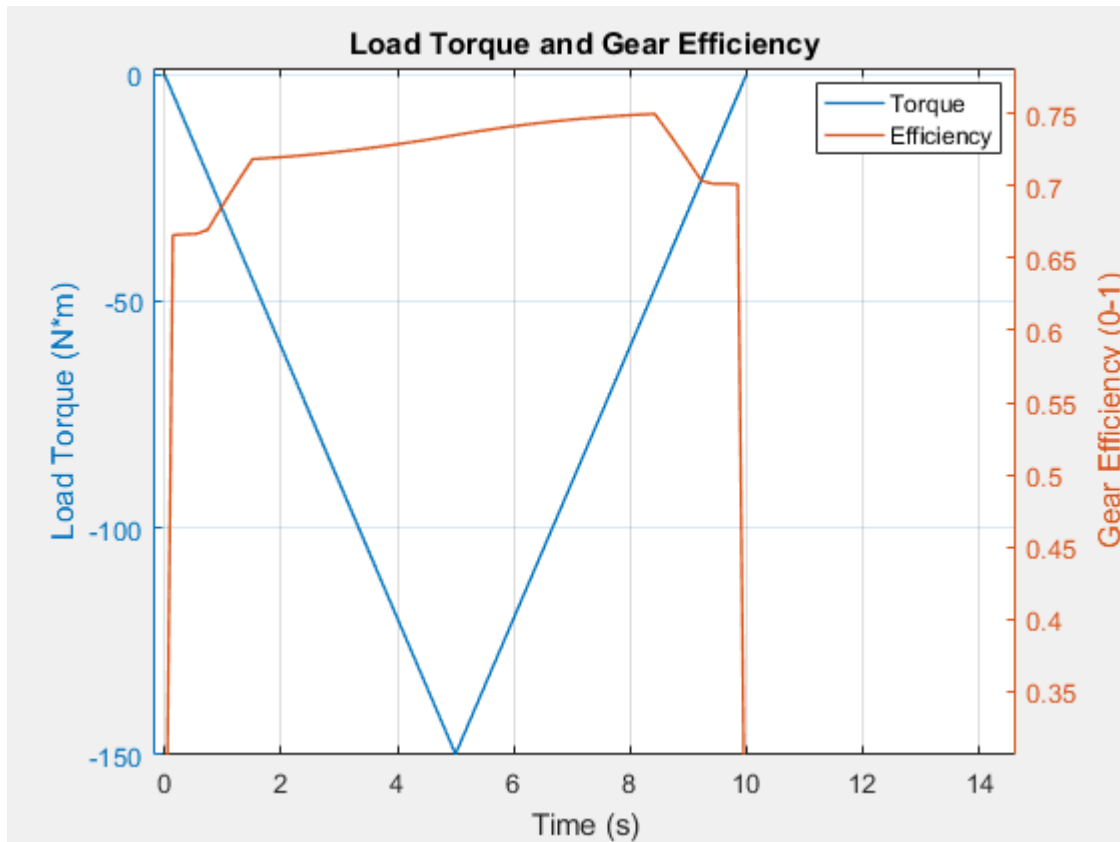
- 1 Add a block that represents heat flow between the gear and the environment. Open the Simulink Library Browser. From the **Simscape > Foundation Library > Thermal > Thermal Elements** library, add a Conductive Heat Transfer block to the model.
- 2 Add a block that represents a thermal reference point. Also from the **Thermal Elements** library, add a Thermal Reference block to the model.
- 3 Add blocks for modeling the ambient temperature as a constant, ideal source of thermal energy.
 - From the **Simscape > Foundation Library > Thermal > Thermal Sources** library, add an Controlled Temperature Source block.
 - From the **Simscape > Foundation Library > Physical Signals > Sources** library, add a PS Constant block. Specify a value of 320 for the PS Constant block.
- 4 Arrange and connect the blocks as shown in the figure.



Measure Temperature and Load-Dependent Efficiency

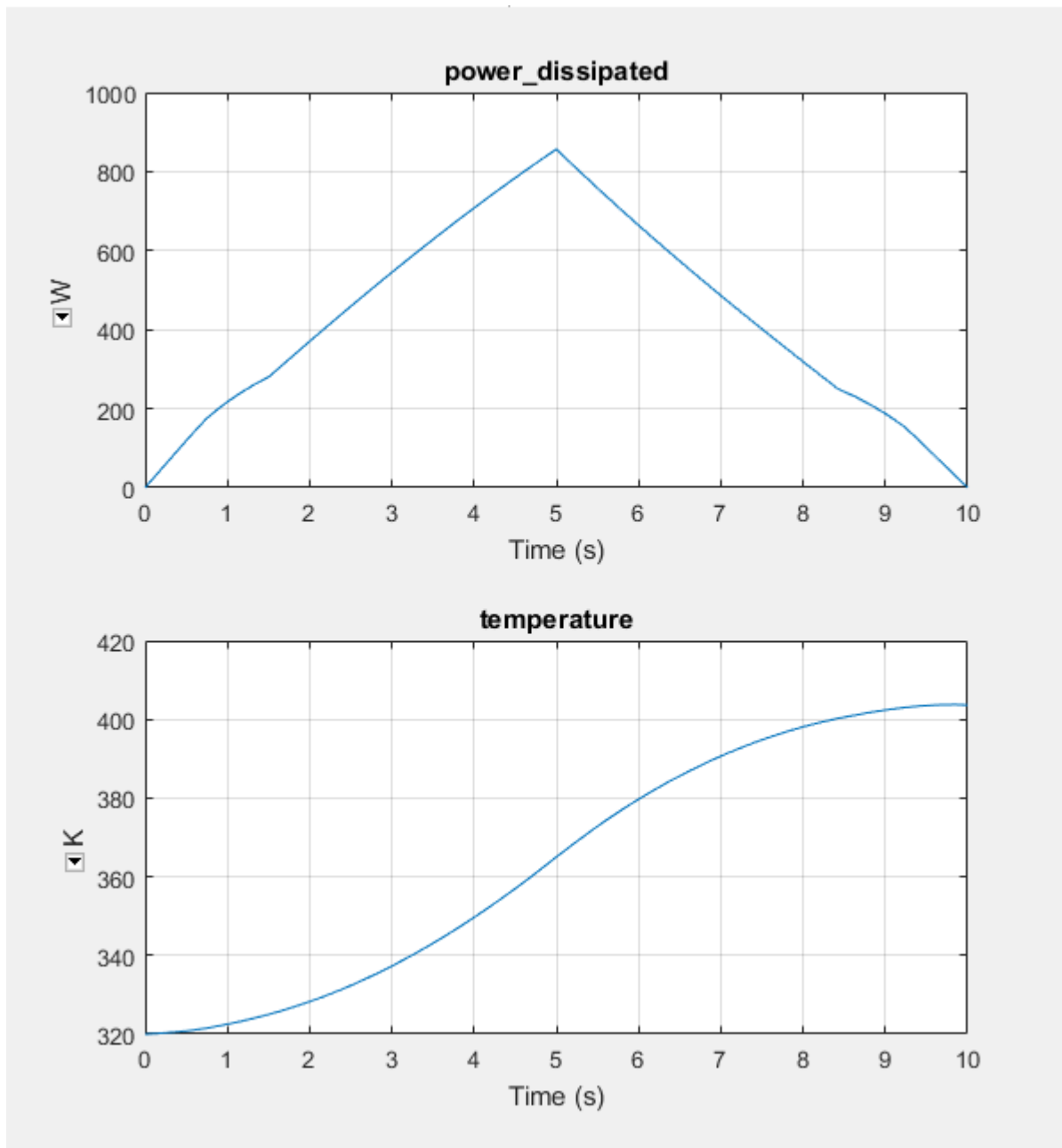
Evaluate efficiency as a function of both the load torque and the gear temperature.

- 1 Simulate the model.
- 2 To plot gearbox efficiency, in the model window, click **Plot efficiency**. Zoom out for a better view of the efficiency curve.



The efficiency peaks at 8.5 seconds when the magnitude of the load torque is approximately 33% of its maximal value. The efficiency is no longer dependent only on the load torque.

- 3 To view the data for the dissipated power and for gear temperature:
 - a In the model window, click **Explore simulation results**.
 - b In the node tree window, expand the **Gear > simple_gear_model** nodes.
 - c CTRL + click the **power_dissipated** and **temperature** nodes.



As the magnitudes of the torque load and temperature increase during the first half of the simulation, so does the amount of power that is dissipated. The amount of power that is dissipated decreases in the second half of the simulation due to the decrease in the torque load. However, the temperature of the gear continues to rise, as does the efficiency due to the vector specified for the **Temperature** parameter. The efficiency depends on both the load torque and the gear temperature.

See Also

Simple Gear

More About

- “About Simulation Data Logging”

Simscape Driveline Limitations

In this section...
“Simscape Driveline and Simulink Limitations” on page 16-42
“Additional Simscape Driveline Limitations” on page 16-42

Simscape Driveline and Simulink Limitations

Simscape Driveline software shares the limitations of Simscape software concerning Simulink features and tools. For Simscape limitations, see “Limitations”.

Additional Simscape Driveline Limitations

Simscape Driveline software also has additional limitations of its own.

Index-2 Differential-Algebraic Equations from Variable Ratio Transmission

If you use the Variable Ratio Transmission block in your model, you might make your simulation slower or less accurate, depending on where the variable ratio comes from. A physical signal input defines the variable ratio as a function of time.

- If this time function is defined autonomously, independently of the dynamics of the physical system, the variable ratio is not a simulation problem.
- If this time function is defined in terms of feedback from the physical system itself, the variable ratio makes the physical-mathematical model into an *Index-2 differential-algebraic equation* (DAE) system. Its solution requires two differentiations of constraints and results in simulation warnings or errors.

To avoid this problem, do one of the following:

- Remove the Variable Ratio Transmission blocks that use physical system feedback to define their variable ratios.
- Delay the feedback signal, so that the feedback is no longer instantaneous.

Troubleshoot Driveline Simulation Issues

- “Troubleshoot Driveline Modeling and Simulation Issues” on page 17-2
- “Troubleshoot Overconstrained and Conflicting Degrees of Freedom” on page 17-3
- “Troubleshoot Clutch and Transmission Errors” on page 17-4
- “Troubleshoot Inconsistent Initial Conditions” on page 17-5
- “Troubleshoot Pulley Network Issues” on page 17-6
- “Troubleshoot Engine Issues” on page 17-7

Troubleshoot Driveline Modeling and Simulation Issues

Various errors can cause your Simscape Driveline simulation to stop before completion. Some of these errors arise from nonphysical configurations of the driveline. To learn how to correct such issues, see:

- “Troubleshoot Overconstrained and Conflicting Degrees of Freedom” on page 17-3
- “Troubleshoot Clutch and Transmission Errors” on page 17-4
- “Troubleshoot Inconsistent Initial Conditions” on page 17-5
- “Troubleshoot Pulley Network Issues” on page 17-6
- “Troubleshoot Engine Issues” on page 17-7

Troubleshoot Overconstrained and Conflicting Degrees of Freedom

Analyzing and counting the driveline degrees of freedom (DoFs) are essential to fixing one type of simulation error. For more information, see “Driveline Degrees of Freedom” on page 16-21.

To run successfully, your driveline simulation must have a positive number of independent DoFs, from the start to the end of the simulation. Furthermore, the model DoFs must not conflict with each other.

If you encounter a simulation error where the driveline cannot move, check whether the number N_{DoF} of independent DoFs is positive, and whether the DoFs do not conflict with each other.

Checking the Number of DoFs

If N_{DoF} is not positive:

- Remove one or more constraining blocks, such as Gears, Clutches or clutch-like elements, and Mechanical Rotational References.
- Remove one or more Ideal Angular Velocity Source blocks.

Try one or both of these steps repeatedly until you locate the origin or origins of the simulation failure and make N_{DoF} positive.

Checking the Consistency of DoFs

Consider also whether two or more DoFs are in conflict. For example, check whether two velocity sources are trying to move a single DoF in two different ways. Such a configuration creates a motion conflict and leads to a simulation error.

Troubleshoot Clutch and Transmission Errors

Faulty clutch and transmission configurations generate many driveline motion failures and usually arise from DoF conflicts and errors. Clutches impose *conditional* or *dynamic* constraints.

To avoid or solve such problems, pay close attention to the collective state of your clutches, including clutches occurring inside transmission subsystems. The key to avoiding errors with transmissions is to work out and implement a complete and consistent clutch schedule.

Common mistakes include:

- Locking too many clutches simultaneously, leading to redundant dynamic constraints and overconstrained (not enough) DoFs.
- Locking conflicts among clutches, leading to nonredundant but still conflicting constraints.

Example: Locking one clutch locks one driveline axis to another. You could also lock the first driveline axis simultaneously to a third axis with another clutch. If the second and third axes cannot turn at the same velocity, these DoFs are in conflict.

- Locking too few clutches simultaneously. This error does not overconstrain DoFs or put them in conflict. However, it puts a transmission into a neutral state where it cannot transmit any torque or motion.

For information about adjusting simulation for clutches, see “Optimize Simulation of Clutches” on page 16-3 and “Transmissions with Gear Ratios and Clutch Schedules”.

Troubleshoot Inconsistent Initial Conditions

Like motion sources, initial conditions can cause motion conflicts. Unlike motion sources, they do not impose constraints or remove DoFs from the driveline, because they act only at the start of the simulation. However, under certain circumstances, initial conditions can cause errors when you start the simulation:

- Initial conditions conflict with one another.

Example: You couple two driveline axes through a Gear with a gear ratio of 2. The base axis must spin twice as fast as the follower, in the same direction. If you actuate the base with a velocity source, and the follower is connected to an inertia with initial velocity not set to half the initial base velocity, the simulation stops with an error.

- Initial conditions conflict with motion sources. When the simulation starts, the signal controlling a velocity source acting on an axis and the initial velocity value specified on an Inertia or a Mass attached to that axis must agree. Analogous requirements hold for velocities transformed by gear couplings.

Regardless of how you set the initial conditions of your driveline axes, the complete set of initial conditions must be consistent with itself. Driveline connection lines satisfying angular velocity constraints (for example, branched lines, or lines in closed loops) must have the same initial angular velocities.

Troubleshoot Pulley Network Issues

Like real-world pulleys, Simscape Driveline pulley blocks rely on belt tension and inertia for motion. To fix a pulley network in your model that is not acting as expected:

- Make sure that there are no conflicts in the settings for the belt direction. For more information, see “Best Practices for Modeling Pulley Networks” on page 6-2.
- Ensure that there is inertia in the system. If there is no inertia, add it by including an Inertia block from the Simscape Rotational Elements library, by adding inertia to a downstream component, or including inertia in one of the pulleys. As needed, specify an initial velocity for the inertia.
- Compare the number of pulley pairs to the number of tensioners. If the number of tensioners is not equal to either the number of pulley pairs or the number of pulley pairs less one, add tensioners. For example, if there are five pulley pairs, include at least four tensioners. Use spring and damper blocks to model the tensioners.

See Also

Belt Drive | Belt Pulley | Rope Drum | Rotational Free End | Inertia | Worm Gear | Translational Spring | Inertia | Variable Inertia

Related Examples

- “Power Window System” on page 18-88

More About

- “Best Practices for Modeling Pulley Networks” on page 6-2

Troubleshoot Engine Issues

Like real-world engines, blocks from the Simscape Driveline Engines library rely on the inertia from each cycle to initiate the next cycle. If a piston- or engine-driven network in your model is not responding to throttle input, either at the beginning of simulation or when the engine reaches stall speed, examine the engine output for the simulation. If there is no engine velocity in response to throttle input, try these engine-startup methods:

- Add initial velocity to the engine block — Specify a nonzero value for the initial velocity or crank velocity parameter in the engine block settings. Specify a value that is well above stall speed. Iterate to find the correct solution.
- Add an inertia with initial velocity to the engine network — Add an Inertia block from the Simscape Rotational Elements library or add inertia to a downstream component, for example a shaft. Specify an initial velocity using the Variables settings for the Inertia block or downstream component. Set the target velocity to **High Priority**.
- Add an electric starter motor — Use a starter motor, such as the DC motor in the “Permanent Magnet DC Motor” example, to initiate engine motion.

If a model that is not giving expected results contains a Piston block with the **Pressure parameterization** set to **By crank angle and throttle** and a **Pressure matrix (gauge)** that indicates zero velocity, include an external inertia with initial angular velocity.

See Also

Generic Engine | Piston | Piston Engine | Inertia

Related Examples

- “Permanent Magnet DC Motor”

More About

- “Troubleshoot Inconsistent Initial Conditions” on page 17-5

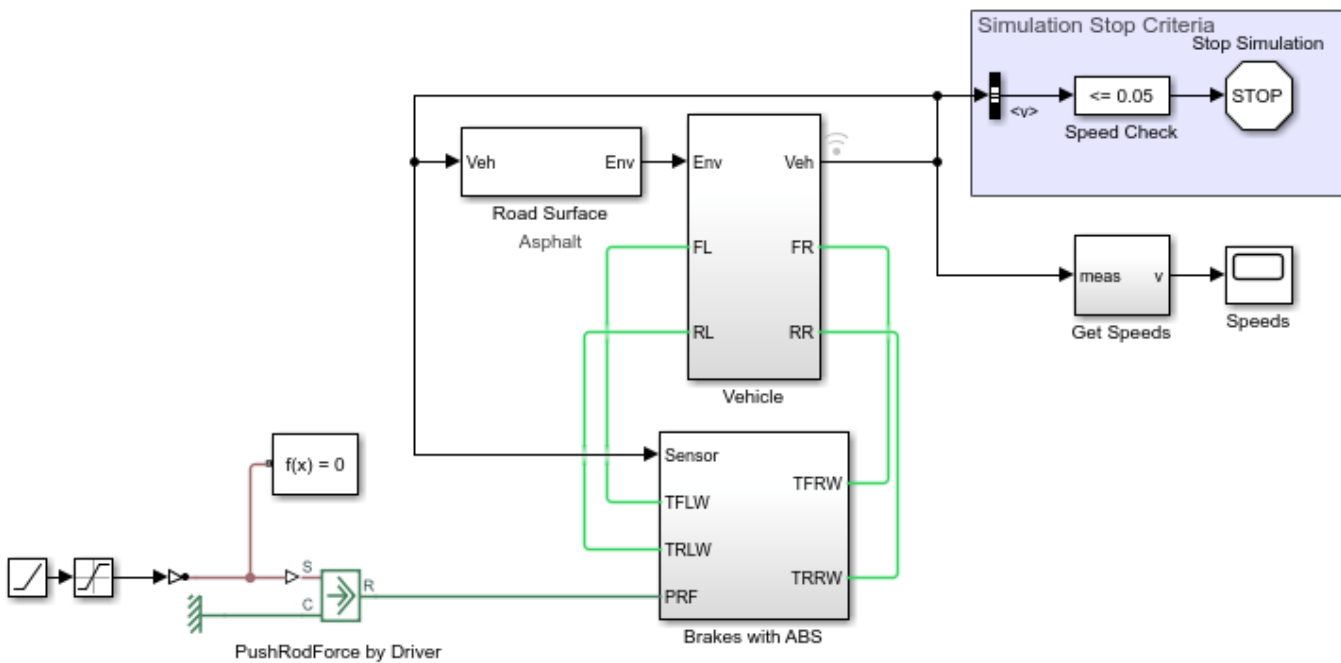
Simscape Driveline Examples

Anti-Lock Braking System (ABS)

This example shows a simple way of modeling an ABS braking system. The model shows the velocity profile responses achieved for the vehicle CG and the wheels.

Model

The following figure shows the model of an ABS system with a test harness. The output of the model is the plot of the velocity profile achieved for the vehicle CG and the wheels.



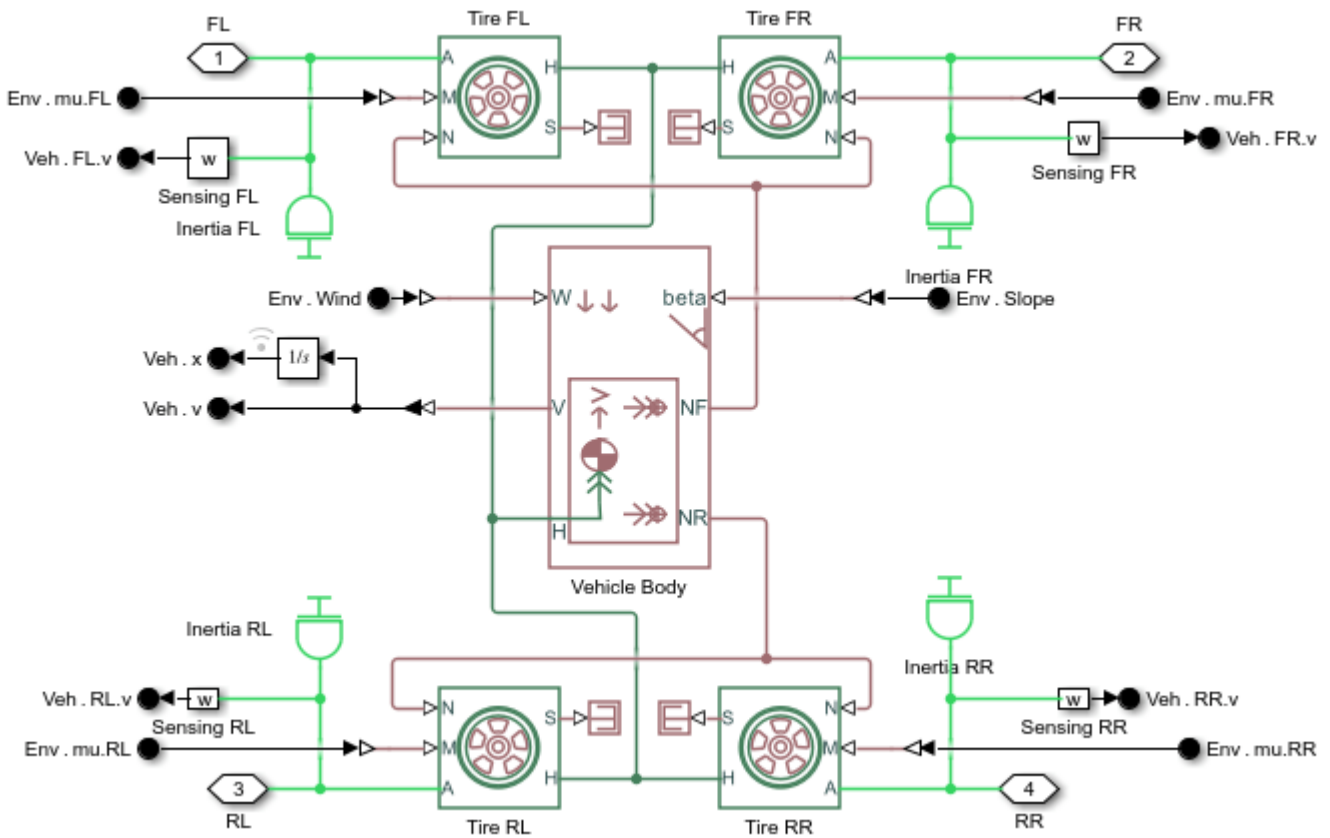
Anti-Lock Braking System (ABS)

1. Open initialization script
2. Plot results (see code) for vehicle CG and wheel velocity profile
3. Explore simulation results using Simscape Results Explorer
4. Learn more about this example

Copyright 2019-2022 The MathWorks, Inc.

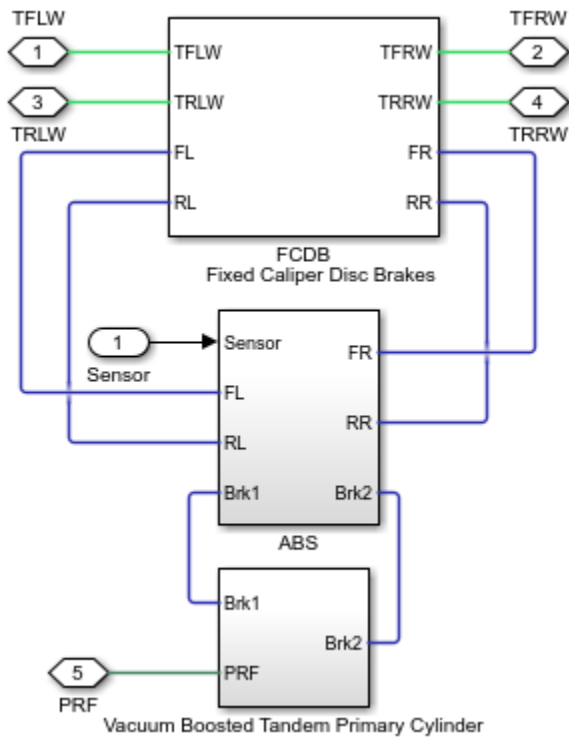
Vehicle Subsystem

This subsystem shows the modeling of the vehicle body with the four wheels.



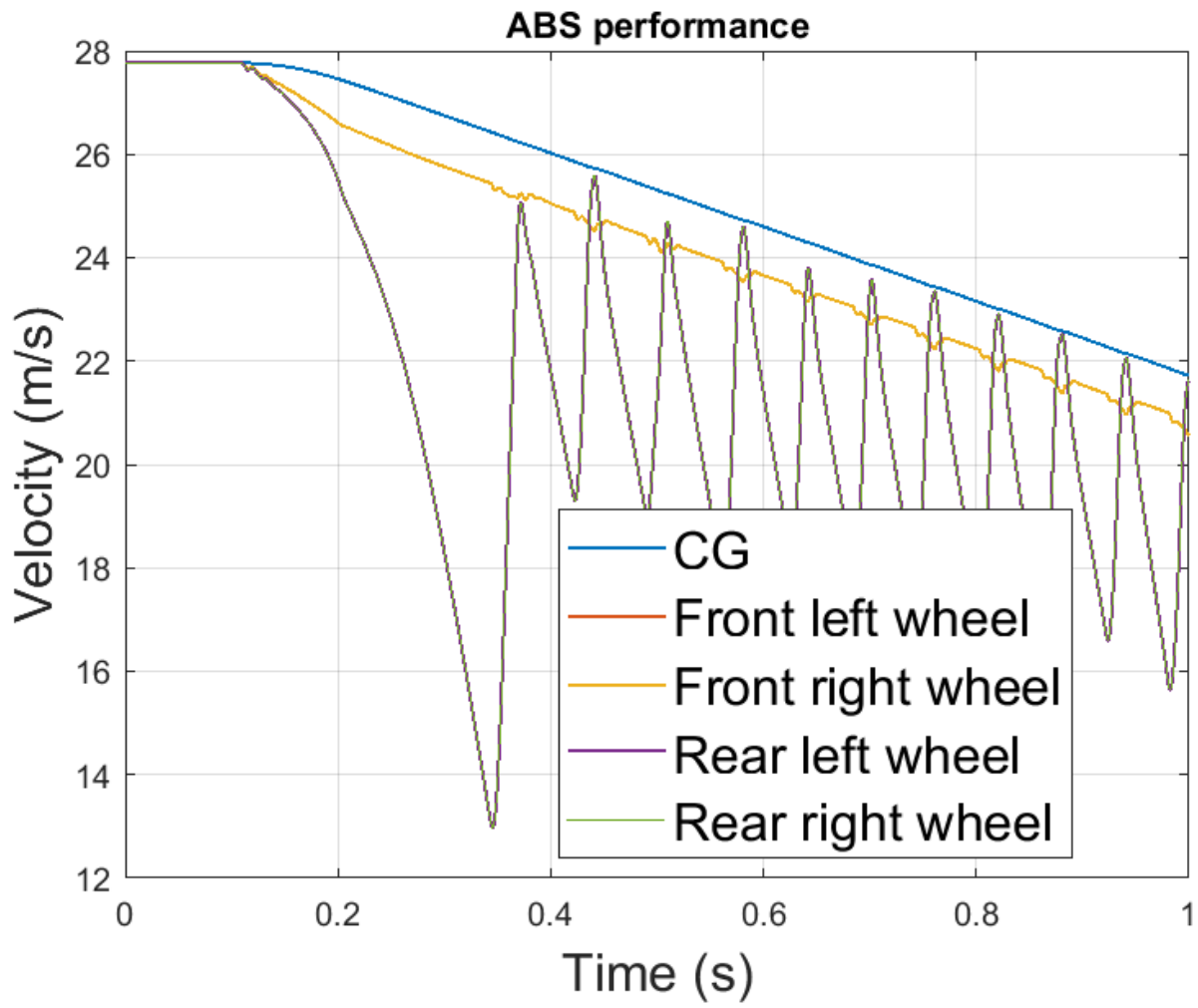
Brakes with ABS

This subsystem shows the modeling of the brakes with the ABS.



Simulation results from Simscape™ logging

Based on the given inputs, the model generates a plot of the velocity profile achieved for the vehicle CG and the wheels.



Abstract Combustion Engine Car

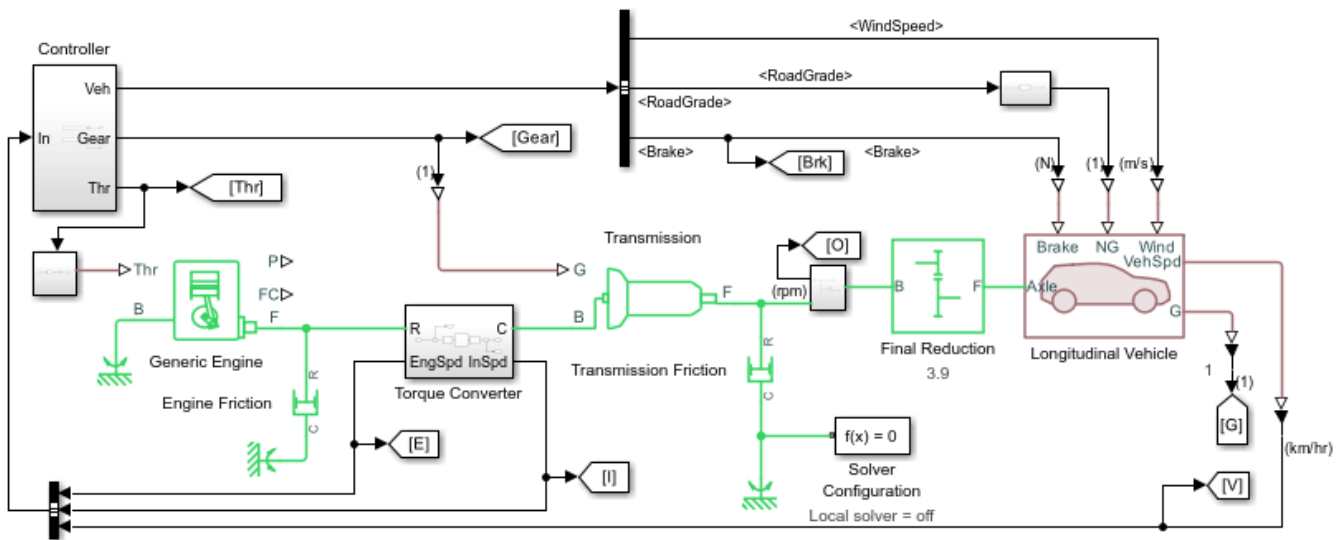
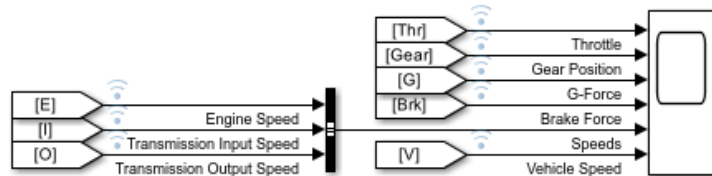
This example shows an abstract passenger car model having an internal combustion engine.

Model

Abstract Combustion Engine Car

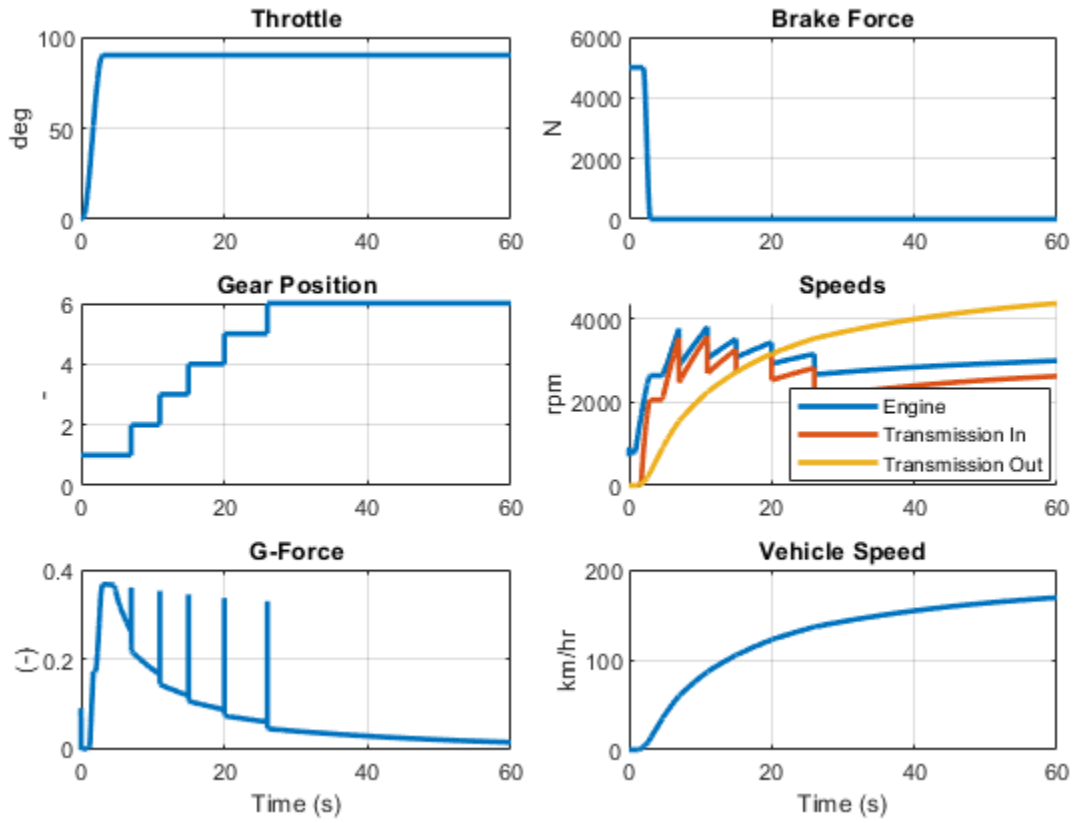
1. Plot input signals (see code)
2. Plot speeds, gear position, and so on (see code)
3. Analyze transmission slip
4. Explore simulation results using Simscape Results Explorer
5. Learn more about this example

Copyright 2022 The MathWorks, Inc.



Simulation Results

The plots below show the result of acceleration. First the vehicle is standing still with engine idle and brake on. The transmission is in the first gear, and the torque converter is absorbing the engine power. Then the vehicle starts accelerating and the gear shifts to the higher positions as the vehicle continues to accelerate.

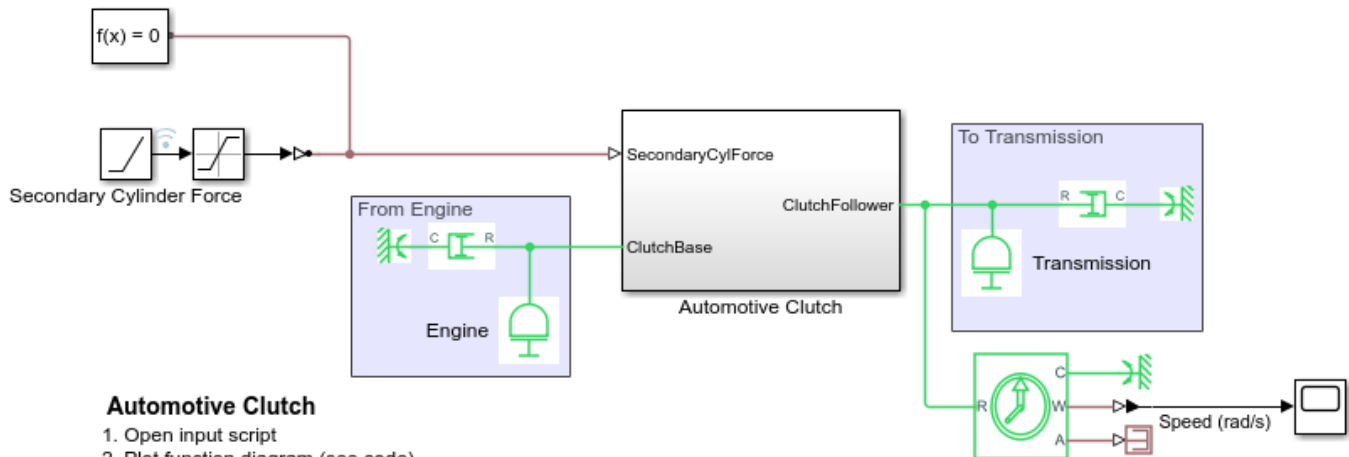


Automotive Clutch

This example shows how to model, parameterize and test an automotive clutch. The model is used to generate the plot of the engine and the transmission system speeds during a declutching scenario.

Model

The following figure shows the model of an automotive clutch in a test harness. The output of the model is a plot of speed vs time of the engine and the transmission systems during the declutching scenario.



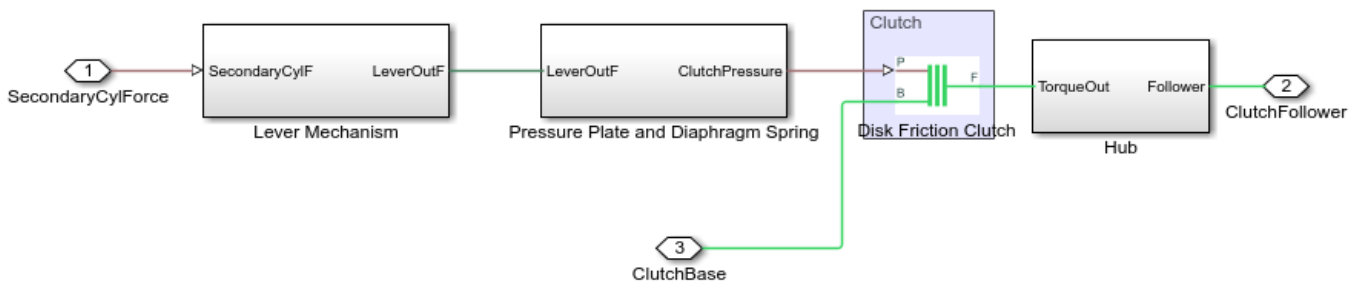
Automotive Clutch

1. Open input script
2. Plot function diagram (see code)
3. Explore simulation results using Simscape Results Explorer
4. Learn more about this example

Copyright 2019-2022 The MathWorks, Inc.

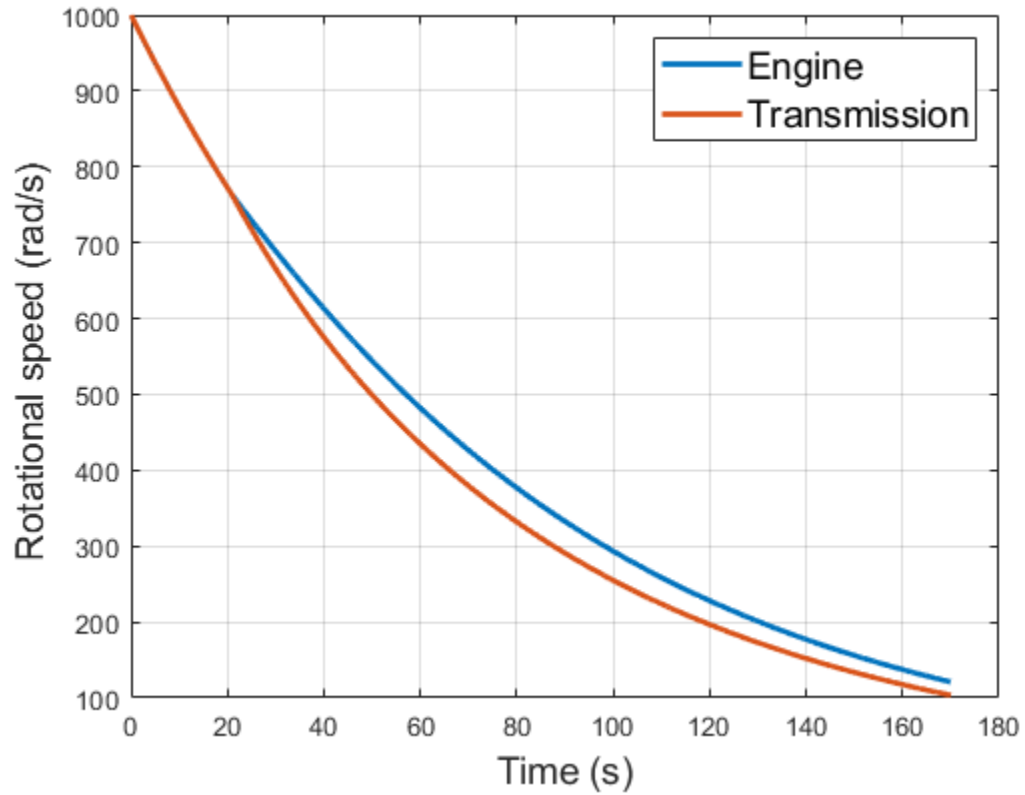
Automotive Clutch Subsystem

This subsystem shows how the clutch is modeled. The clutch consists of a lever mechanism, a diaphragm spring, a pressure plate, a set of clutch plates, and a hub. The lever transfers force to the diaphragm spring when a force is applied on the other end of the lever by the secondary cylinder. The applied force on the diaphragm spring causes the pressure plate to remove the pre-applied force on the clutch plate. The declutching takes place when the force on the clutch plate is removed through the removal of the force on the pressure plate by the diaphragm spring's action. No torque is transferred from the engine to the transmission after the declutching event.



Simulation Results from Simscape™ Logging

This model generates a plot of speed vs time of the engine and the transmission systems during the declutching scenario.

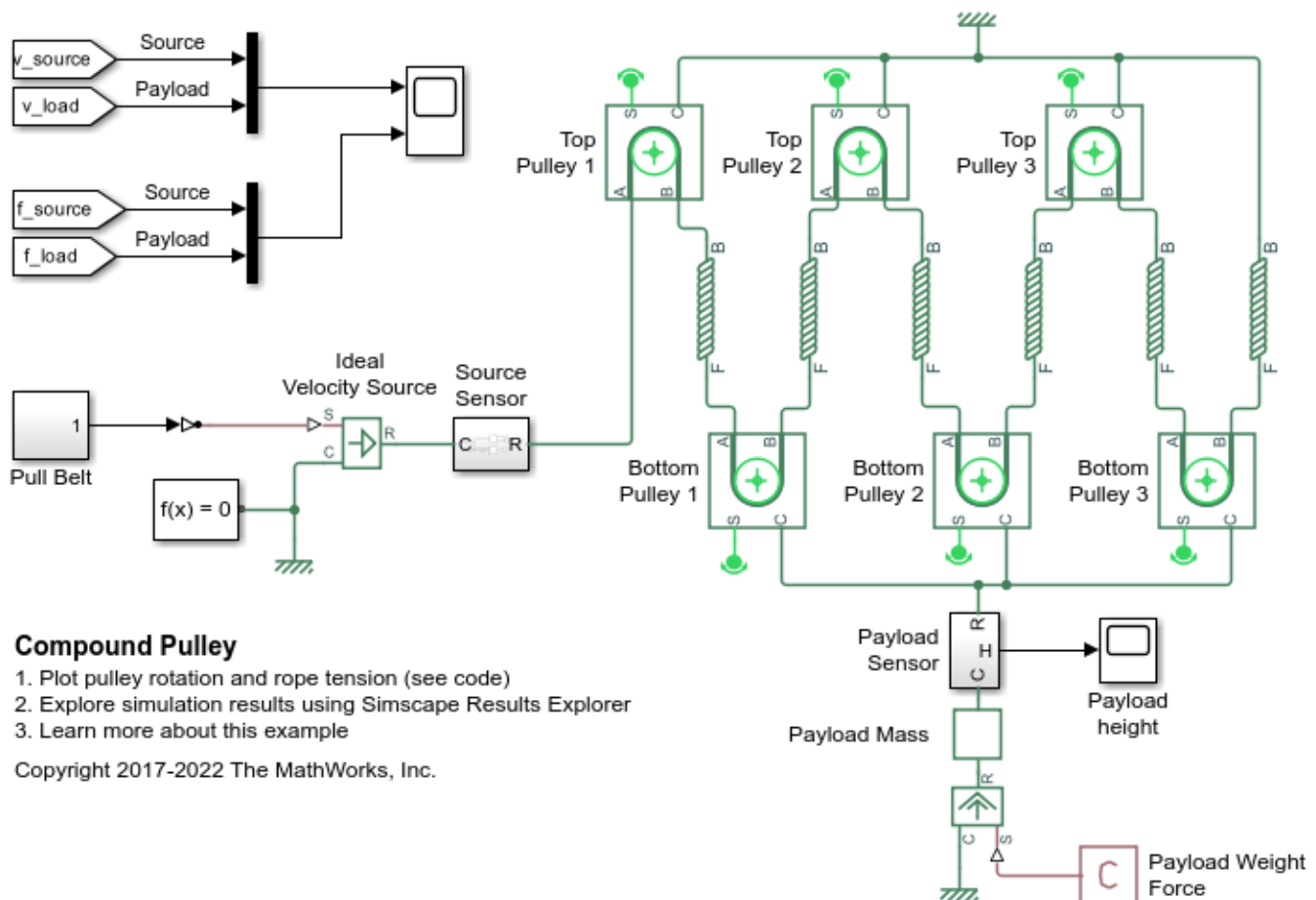


Compound Pulley

This example shows a block and tackle system that is represented by the Belt Pulley and Rope blocks from the Simscape™ Driveline™ library. This compound pulley is rigged as a threefold purchase with two triple blocks. The system has a mechanical advantage of 6.

A payload of 10 metric tons subject to gravity is raised by an Ideal Velocity Source pulling on the rope end. The system starts at rest with the Velocity Source providing a force that is in equilibrium with the load. As the velocity source accelerates and decelerates, the rope tension increases and decreases, respectively.

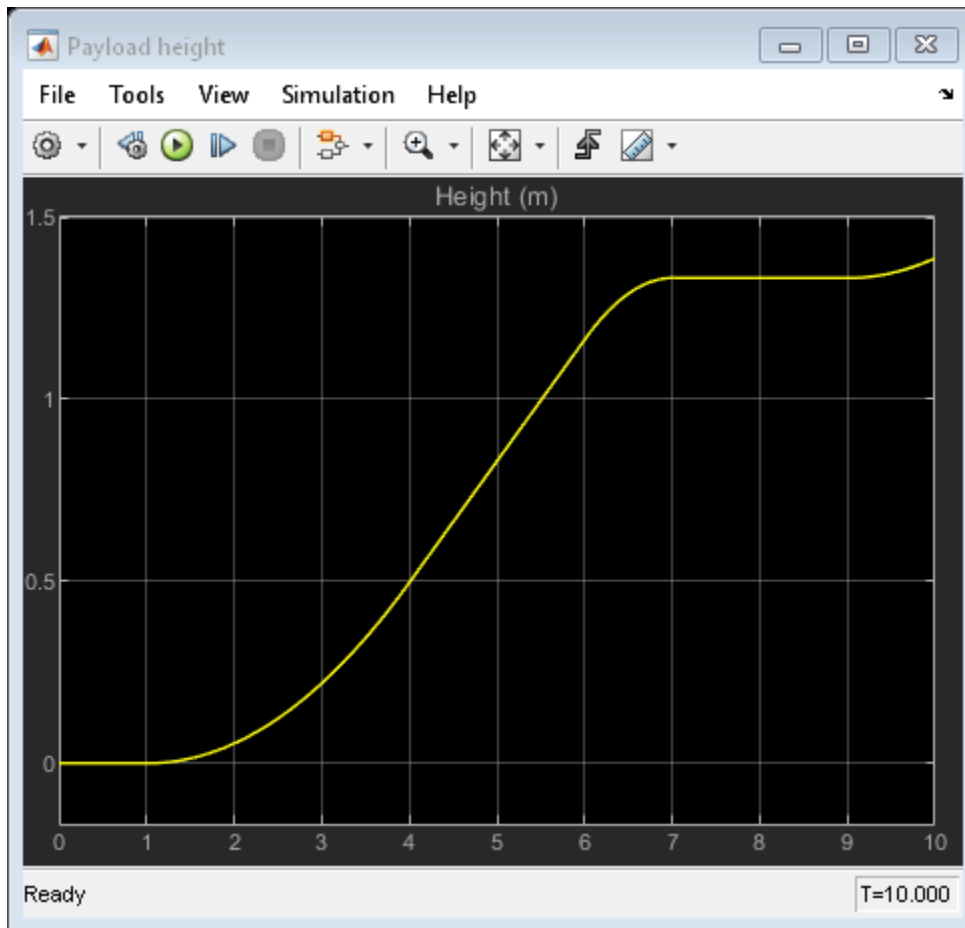
Model

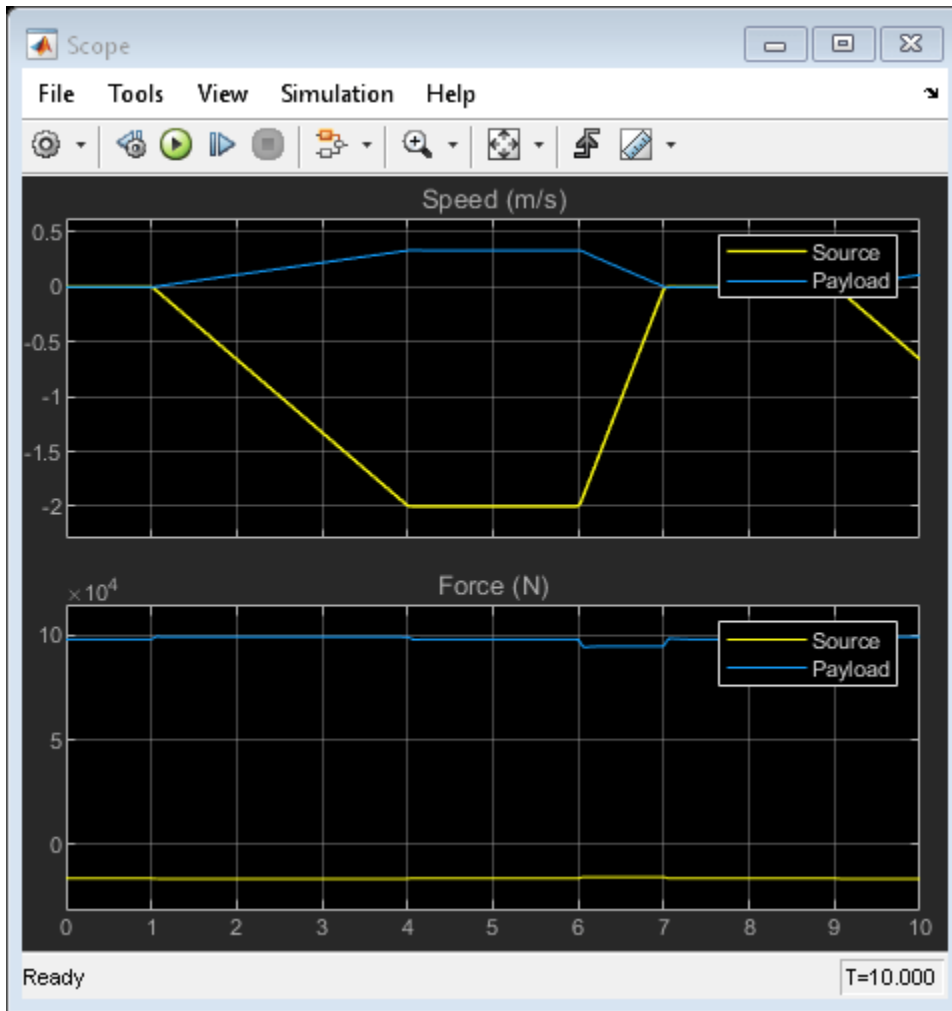


Compound Pulley

1. Plot pulley rotation and rope tension (see code)
2. Explore simulation results using Simscape Results Explorer
3. Learn more about this example

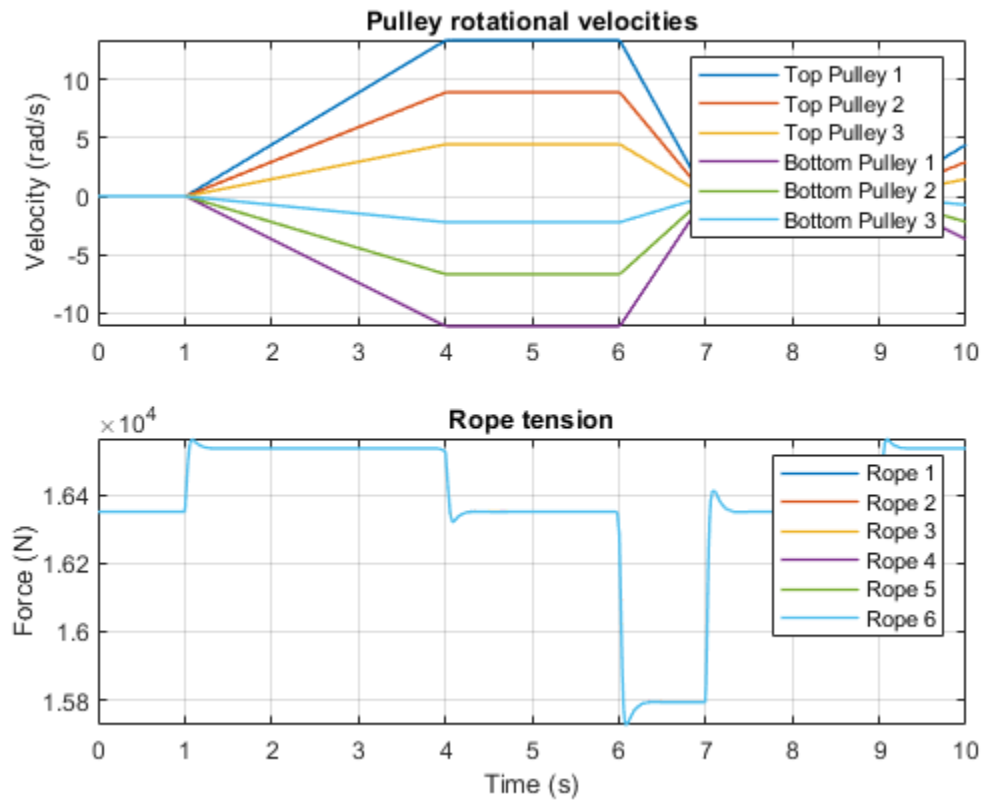
Copyright 2017-2022 The MathWorks, Inc.

Simulation Results from Scopes



Simulation Results from Simscape Logging

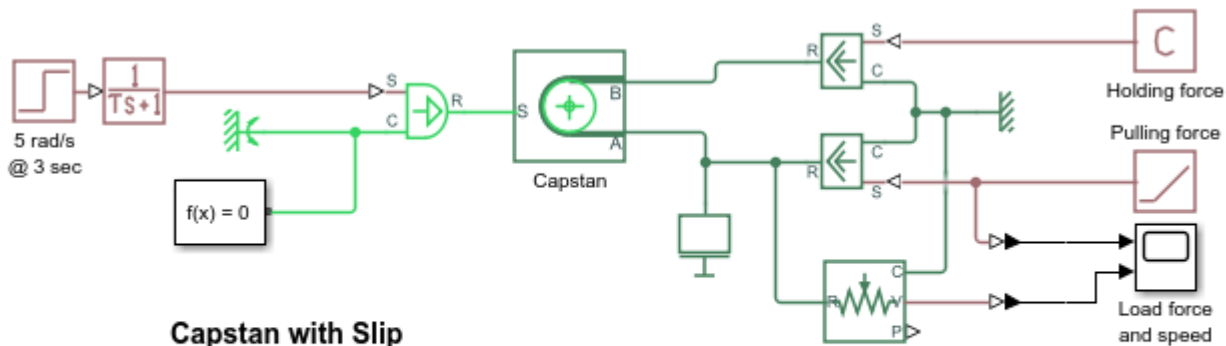
These plots show the pulley rotational velocities and rope segment tensions. Successive pulleys along the driveline rotate at slower velocities. Each rope segment supports a tension load that is one-sixth of the payload force. The rope tension increases as the payload accelerates and decreases as the payload decelerates.



Capstan with Slip

This example shows a capstan represented by a Belt Pulley block from the Simscape™ Driveline™ library. A 100-N tension applied to belt end B holds a variable load at end A in place, even as the load increases. When the load force increases past the friction limit, the belt slips. Increasing the friction by using a higher friction coefficient or wrap angle allows the pulley to hold an even greater load force. This example also shows how suddenly turning the capstan, starting at 3 seconds, causes momentary slip, which is quickly recovered.

Model



Capstan with Slip

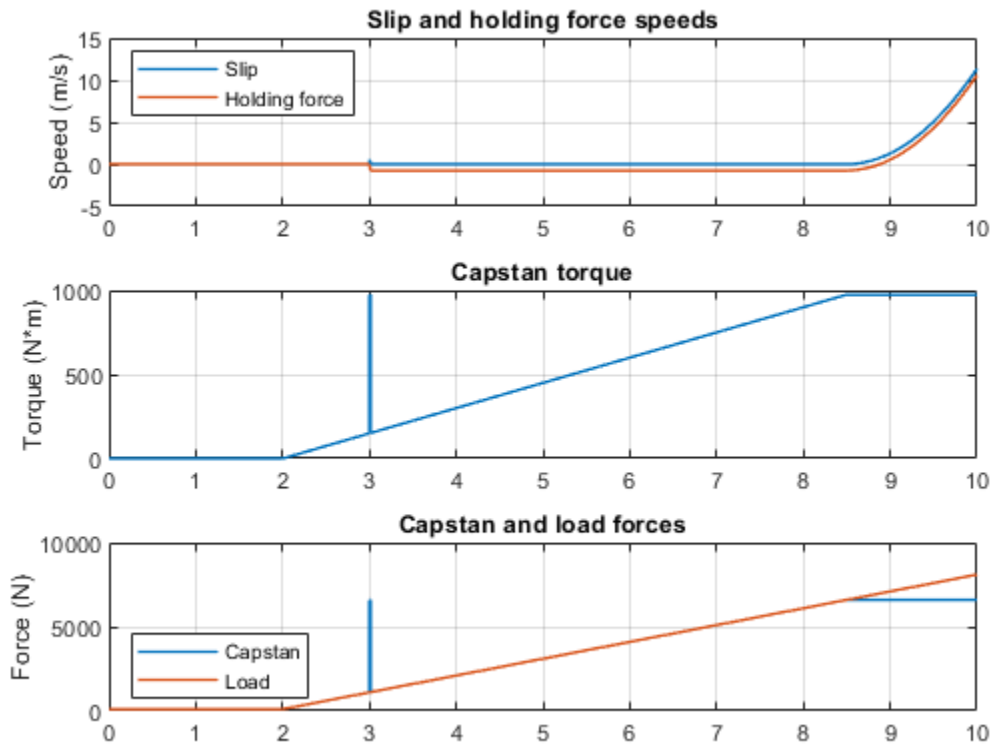
1. Plot speeds, torque, and forces (see code)
2. Explore simulation results using Simscape Results Explorer
3. Learn more about this example

Copyright 2017-2022 The MathWorks, Inc.

Simulation Results from Simscape Logging

The plots show:

1. The relative velocity between the capstan and the belt along with the holding force speed.
2. The capstan torque required to produce a change in the rotational speed and to resist the holding and load forces.
3. How the load force eventually exceeds the capstan force limit and causes slip.

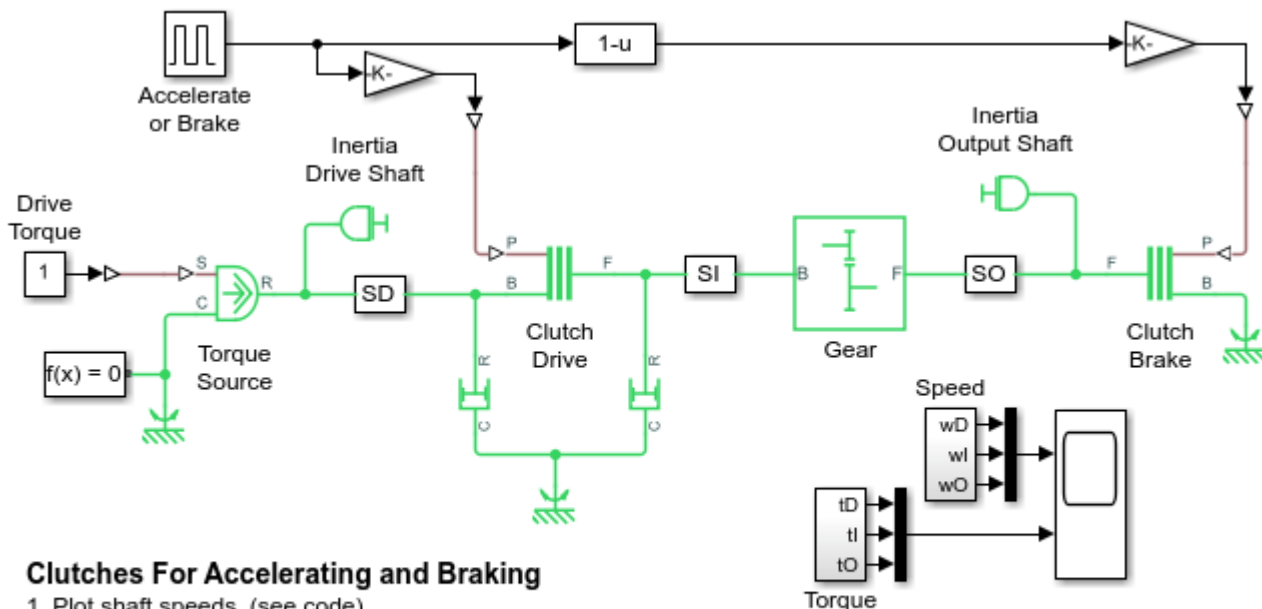


Clutches For Accelerating and Braking

This example the use of clutches to accelerate and decelerate a pair of shafts connected by a gearbox. The gear connects the output and intermediate shafts. The inertia for both of these shafts is included in the Inertia Output Shaft component. This pair of shafts is connected via one clutch to a drive shaft which is driven by a torque. These shafts are also connected via a second clutch to a stationary point.

Initially, the drive clutch is engaged and all three shafts are accelerated by the torque source. The states of the drive clutch and brake clutch are then changed simultaneously. This disconnects the drive shaft from the geared shafts and engages the brake clutch, which acts as a brake and dissipates the energy of the geared shafts. The drive shaft accelerates upon the change of clutch states, for less inertia is connected to the torque source.

Model



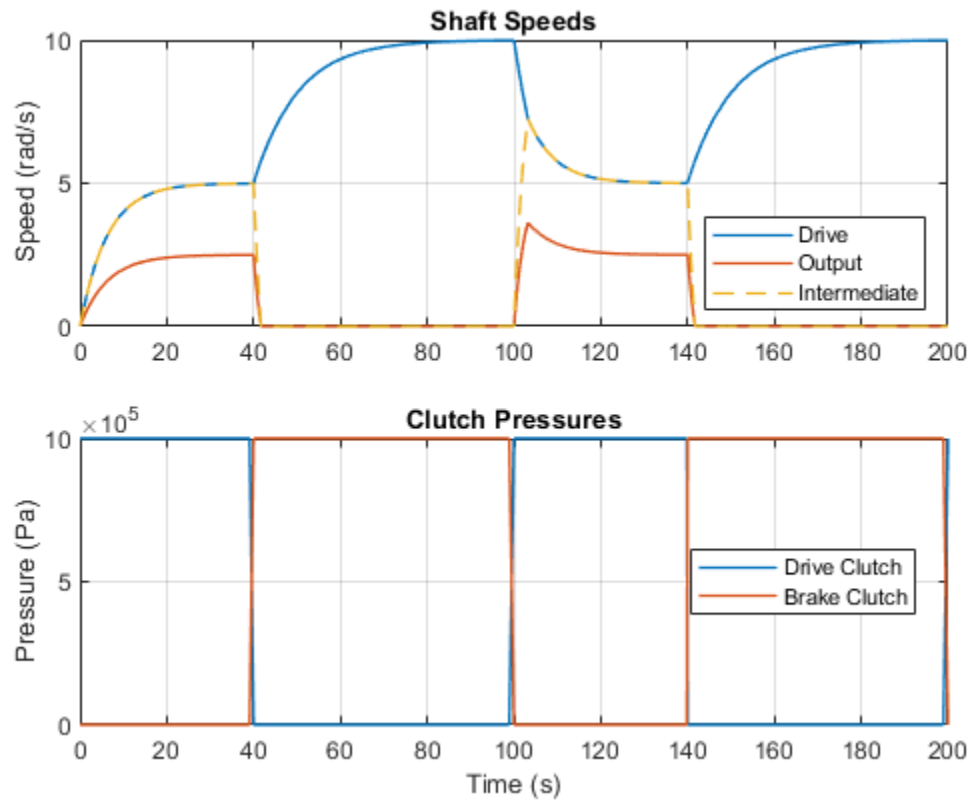
Clutches For Accelerating and Braking

1. Plot shaft speeds (see code)
2. Explore simulation results using Simscape Results Explorer
3. Learn more about this example

Copyright 2003-2022 The MathWorks, Inc.

Simulation Results from Simscape Logging

The plot below shows the speeds of a drive shaft, intermediate shaft, and an output shaft. The intermediate and output shafts are connected by a gearbox. Two clutches control the rotation of these shafts. The drive clutch connects them to the drive shaft which is driven by a constant torque source. The brake clutch connects them to a stationary point. As these clutches engage and disengage, the speed of all shafts change.



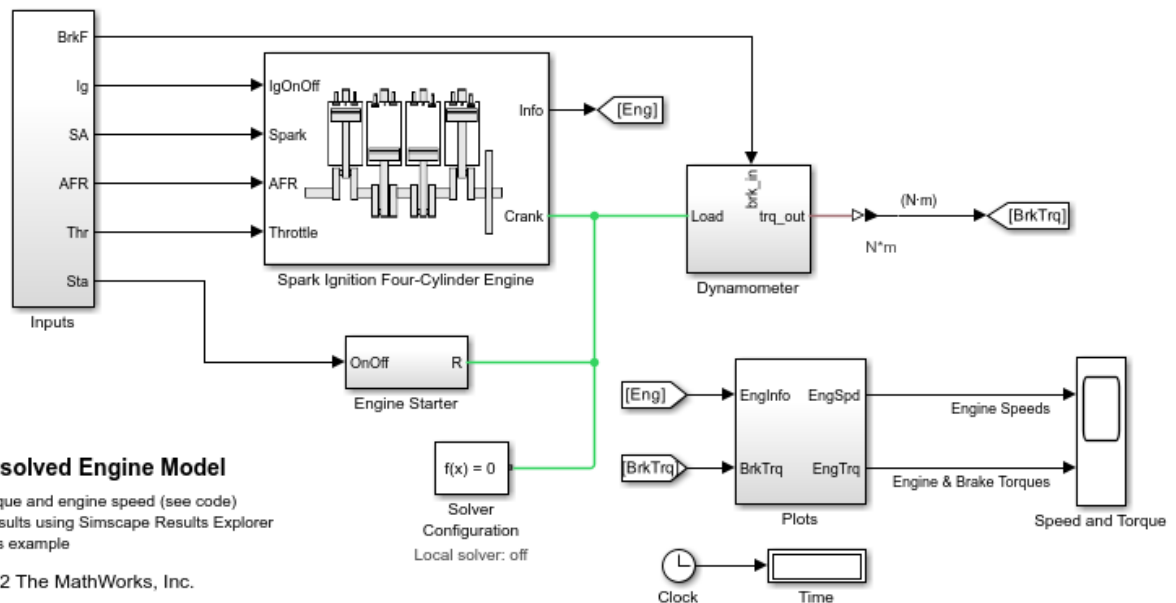
Crank-Angle-Resolved Engine Model

This example shows a 2.0-liter, four-cylinder, naturally aspirated, spark-ignited, four-stroke engine which computes crank-angle-resolved instantaneous torque.

To start the engine from the rest, first a starter motor starts rotating the crank wheel, followed by fuel injection and spark ignition to start firing. When combustion produces more torque at the crank than the starter motor torque, one-way clutch (also called overrunning clutch) in the starter prevents the transmission of torque from crank to starter. Once the crank gets enough rotational momentum from successive combustions, starter motor is turned off and engine continues to operate by injection and spark.

The engine torque can be controlled by varying throttle, air-fuel ratio, and spark timing.

Model

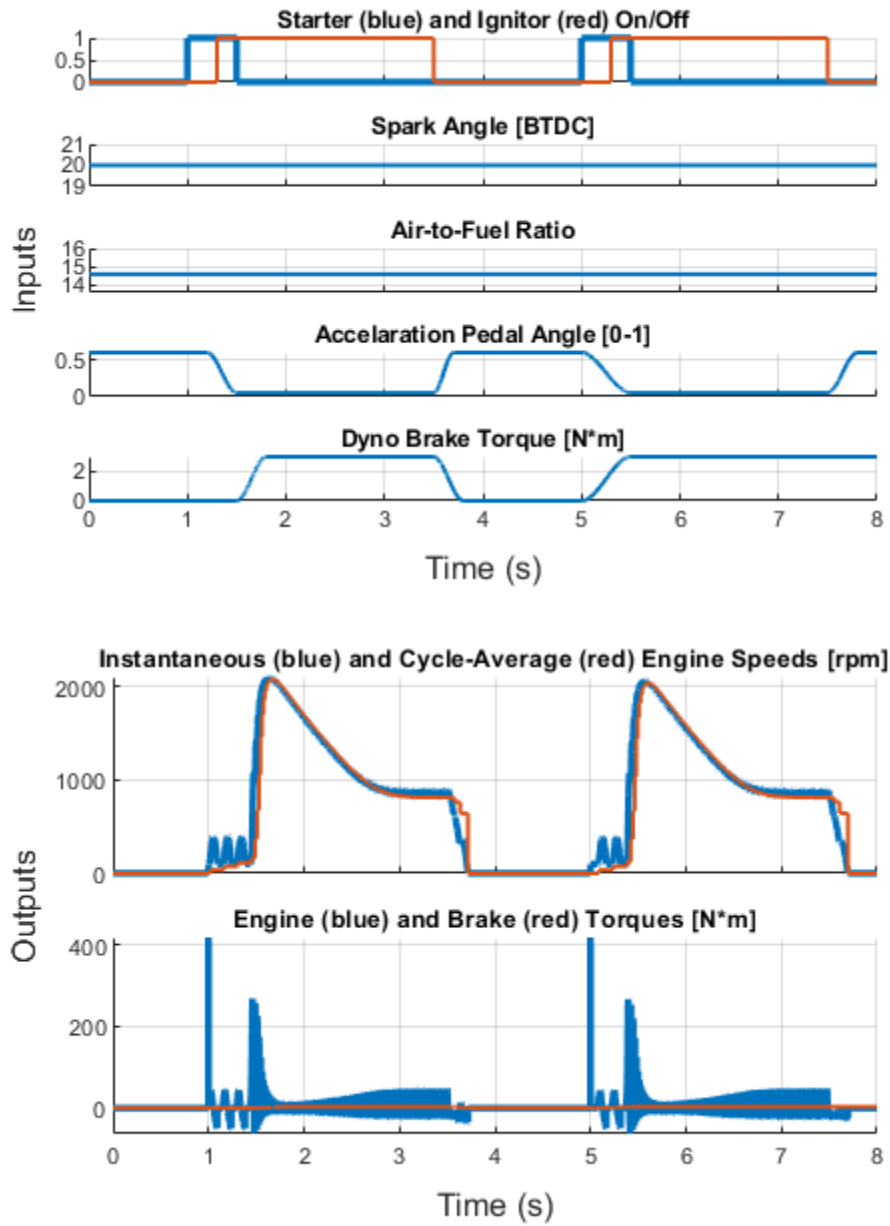


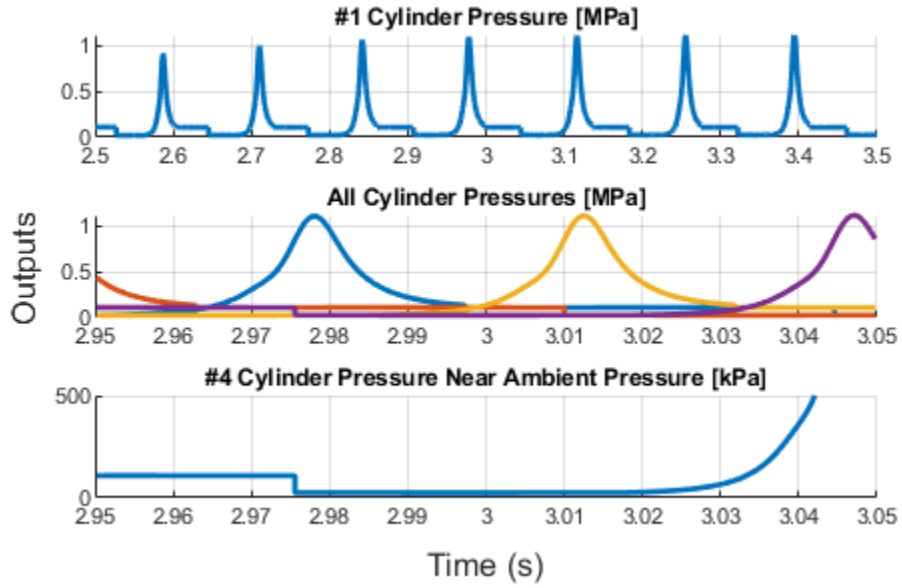
Crank-Angle-Resolved Engine Model

1. Plot inputs, brake torque and engine speed (see code)
2. Explore simulation results using Simscape Results Explorer
3. Learn more about this example

Copyright 2017-2022 The MathWorks, Inc.

Simulation Results



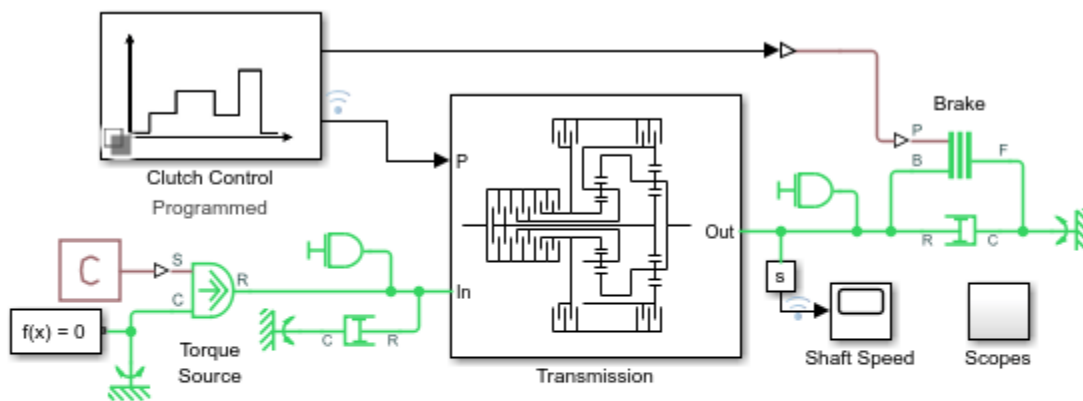


CR-CR Four-Speed Transmission

This example shows a four-speed transmission with two planetary gears and five clutches. The test sequence steps through the four forward gear ratios, switches to neutral, and then applies a brake clutch to the output shaft of the transmission.

To manually select the gear for the transmission, configure the Clutch Control subsystem to use the Manual variant using the hyperlinks. The Manual control subsystem has blocks from the Dashboard library that let you select the gear and apply the brake by clicking on them with the mouse.

Model

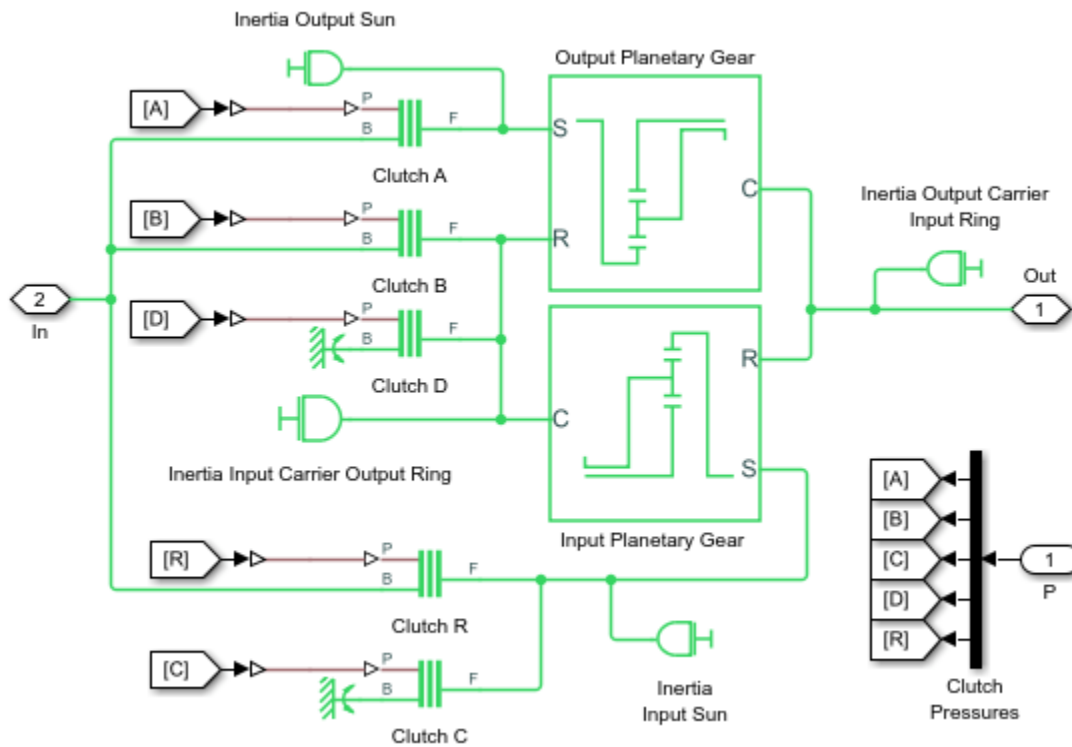


CR-CR Four-Speed Transmission

1. Plot speeds of shafts (see code)
2. Configure Clutch Control: Programmed, Manual
3. Explore simulation results using Simscape Results Explorer
4. Learn more about this example

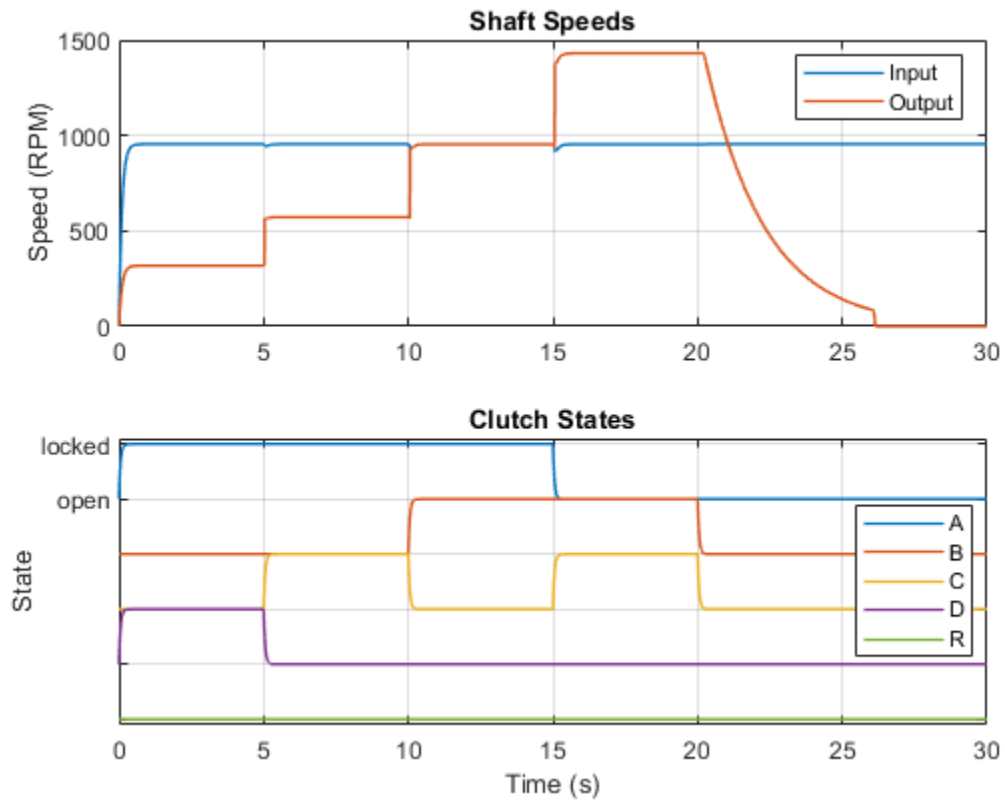
Copyright 2016-2022 The MathWorks, Inc.

Transmission Subsystem



Simulation Results from Simscape Logging

The plot below shows the input and output shaft speeds of the transmission. The clutch states are also plotted (locked or open), indicating the selected gear of the transmission.

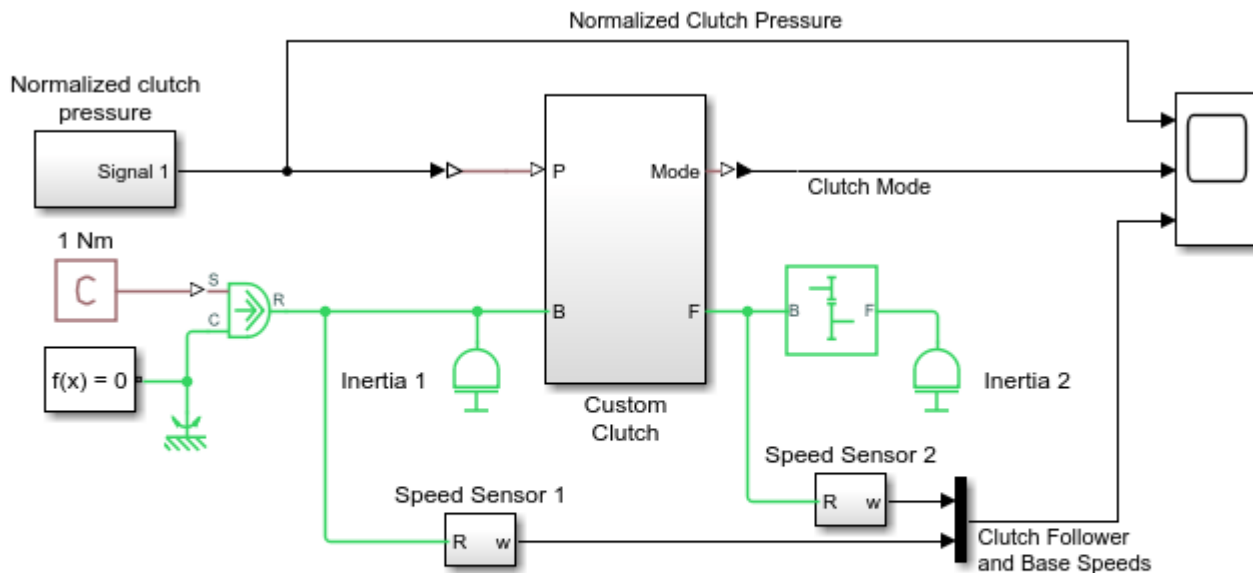


Custom Clutch

This example shows two inertias coupled by a simple gear controlled by a clutch. Initially, the clutch pressure is zero, and the inertias have zero velocity. A constant torque is applied to Inertia 1. Once the clutch pressure starts increasing from zero at 2 seconds, Inertia 2 starts spinning, then locks with Inertia 1. From 6 to 7 seconds, the clutch pressure is ramped down to zero, and Inertia 2 starts to spin freely once the clutch unlocks. Because there is no frictional loss, it keeps spinning.

The Custom Clutch block is built using the Fundamental Friction Clutch library block. The example illustrates how you can build your own custom clutch models from this fundamental block. The physical signal output M corresponds to the clutch mode, and equals -1 for negative slip, 0 if locked, and +1 for positive slip.

Model

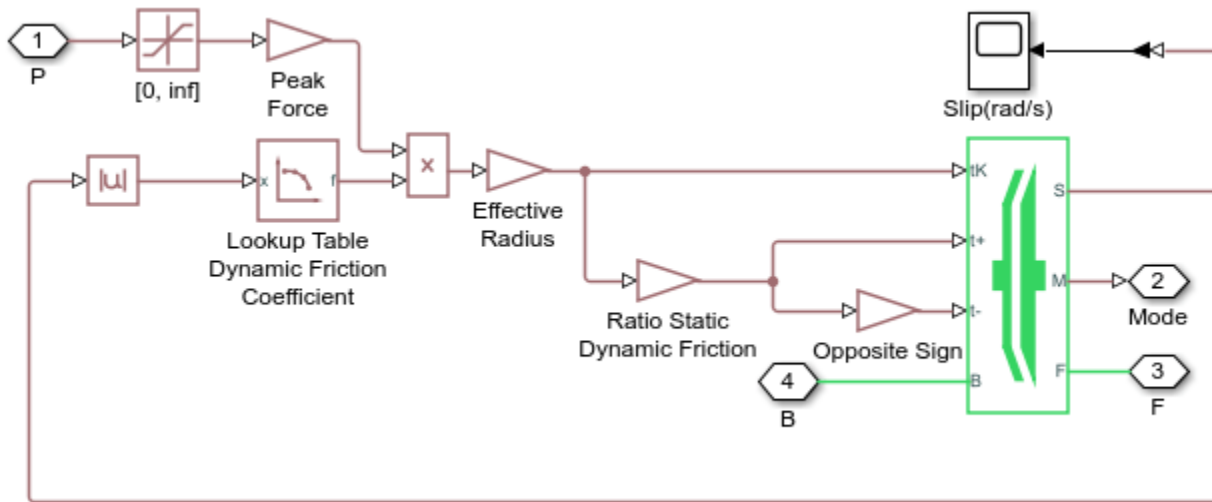


Custom Clutch

1. Plot speeds of clutch shafts (see code)
2. Explore simulation results using Simscape Results Explorer
3. Learn more about this example

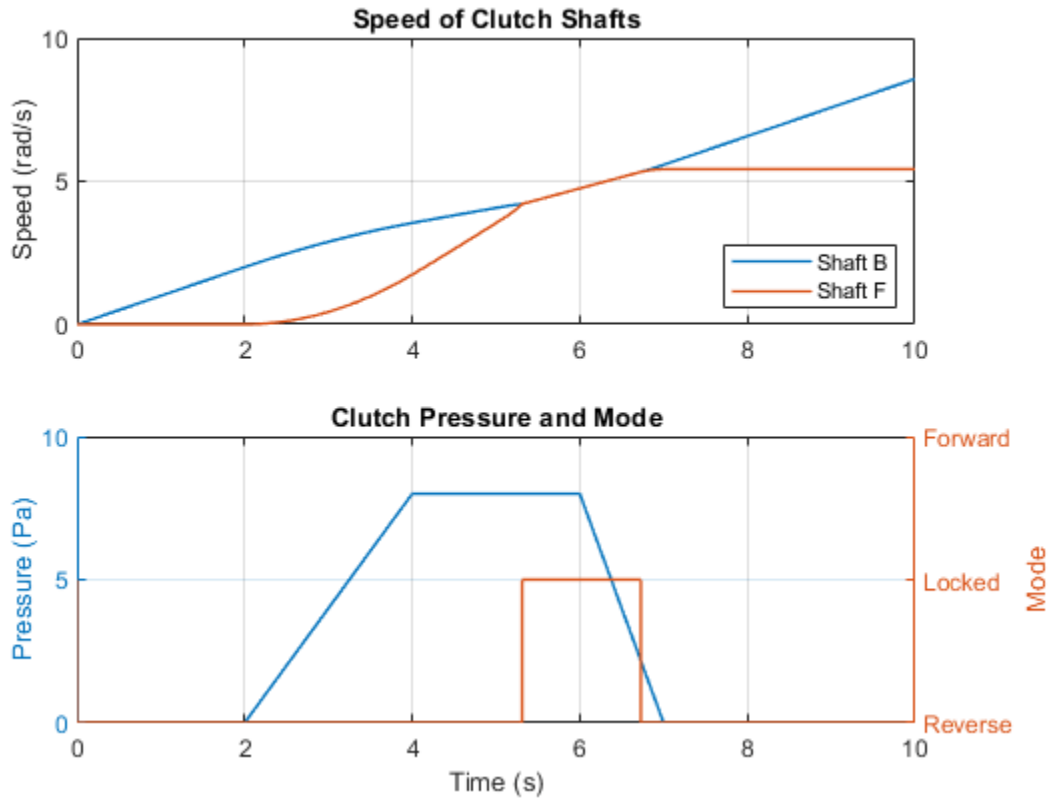
Copyright 2003-2022 The MathWorks, Inc.

Custom Clutch Subsystem



Simulation Results from Simscape Logging

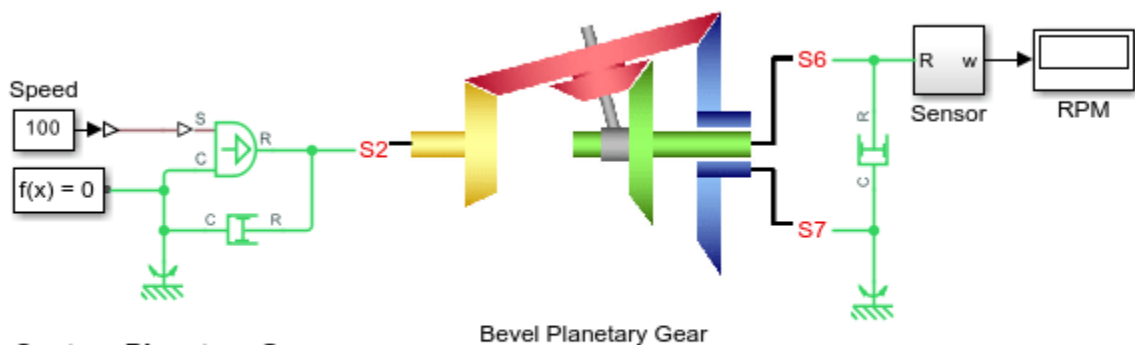
The plot below shows the speeds of the two shafts connected via a clutch. When the clutch is engaged, the shafts spin at the same speed. The pressure applied to the clutch and the mode of the clutch (locked or slipping in forward or reverse) is shown.



Custom Planetary Gear

This example illustrates how you can build your own custom planetary gear components from the Simscape™ Driveline™ Planetary Subcomponents library. The example is based on Example 10.7 published in K.J. Waldron, G.L. Kinzel "Kinematics, Dynamics, and Design of Machinery", 1999. It consists of a coupled planetary train built from simple and compound planetary trains with a common carrier and bevel gears. The simple train fundamental train ratio is $-N_4/N_2 * N_7/N_4 = -N_7/N_2$. It connects shafts S2 and S7. The compound planetary train fundamental ratio is $-N_4/N_2 * N_6/N_5$ and it connects shafts S2 and S6. When the input shaft of the train is rotated at 100 rpm, the model shows the output shaft rotates at 1.4456 rpm as given in the reference.

Model



Custom Planetary Gear

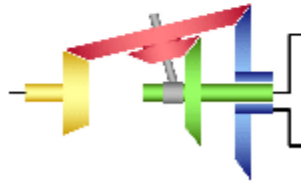
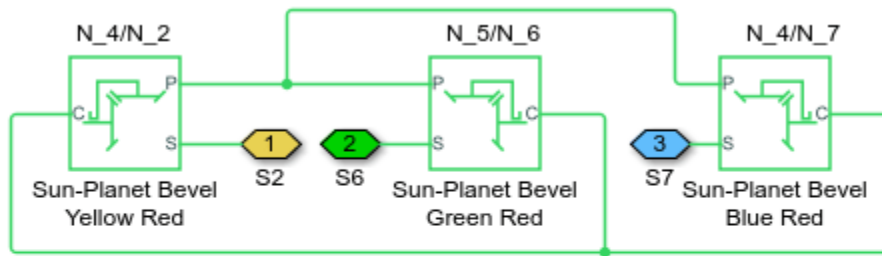
1. Explore simulation results using Simscape Results Explorer
2. Learn more about this example

Copyright 2008-2022 The MathWorks, Inc.

Bevel Planetary Gear Subsystem

Construction notes:

1. Port colors correspond to the image
2. The three bevel gear meshing points are represented by the Sun-Planet Bevel gears.
3. The red gear is a compound planetary gear with two sets of beveled teeth. It is modeled by connecting all P ports together
4. The single carrier for the planetary gear is modeled by connecting all C ports together.
5. Block orientations have been set to reflect the Assembly orientation parameter choice. For example, the blue-red gear meshing has the sun gear to the right of the planet gear, whereas the yellow-red gear meshing has the sun gear to the left of the planet gear.



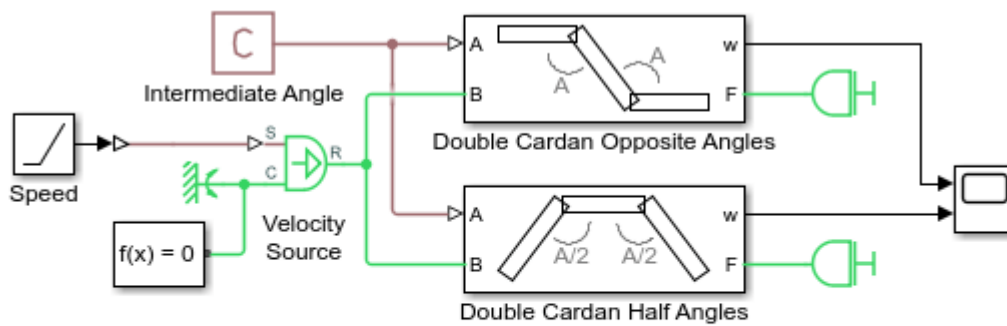
Double Cardan Joint

This example shows two methods to create a constant rotational velocity output using universal joints. In the first method, the angle of the universal joints is exactly opposite. The output shaft axis is parallel to the input shaft axis, but offset by some distance.

In the second method, the axes of the input and output shafts are offset by a specified angle. The angle of each universal joint is half of the angular offset of the input and output axes.

Note that the output rotational velocity can vary from the input due to compliance in the joints. Stiffer compliance can result in more accurate tracking, but higher internal torques and vibrations.

Model

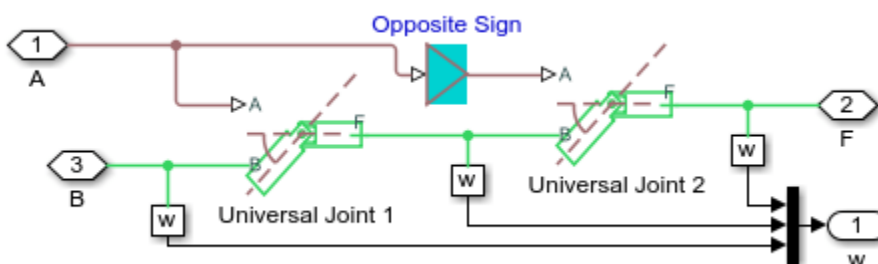


Double Cardan Joint

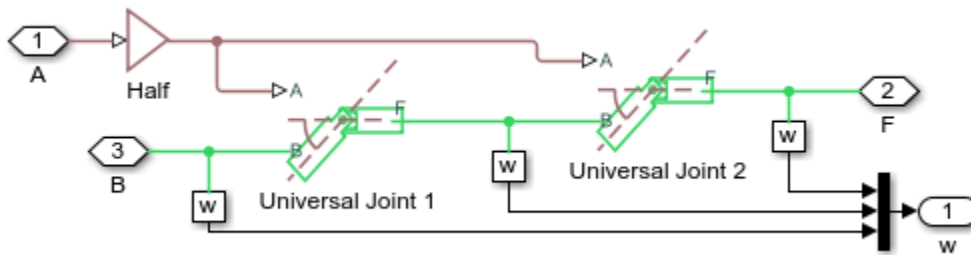
1. Plot speeds of shafts (see code)
2. Explore simulation results using Simscape Results Explorer
3. Learn more about this example

Copyright 2013-2022 The MathWorks, Inc.

Double Cardan Opposite Angles Subsystem

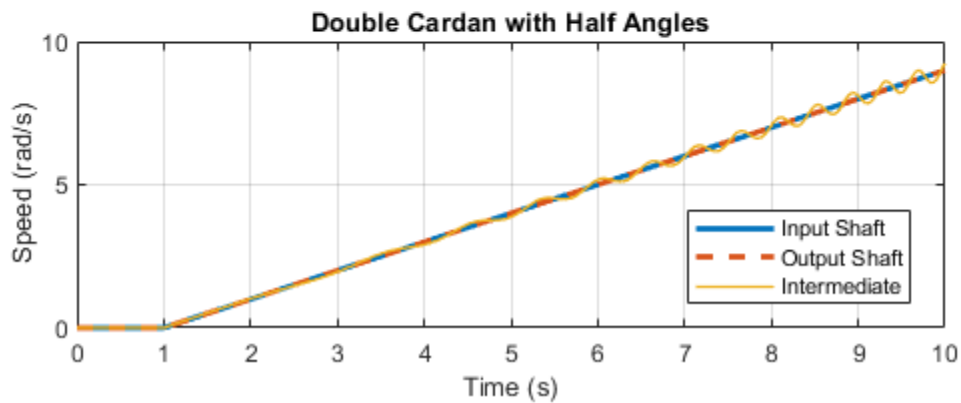
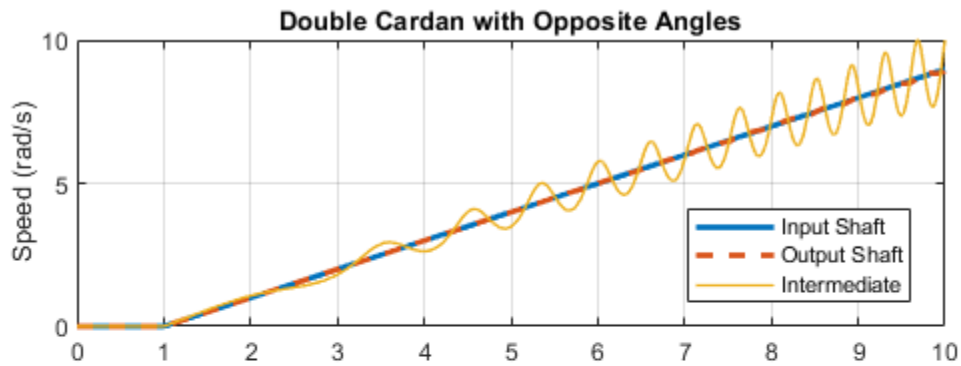


Double Cardan Half Angles Subsystem



Simulation Results from Simscape Logging

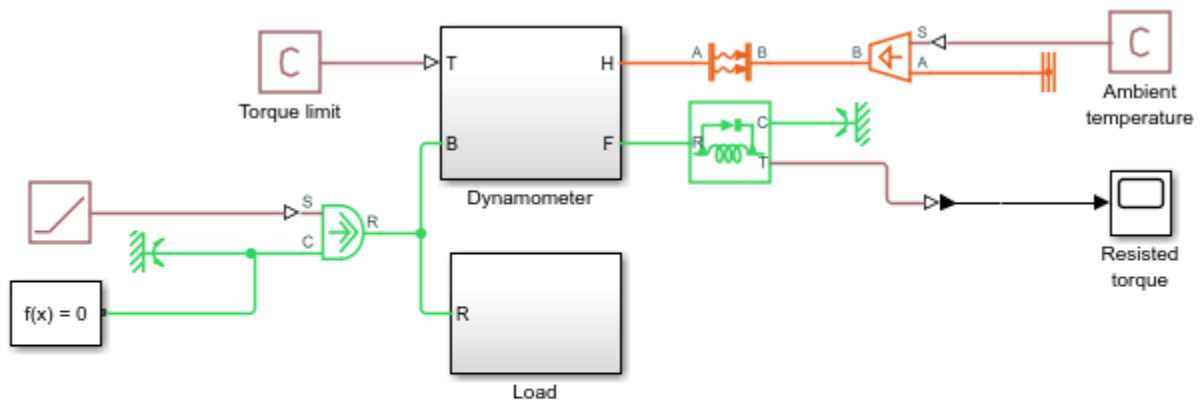
The plots below show the shaft speeds in double cardan joints. In each case, the output and input shaft speeds are identical. The intermediate shaft speed varies with the rotational angle of the shaft.



Dynamometer

This example shows a dynamometer resisting a load from a prime mover. This prime mover is modeled as a torque source and could be an internal combustion engine, electric drive, or a hydraulic motor. The dynamometer absorbs input torque up to its limit, and allows rotation of connected loads for torques beyond the limit. The dynamometer itself is modeled using the Fundamental Friction Clutch with static and dynamic limits set to the torque limit.

Model



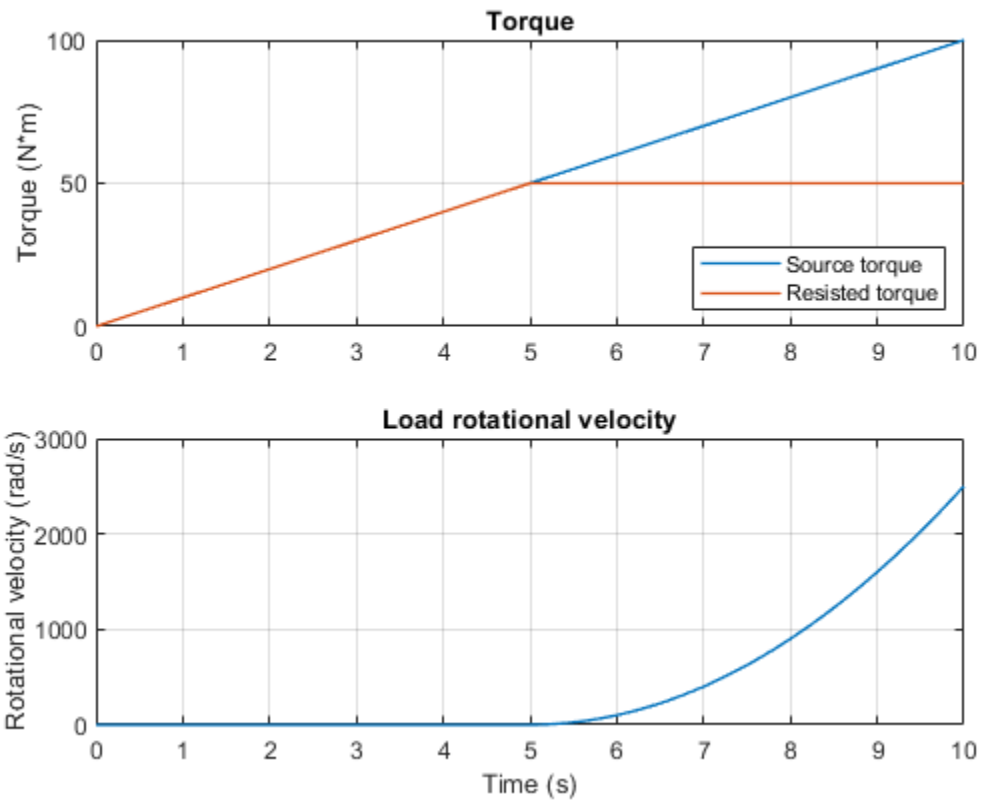
Dynamometer

1. Plot input and absorbed torques and load speed (see code)
2. Explore simulation results using Simscape Results Explorer
3. Learn more about this example

Copyright 2016-2022 The MathWorks, Inc.

Simulation Results from Simscape Logging

These plots show the applied and resistive torques to the dynamometer. Note that the dynamometer completely absorbs torque up to its torque limit. After this, it can continue absorbing the amount up to the limit, but additional torque from the prime mover will cause the attached load to rotate.

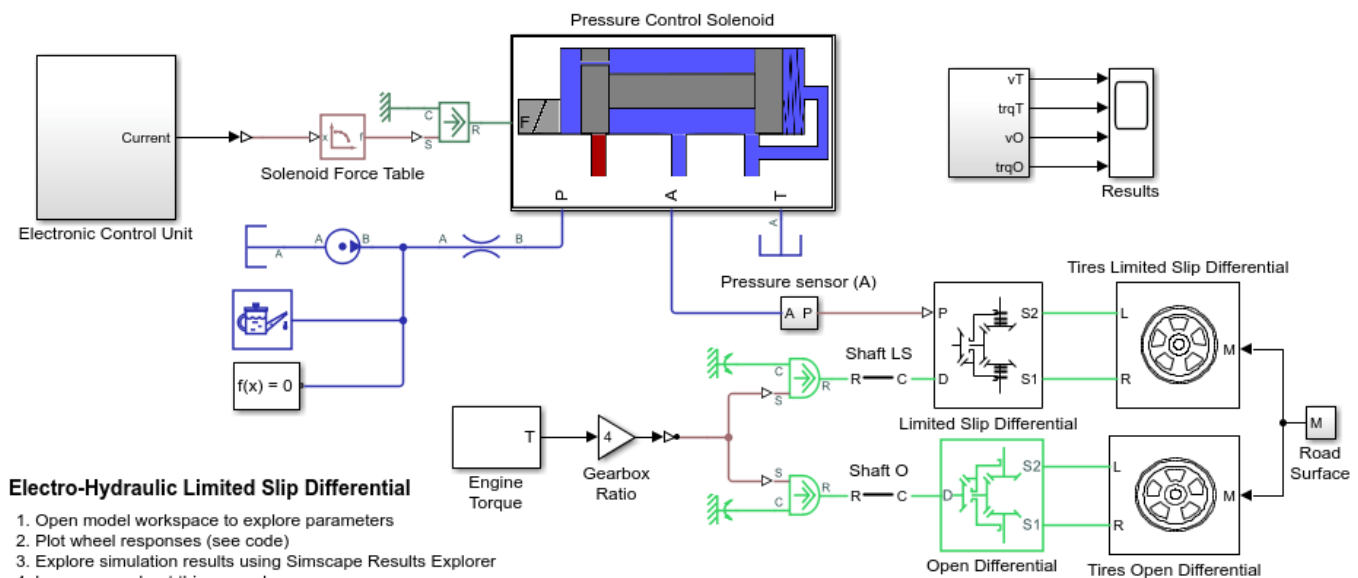


Electro-Hydraulic Limited Slip Differential

This example shows a simple way of modeling an electro-hydraulic limited slip differential system. The model shows the velocity and the torque profile responses achieved for the left and the right wheels.

Model

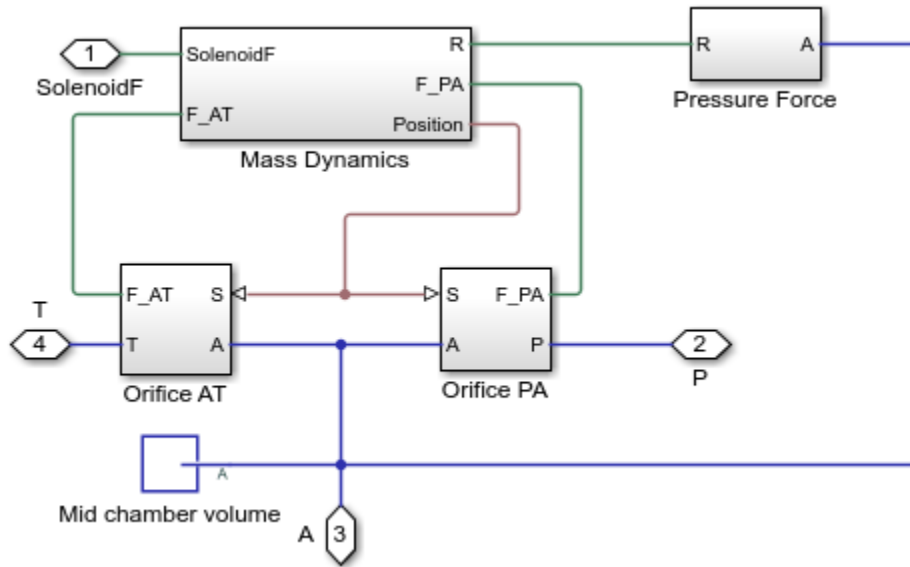
The following figure shows the model of an electro-hydraulic limited slip differential system with a test harness. The electronic control unit sends a command to the pressure control solenoid valve based on the relative slip between the right and the left wheels. The pressure control solenoid valve applies a pressure on the actuating mechanism which in turn applies the limited slip control.



Copyright 2020-22 The MathWorks, Inc.

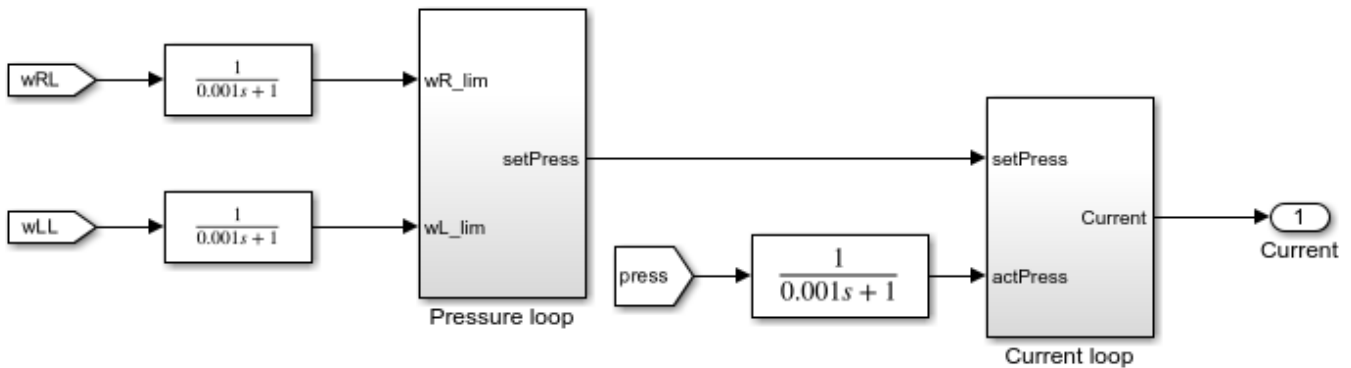
Pressure Control Solenoid Subsystem

This subsystem shows the modeling of the pressure control solenoid valve.



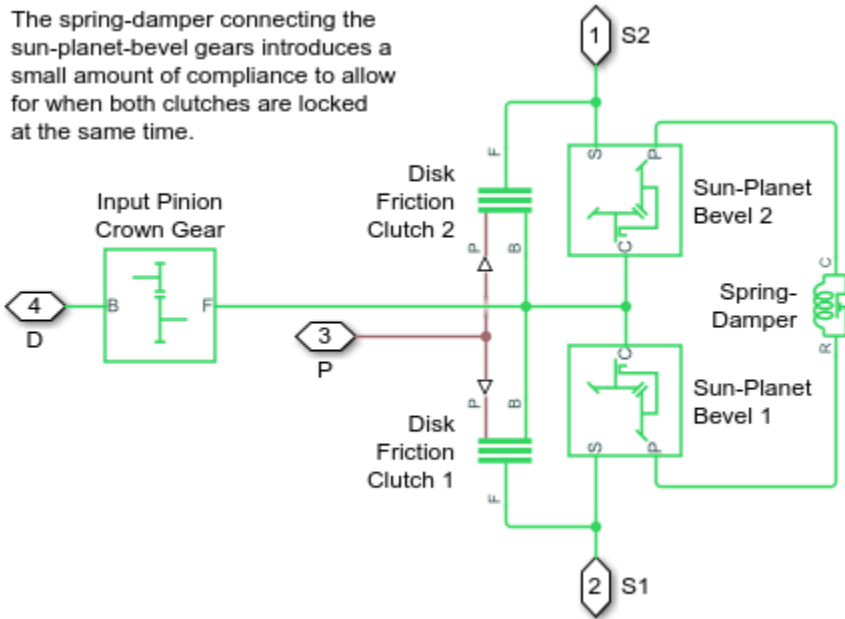
Electronic Control Unit Subsystem

This subsystem shows the modeling of the control of the pressure control solenoid.



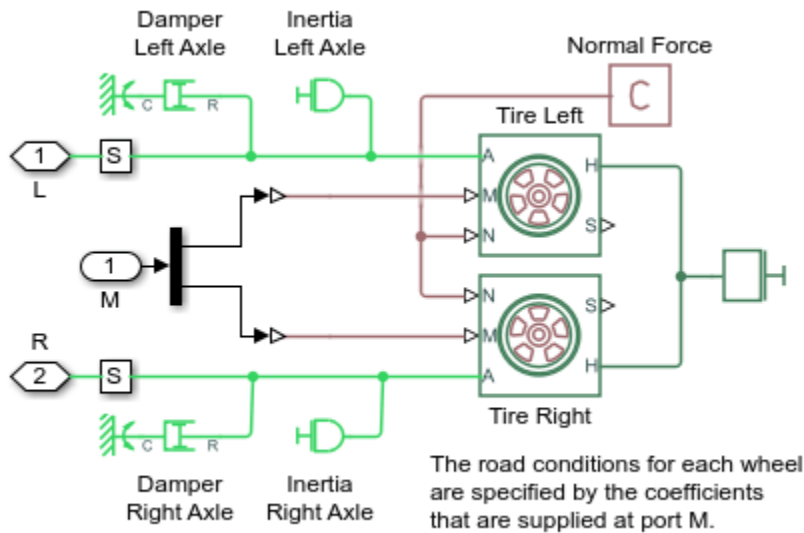
Limited Slip Differential Subsystem

This subsystem shows the modeling of the limited slip differential.



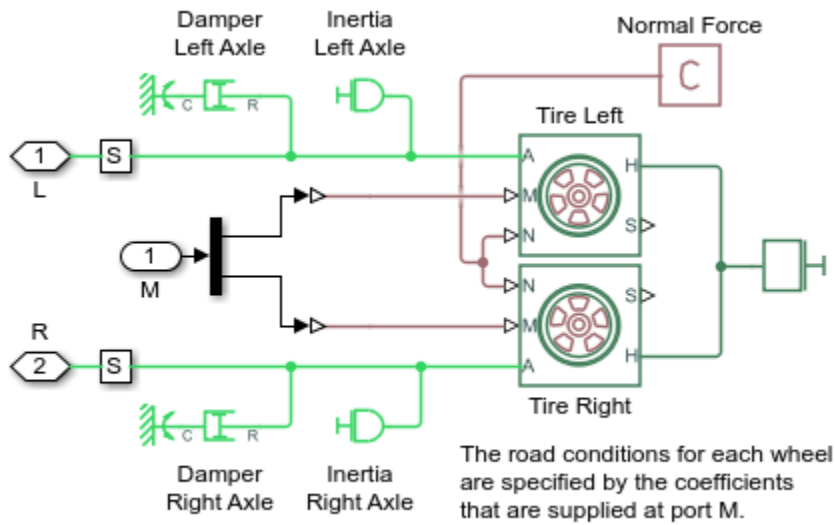
Tires Limited Slip Differential Subsystem

This subsystem shows the modeling of the tires for limited slip differential.



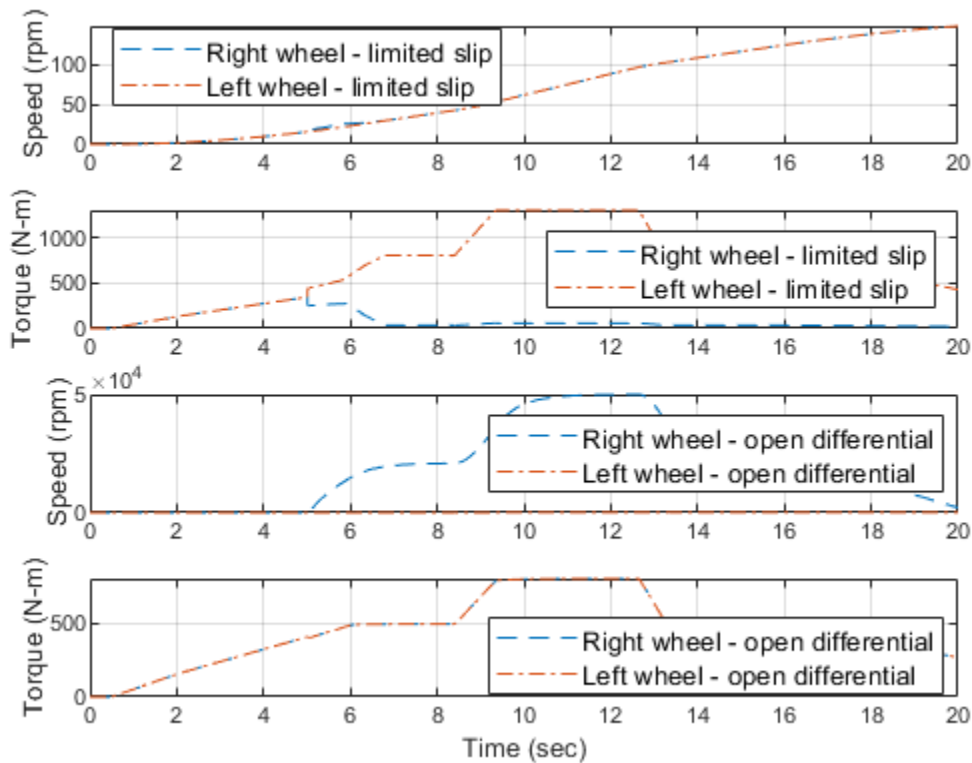
Tires Open Differential Subsystem

This subsystem shows the modeling of the tires for open differential.



Simulation results from Simscape™ logging

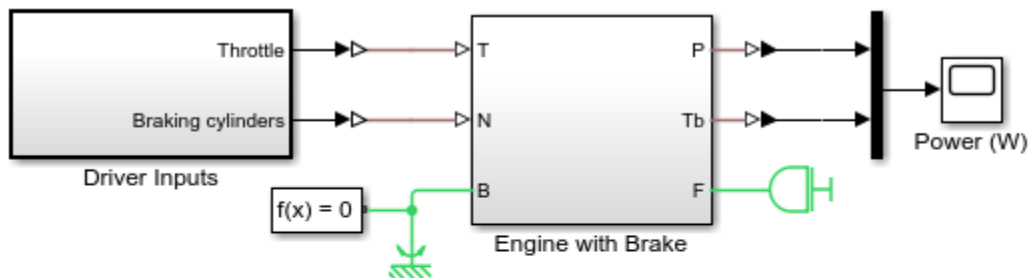
Based on the given inputs, the model generates a plot of the velocity and the torque profile achieved for the left and the right wheels.



Engine Braking

This example shows a diesel engine equipped with a Jake Brake (R) or compression release engine brake. This type of brake is activated by the driver to slow the vehicle without relying on friction braking mechanisms. Air is released from a number of cylinders at the end of their compression strokes. In this example, the driver can dynamically select to activate braking on zero to eight cylinders. The braking torque for each cylinder is set using a lookup table with engine speed as the input.

Model

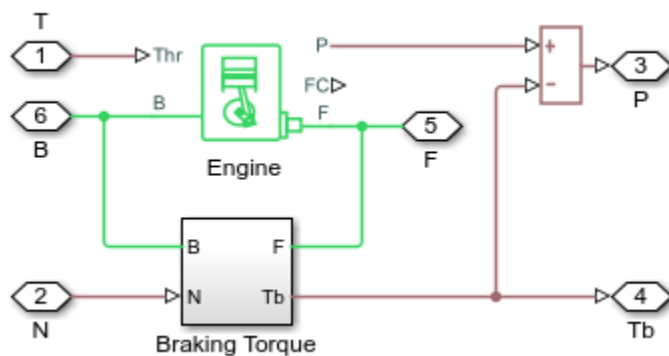


Engine Braking

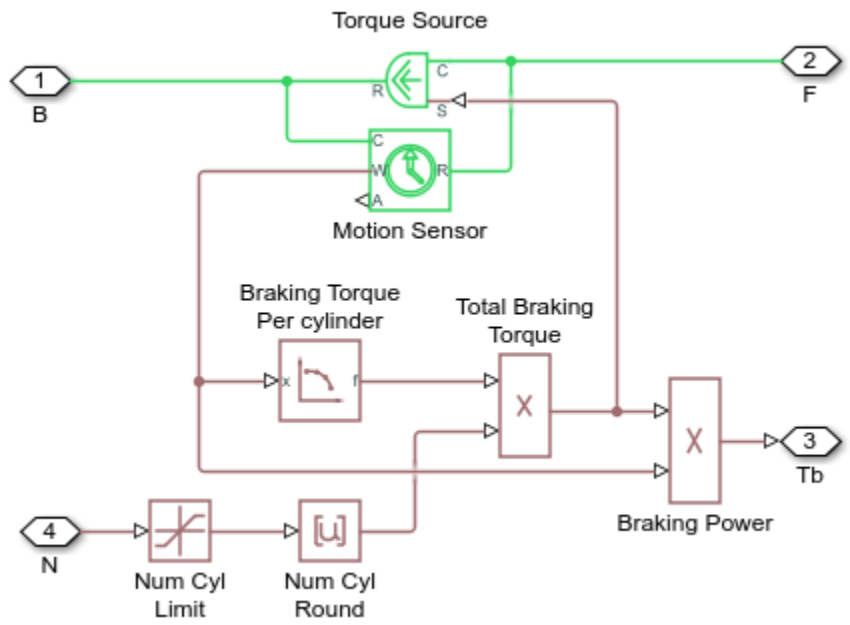
1. Plot engine speed (see code)
2. Plot engine and braking torque (see code)
3. Explore simulation results using Simscape Results Explorer
4. Learn more about this example

Copyright 2012-2022 The MathWorks, Inc.

Engine with Brake Subsystem

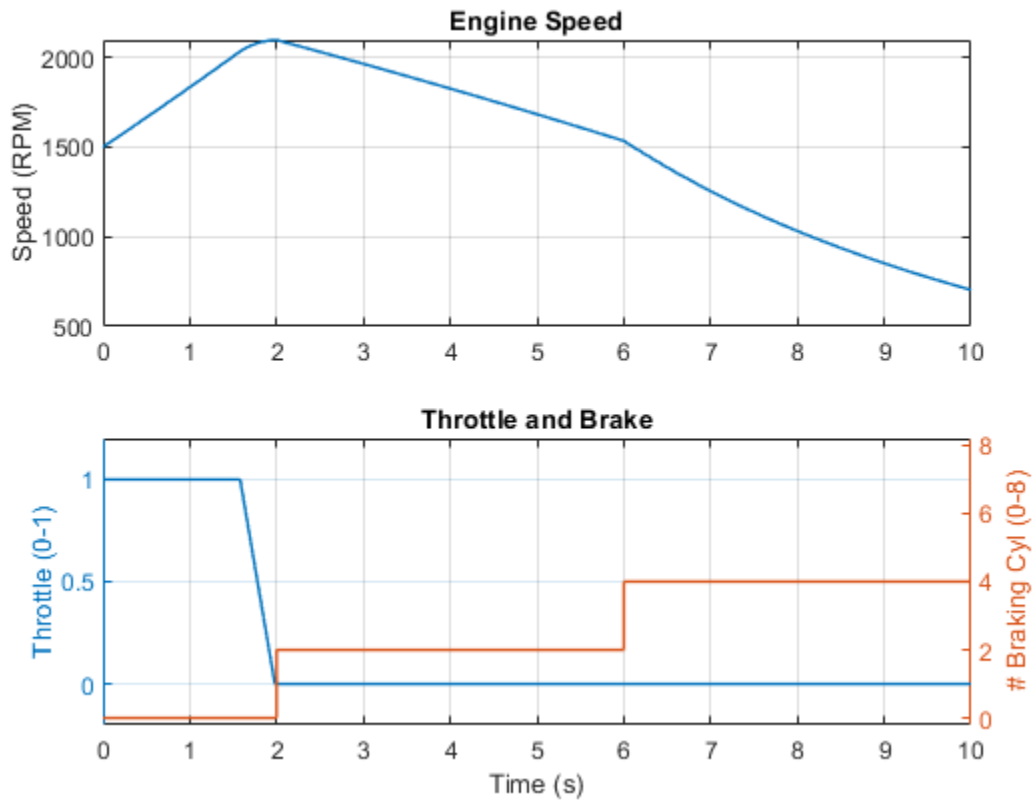


Total Braking Torque Subsystem

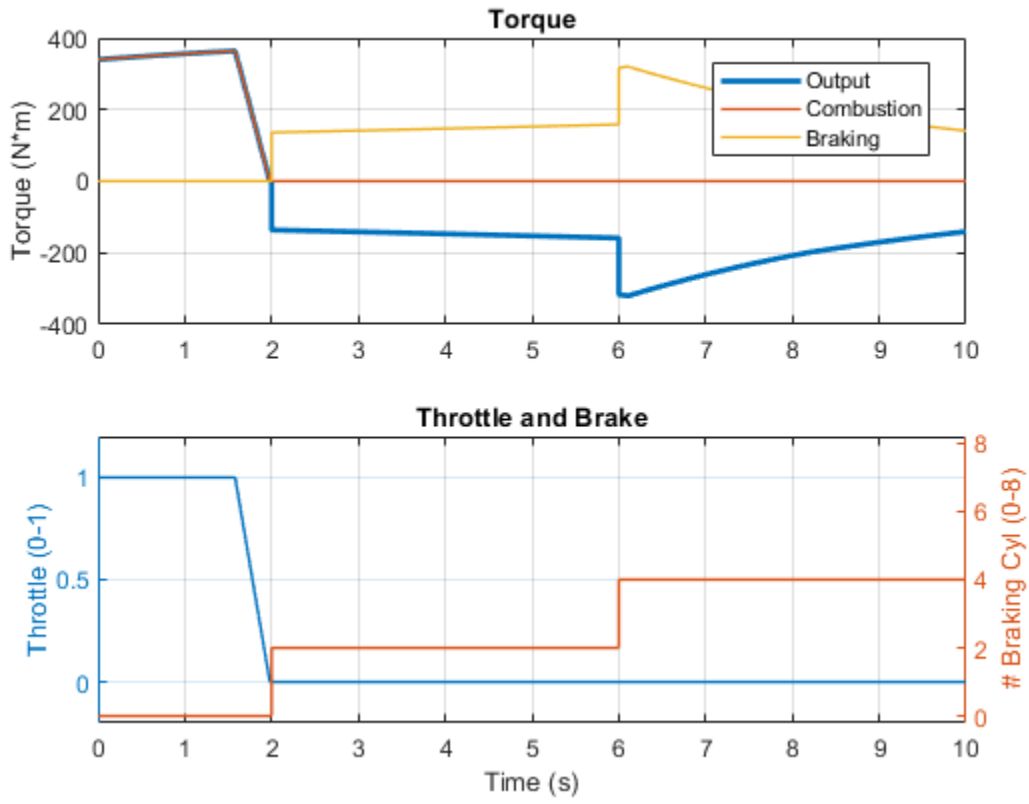


Simulation Results from Simscape Logging

This plot shows how the engine speed changes with the applied throttle and with the number of cylinders that are used for engine braking.



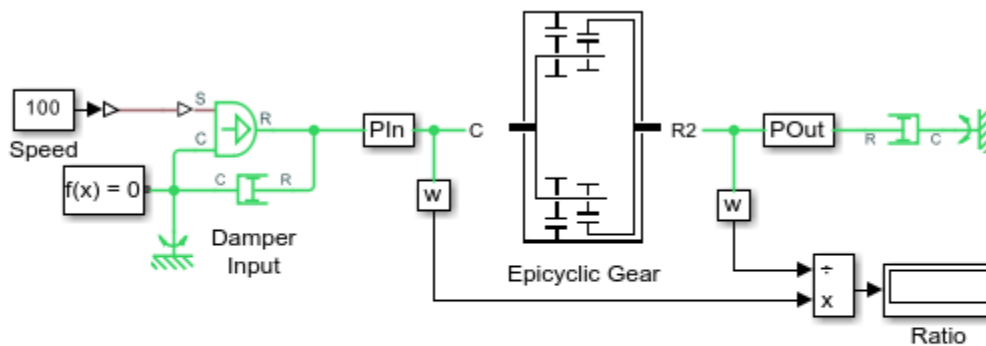
This plot shows the torque supplied by the engine varies with the applied throttle and with the number of cylinders that are used for engine braking.



Epicyclic Gear Efficiency Measurement

This example shows an epicyclic gear drive with overall ratio of 256:1 and verifies its transmission efficiency given the individual gear meshing efficiencies. The example is based on Tuplin, W.A. "Designing Compound Epicyclic Gear Trains for Maximum Speed at High Velocity Ratios", Machine Design, April 4, 1957. The drive is built of two ring-planet elements as shown in the gearbox schematic. The example confirms the transmission efficiency value of 0.22 as predicted by the analytical analysis presented in the paper.

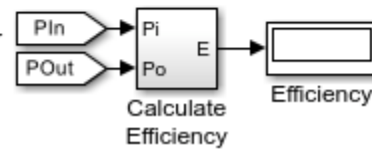
Model



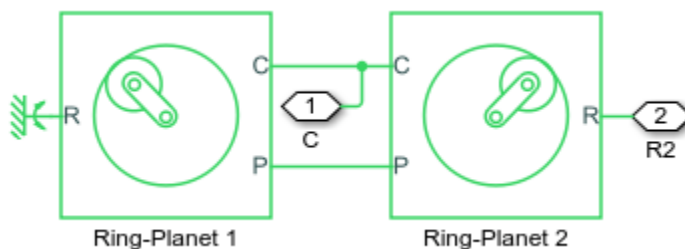
Epicyclic Gear Efficiency Measurement

1. Explore simulation results using Simscape Results Explorer
2. Learn more about this example

Copyright 2008-2022 The MathWorks, Inc.



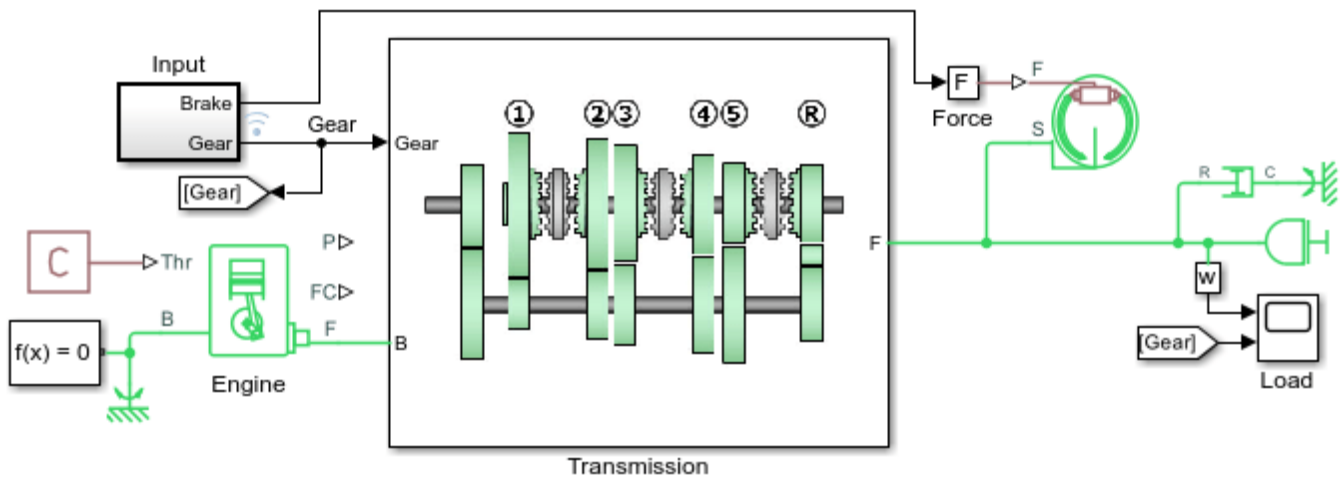
Epicyclic Gear Subsystem



Five-Speed Transmission

This example shows a five-speed transmission with a reverse gear. An engine spins the layshaft and the six sets of output gears. To engage the selected gear, ideal actuators move the selector levers connected to the double-sided synchronizers. The position of the selector levers determines which gears are connected to the output shaft.

Model

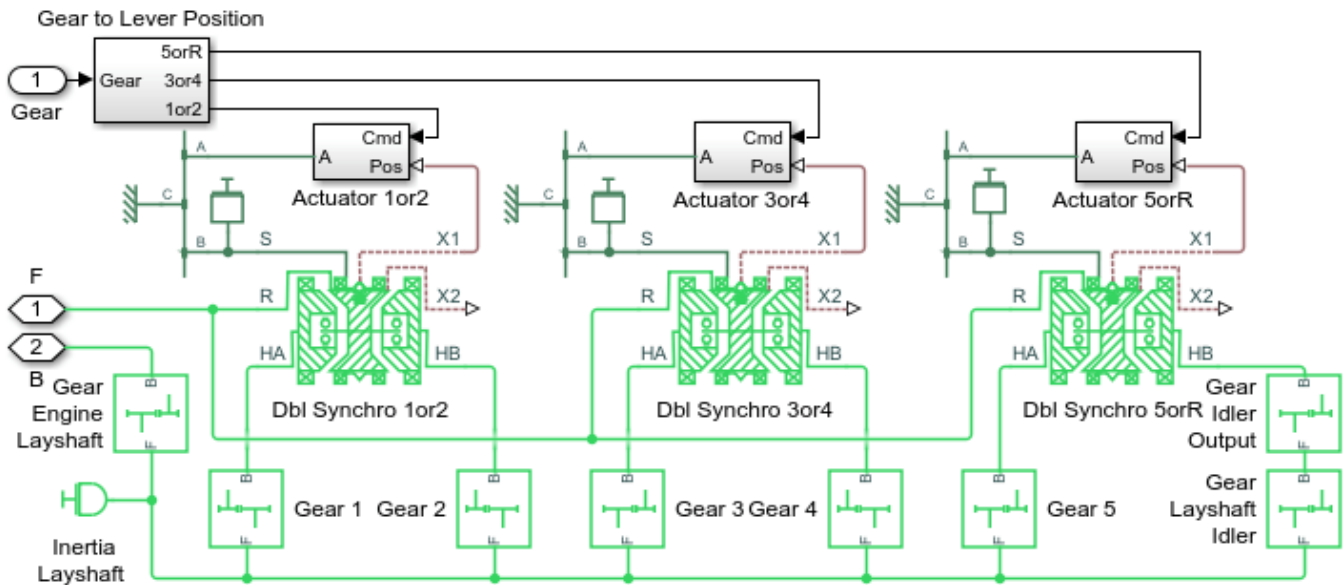


Five-Speed Transmission

1. Plot speeds in transmission (see code)
2. Plot selector lever position in transmission (see code)
3. Explore simulation results using Simscape Results Explorer
4. Learn more about this example

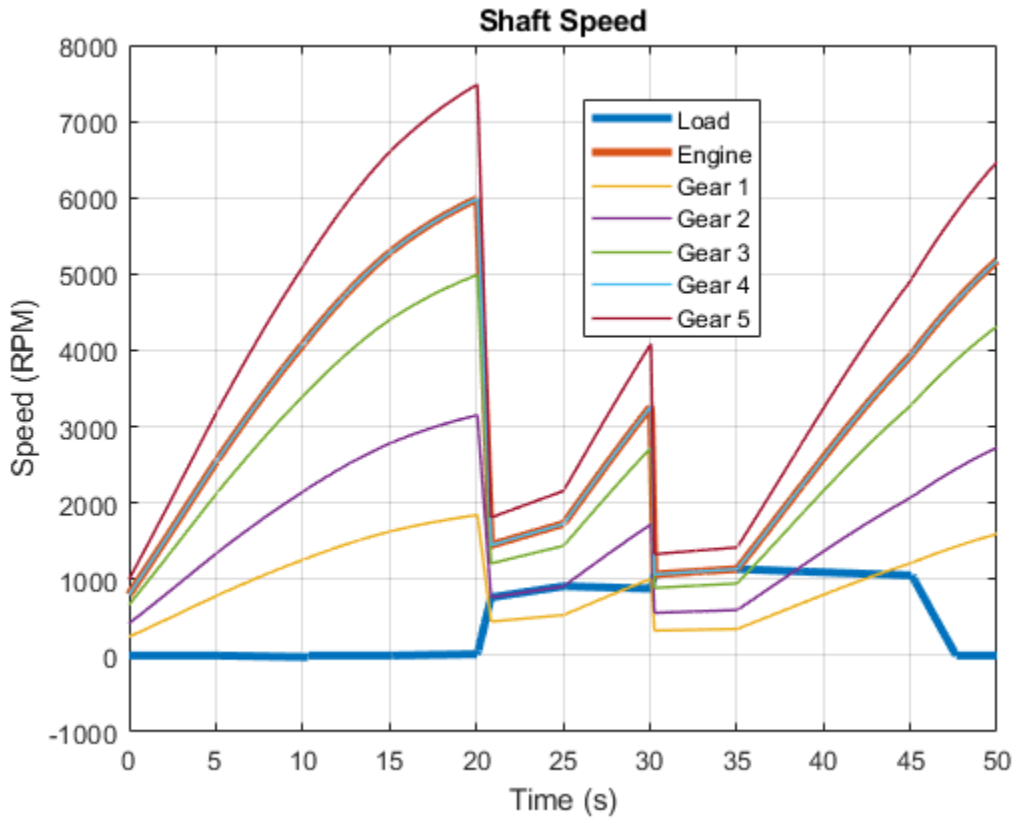
Copyright 2012-2022 The MathWorks, Inc.

Transmission Subsystem

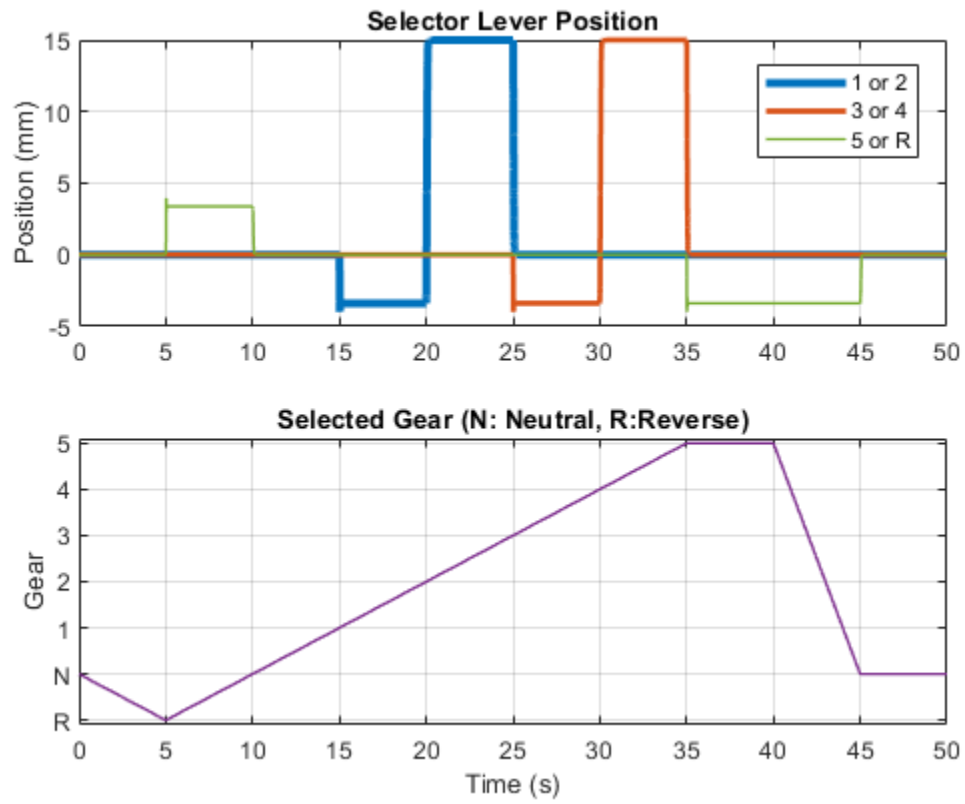


Simulation Results from Simscape Logging

The plot below shows the shaft speeds for the engine, load, and intermediate gears. As the transmission shifts between the gears, the load speed matches the speed of different gears in the transmission. When the reverse gear is engaged, the speed of the output shaft is negative.



The plot below shows the selector lever positions for each of the double-sided synchronizers. Their positions dictate which of the gears are engaged and determines the gear ratio for the transmission.

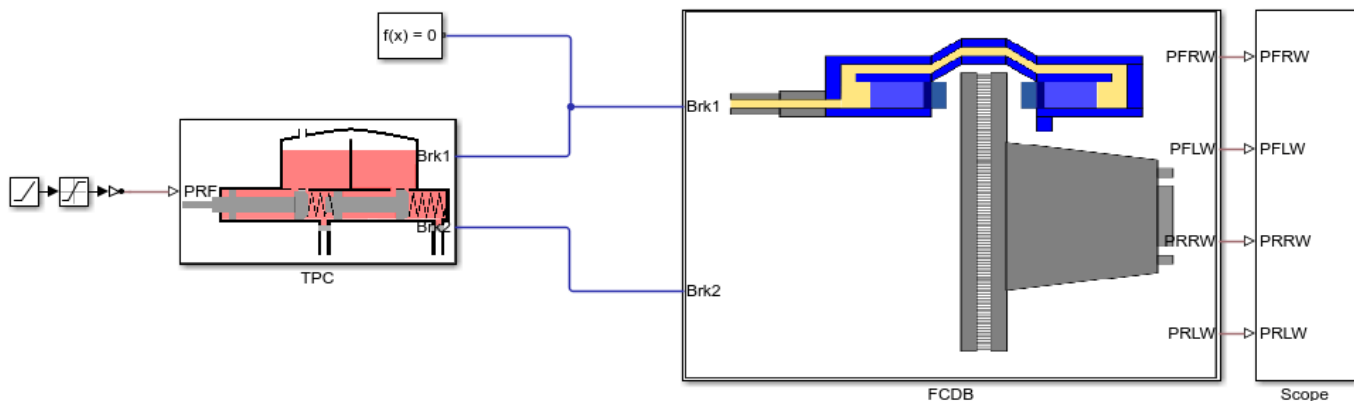


Fixed Caliper Disk Brake

This example shows how to model, parameterize, and test a caliper disk brake. The example uses numerical data extracted from the tandem primary cylinder datasheet to identify an optimal design for the caliper disk brake. The example shows how to generate a compliance curve for the caliper disk brake.

Model

The figure shows a model of the caliper disk brake with the test harness. A tandem primary cylinder is used in the inlet circuit for the caliper disk brake. The output of the model is the compliance curve for the rear and front caliper disk brakes.



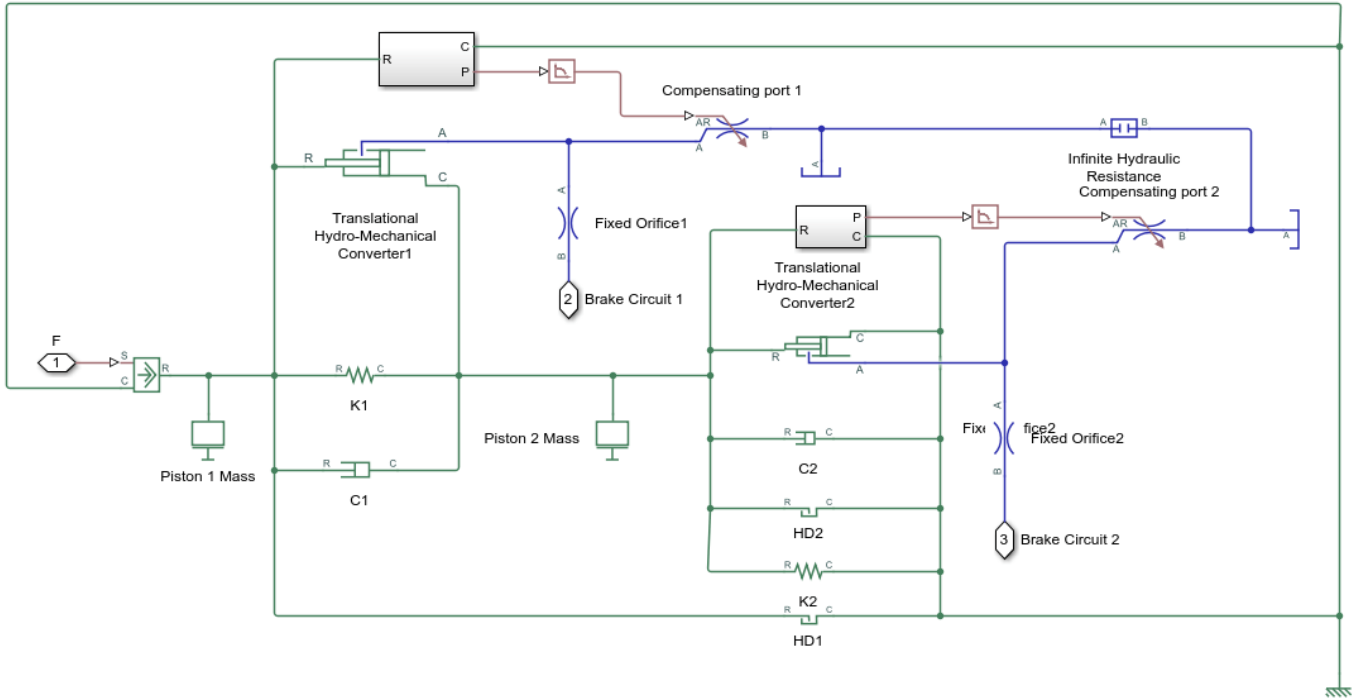
Fixed Caliper Disk Brake

1. Open parameterization file
2. Plot compliance curve (see code)
3. Explore simulation results using Simscape Results Explorer
4. Learn more about this example

Copyright 2018-2022 The MathWorks, Inc.

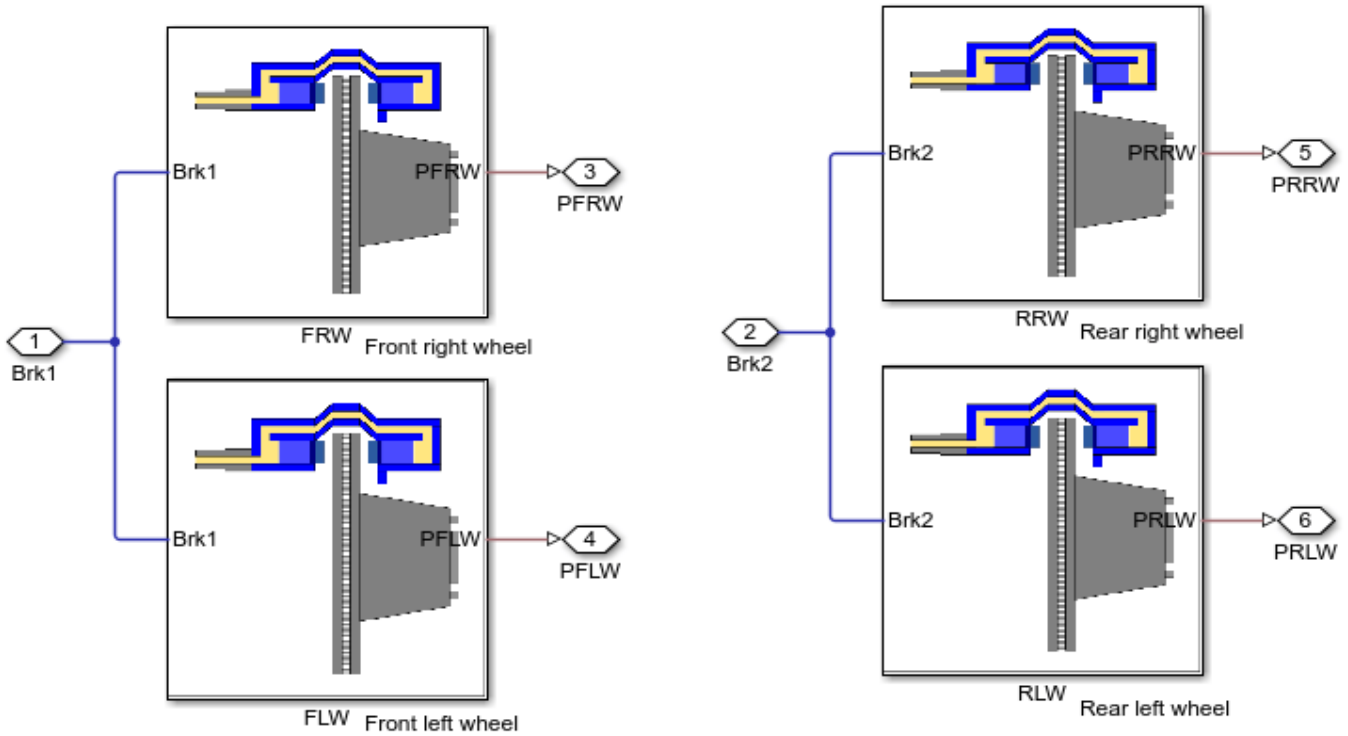
Tandem primary cylinder (TPC) subsystem

This subsystem shows the model of a tandem primary cylinder, which is used to pressurize the caliper disk brake.

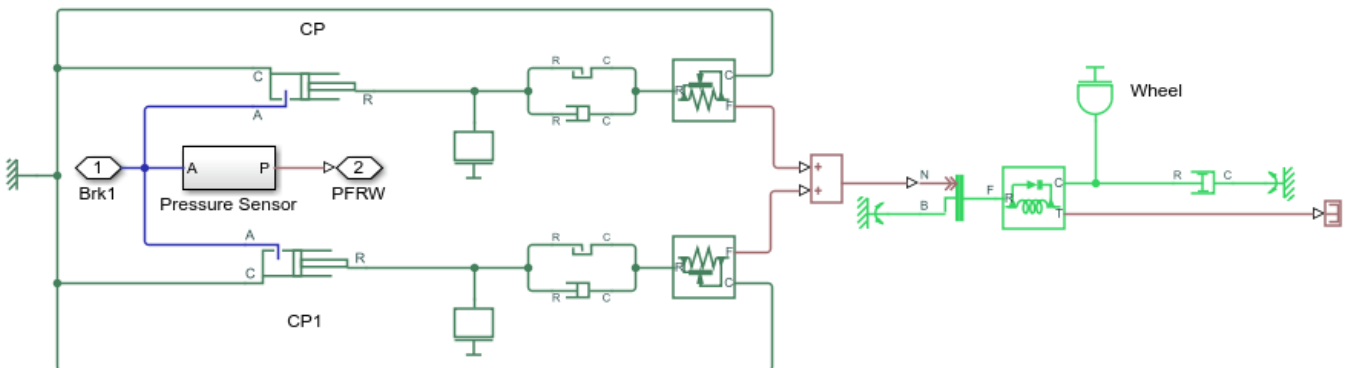


Fixed caliper disk brake (FCDB) subsystem

The subsystem shows the model of the caliper disk brakes for the four wheels. The fluid from the Brk1 channel of TPC is divided between the front two wheels. The fluid from the Brk2 channel of TPC is divided between the rear two wheels of the vehicle.

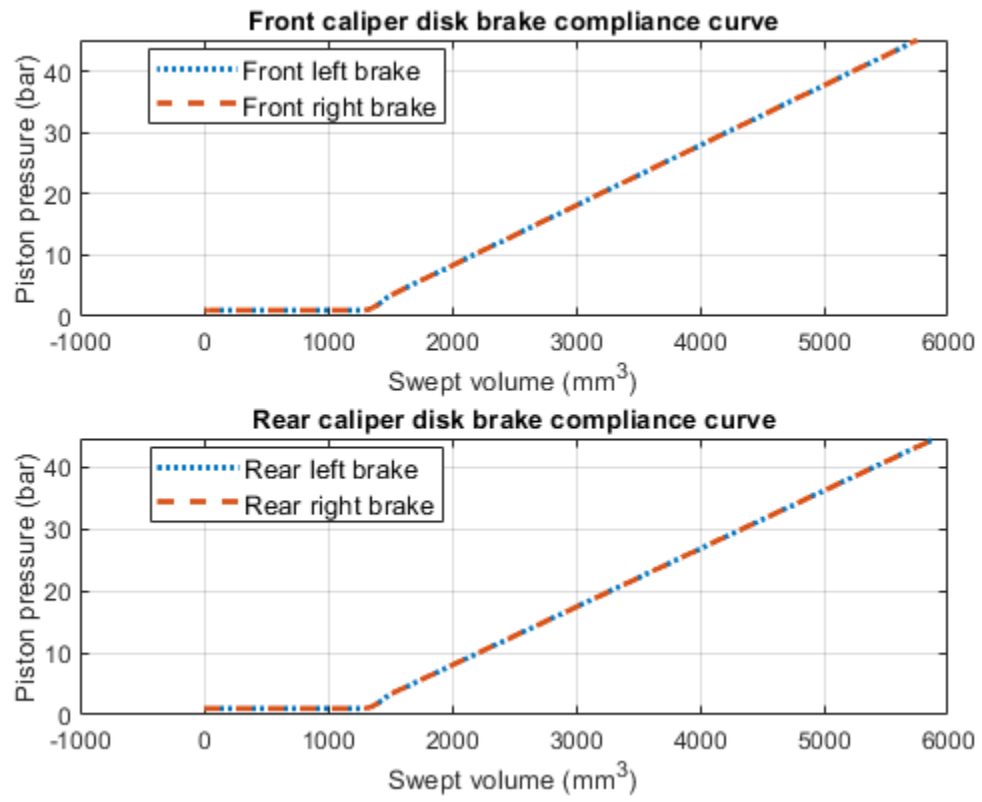


The figure shows the model of a caliper disk brake. The model does not have caliper disks connected to the vehicle wheels. The disks are grounded in the model. Therefore, the generated braking torque can be neglected.



Simulation results from Simscape logging

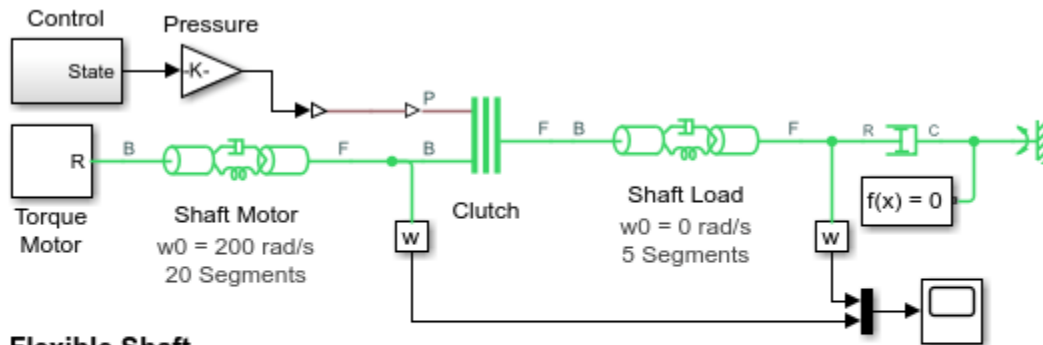
The model generates compliance curves of the caliper disk brakes for a selected case.



Flexible Shaft

This example shows two flexible aluminum shafts modeled using a lumped parameter approach. Both shafts consist of 20 segments containing inertias, damping, and stiff torsional springs. At the start of the simulation, the clutch is unlocked and the driven shaft is free. Note that the initial velocity of the motor shaft is set to 200 rad/s and that the system starts at steady state. The clutch engages and disengages during the test, and during each event the effect of the shaft flexibility can be seen in the simulation results.

Model



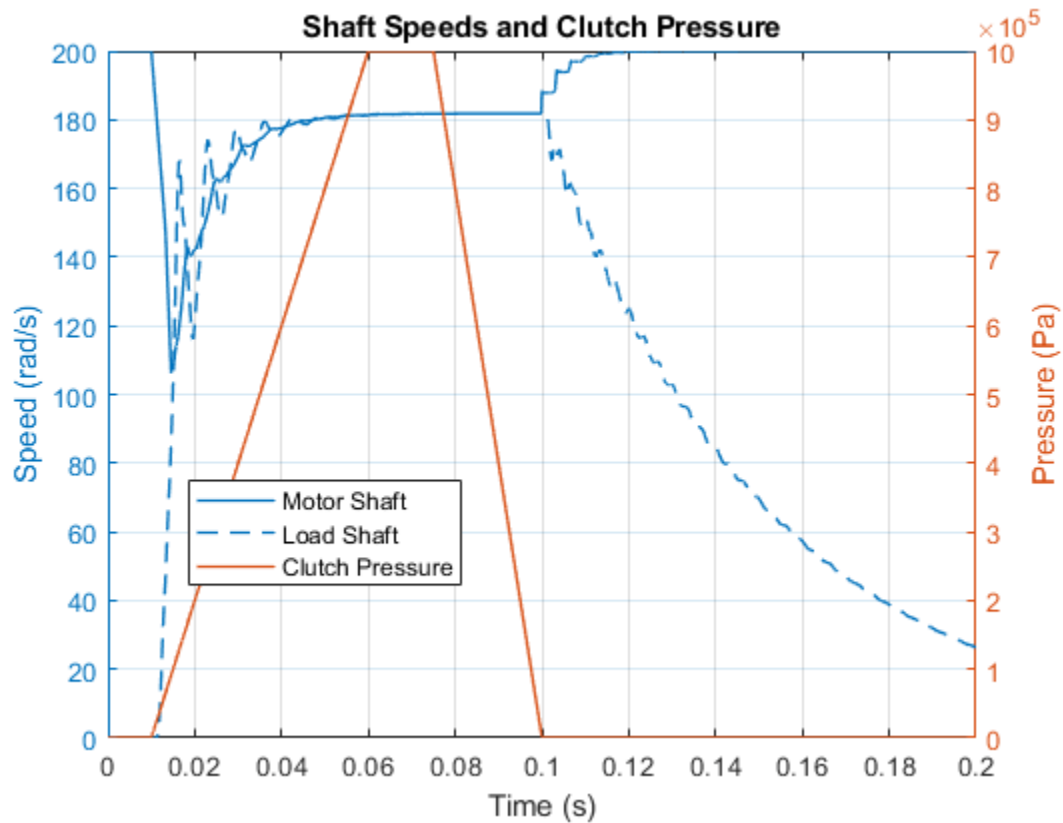
Flexible Shaft

1. Plot shaft speed (see code)
2. Plot twist in shaft compliance elements (see code)
3. Explore simulation results using Simscape Results Explorer
4. Learn more about this example

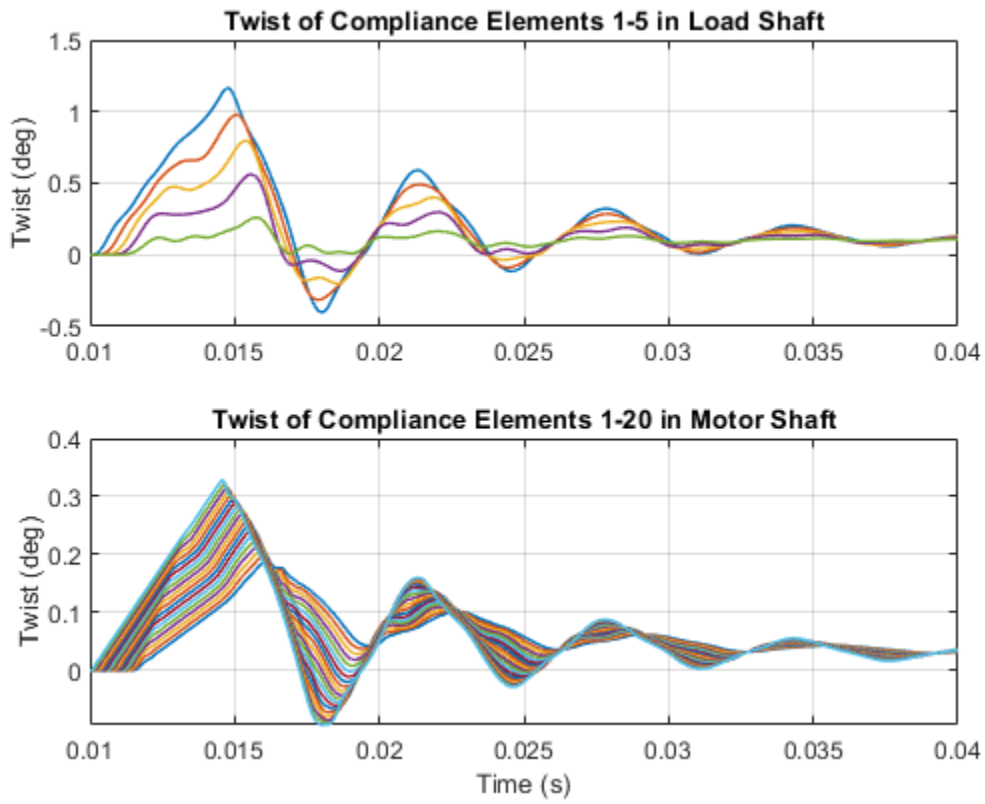
Copyright 2005-2022 The MathWorks, Inc.

Simulation Results from Simscape Logging

The plots below show the rotational speed of two flexible shafts as a clutch between them is locked and unlocked. The motor shaft is driven by a motor, and the load shaft is connected to a viscous damper. The oscillations triggered by the sudden engaging and disengaging of the clutch are due to the flexibility in the shafts.



The plots below show the amount of twist in each compliance element of the flexible shafts as a clutch between them is locked and unlocked. Though the stiffness and dimensions are the same for each shaft, their behavior is slightly different because they are subjected to different loads. The motor shaft is driven by a motor, and the load shaft is connected to a viscous damper.

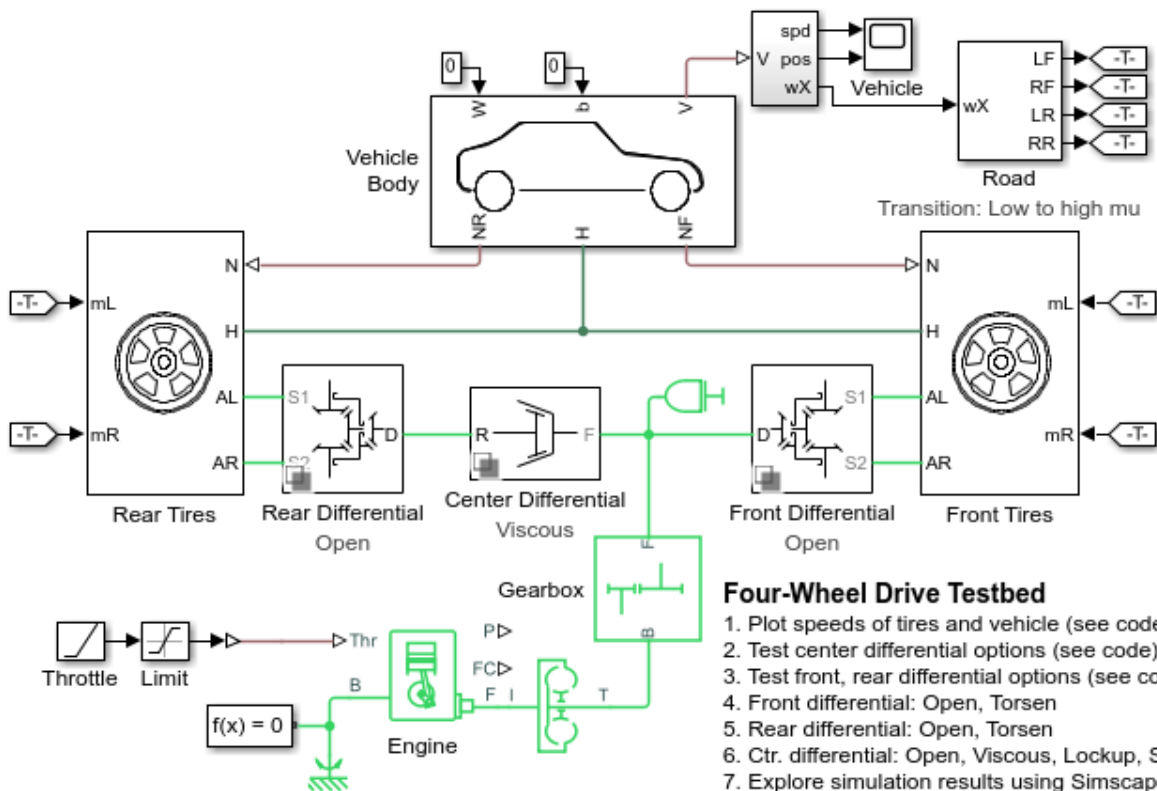


Four-Wheel Drive Testbed

This example shows a four-wheel drive vehicle with open and limited slip differentials. The front and rear differentials can be standard or Type I Torsen. The center differential can be a solid shaft, viscous coupling, viscous coupling with a locking clutch, or open with no torque transfer. The differential options are in variant subsystems. The variants can be selected using the hyperlinks embedded in the model.

To test the effect of limited slip differentials, the road surface in the model can be configured such that the tires are on surfaces with varying coefficients of friction. This is configured in the Road subsystem.

Model

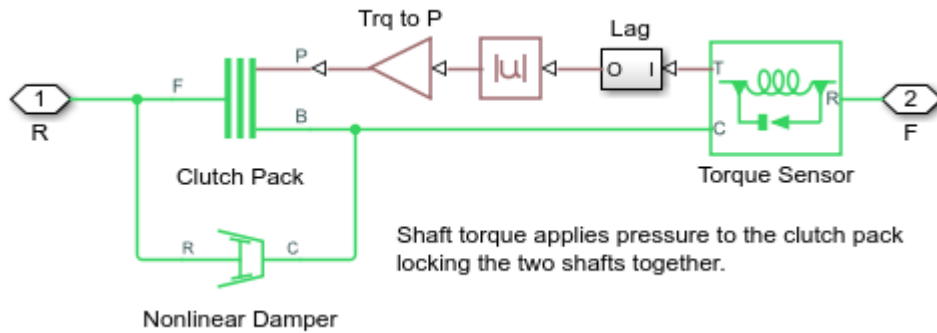


Four-Wheel Drive Testbed

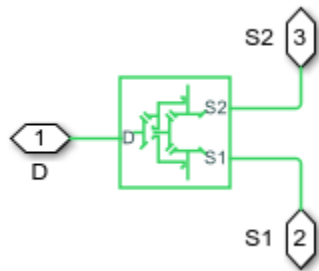
1. Plot speeds of tires and vehicle (see code)
2. Test center differential options (see code)
3. Test front, rear differential options (see code)
4. Front differential: Open, Torsen
5. Rear differential: Open, Torsen
6. Ctr. differential: Open, Viscous, Lockup, Shaft
7. Explore simulation results using Simscape Results Explorer
8. Learn more about this example

Copyright 2014-2022 The MathWorks, Inc.

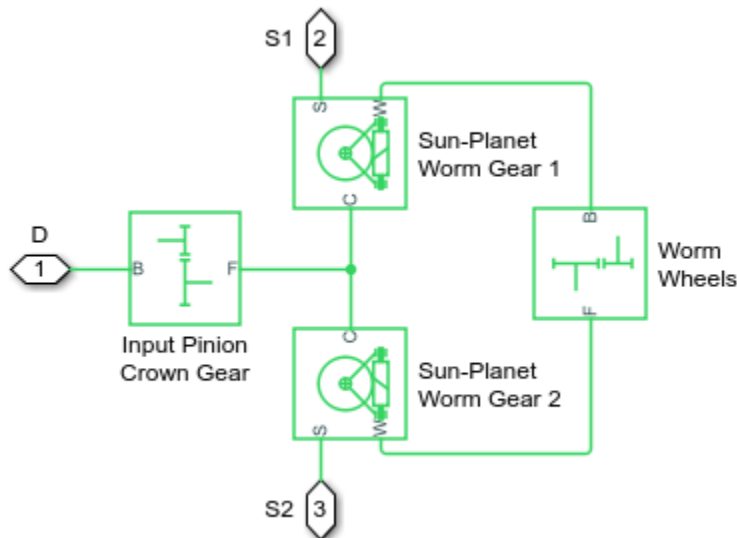
Lockup Clutch Subsystem for Center Differential



Open Differential Subsystem for Front and Rear



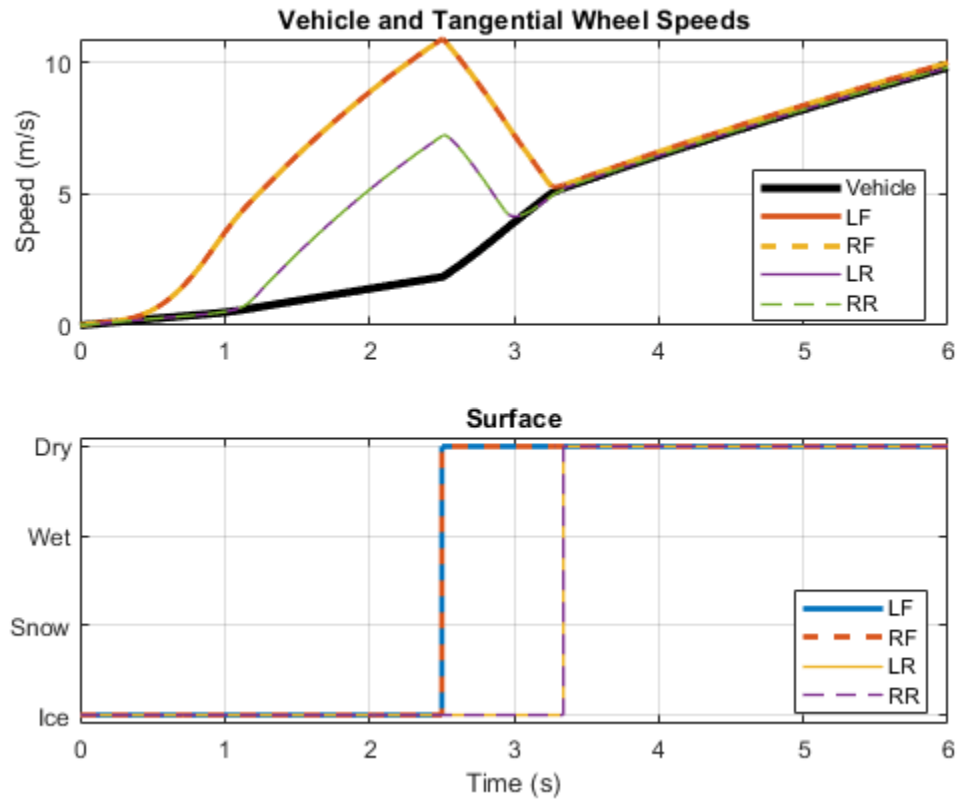
Torsen Differential Subsystem for Front and Rear



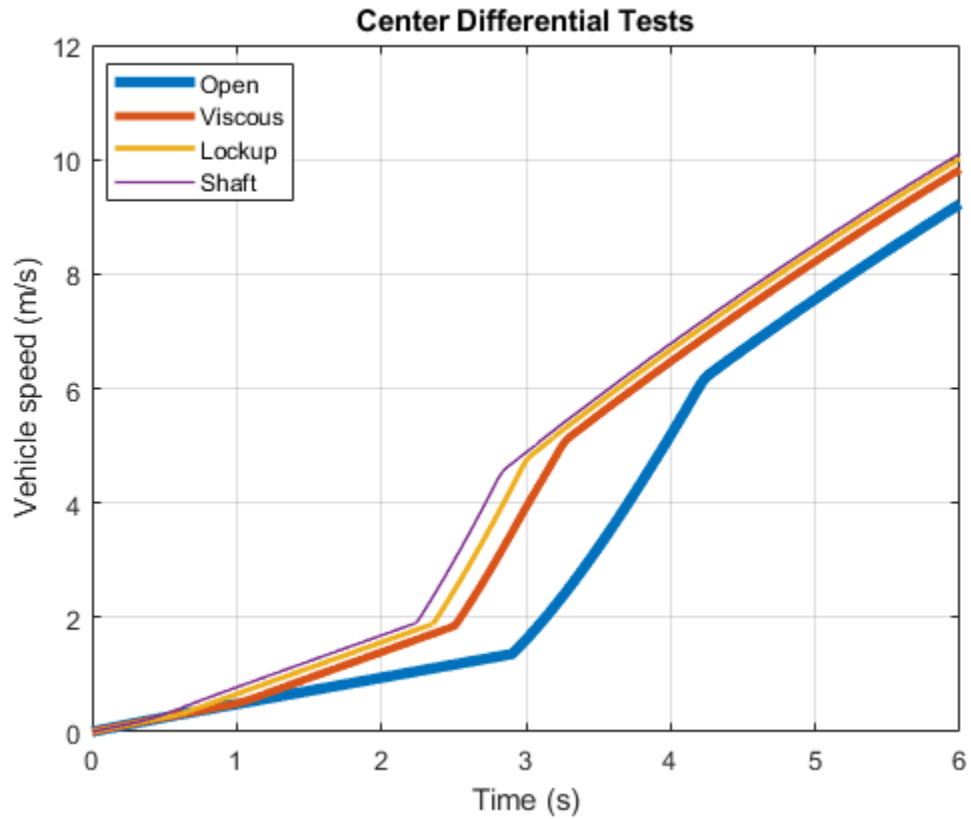
Simulation Results from Simscape Logging

The plot below shows the results of a four-wheel drive vehicle accelerating from rest with all four wheels on a low friction surface, such as ice. Two meters in front of the vehicle starting position is a high friction surface, such as dry tarmac. Open differentials are used on the front and rear and a

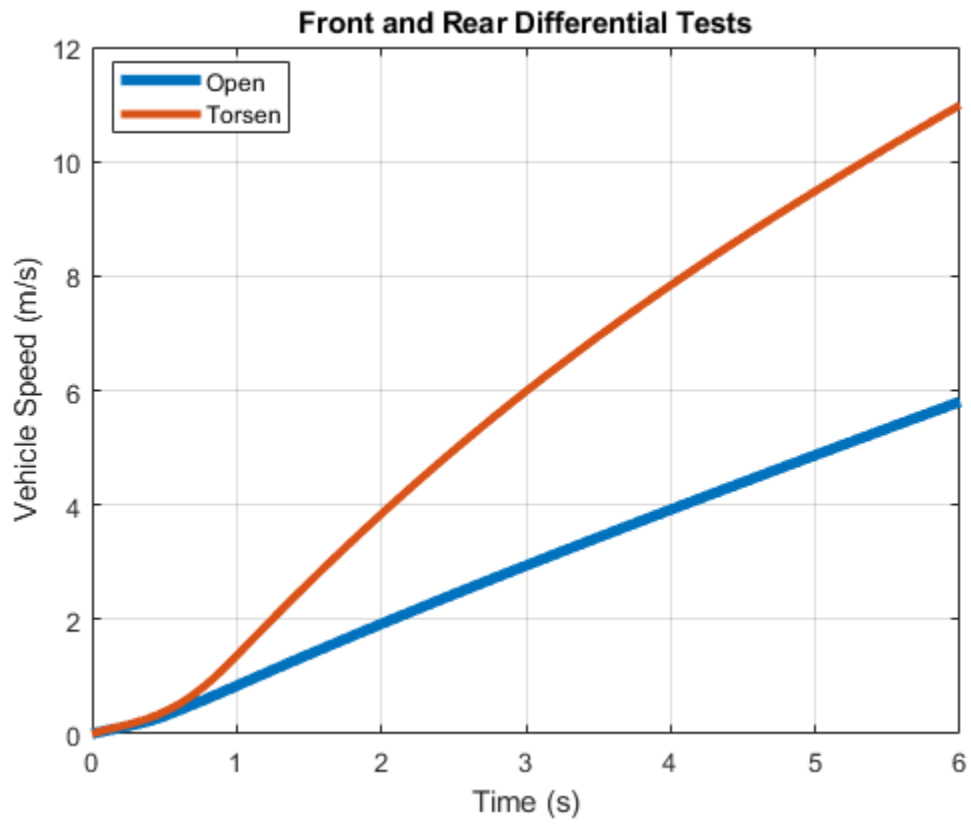
viscous coupling is used on the center differential. As a result, all four wheels slip until all reach a high friction surface.



The plot below shows the results of a four-wheel drive vehicle accelerating from rest with the front wheels on a low friction surface, such as ice, and the front wheels on a high friction surface, such as dry tarmac. The test is run four times with different configurations for the center differential. The shaft connection and the lockup clutch provide the fastest acceleration.



The plot below shows the results of a four-wheel drive vehicle accelerating from rest with the left wheels on a low friction surface, such as ice, and the right wheels on a high friction surface, such as dry tarmac. The test is run twice with different configurations for the front and rear differential. The Torsen differential automatically locks under the split surface condition, resulting in faster acceleration.

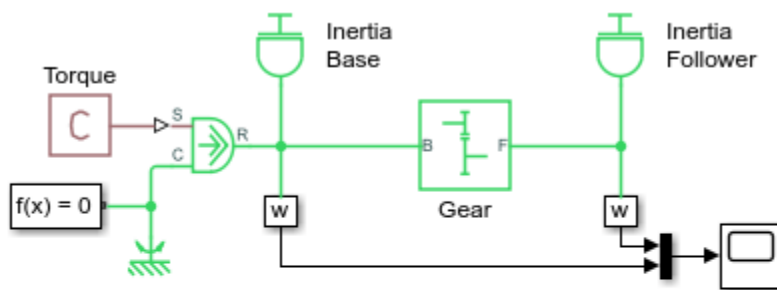


Gear with Backlash

This example shows how to model a gear with backlash using the Simple Gear block with two different backlash models: a spring-damper model and a coefficient of restitution model. The backlash is set to 1 mm, which is equivalent to 0.032 degrees rotation of the input gear that has a radius of 100 mm or 0.016 degrees rotation of the output gear that has a radius of 200 mm. Parameters in the two backlash models are adjusted so that there are a few oscillations when the torque is applied, and the models have similar behavior.

The Simscape™ Local Solver is configured so that it will provide nearly equivalent results when it is enabled. Discontinuous physical effects such as backlash require a small step size to capture the behavior accurately. The Simscape Local Solver can be enabled in the Solver Configuration block. The coefficient of restitution gear backlash model is better suited than the spring-damper backlash models for fixed step simulations such as Hardware In the Loop (HIL). In this model, the coefficient of restitution model can take step sizes ten times larger than the spring-damper model.

Model



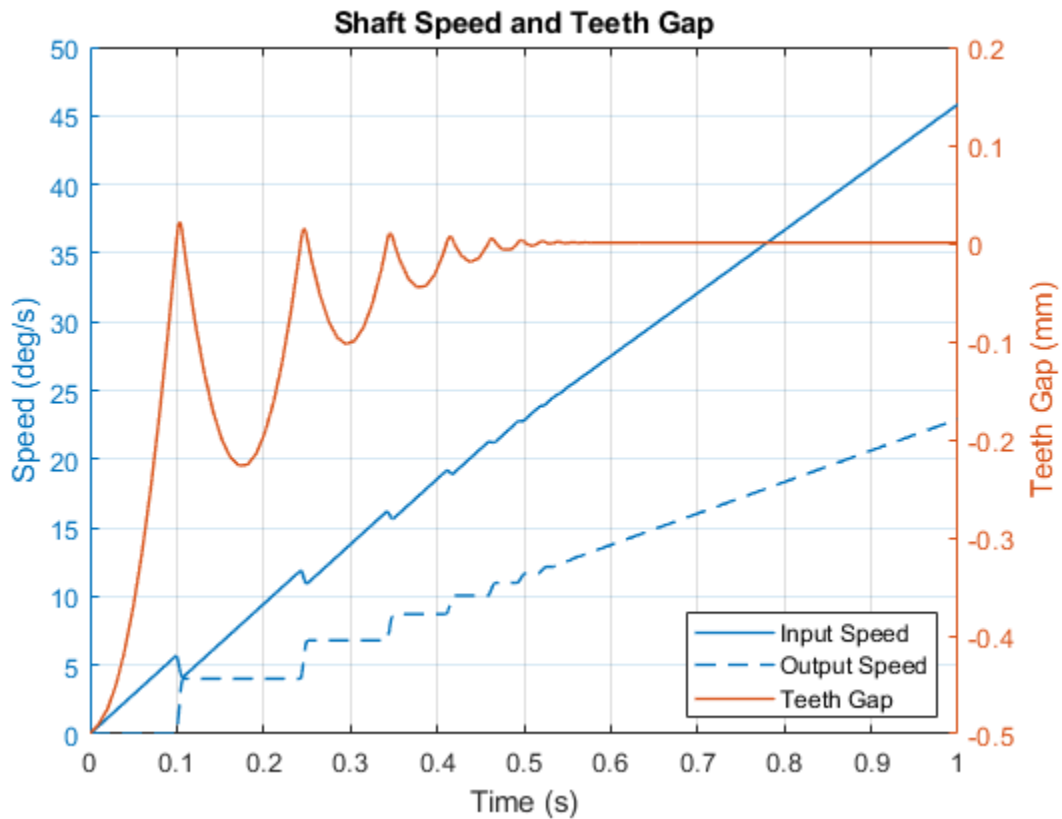
Gear with Backlash

1. Set backlash model: Spring-damper or Coefficient of restitution
2. Set solver: Global Simulink Solver or Local Simscape Solver
3. Plot shaft speeds and gap between gears (see code)
4. Compare backlash models (see code)
5. Explore simulation results using Simscape Results Explorer
6. Learn more about this example

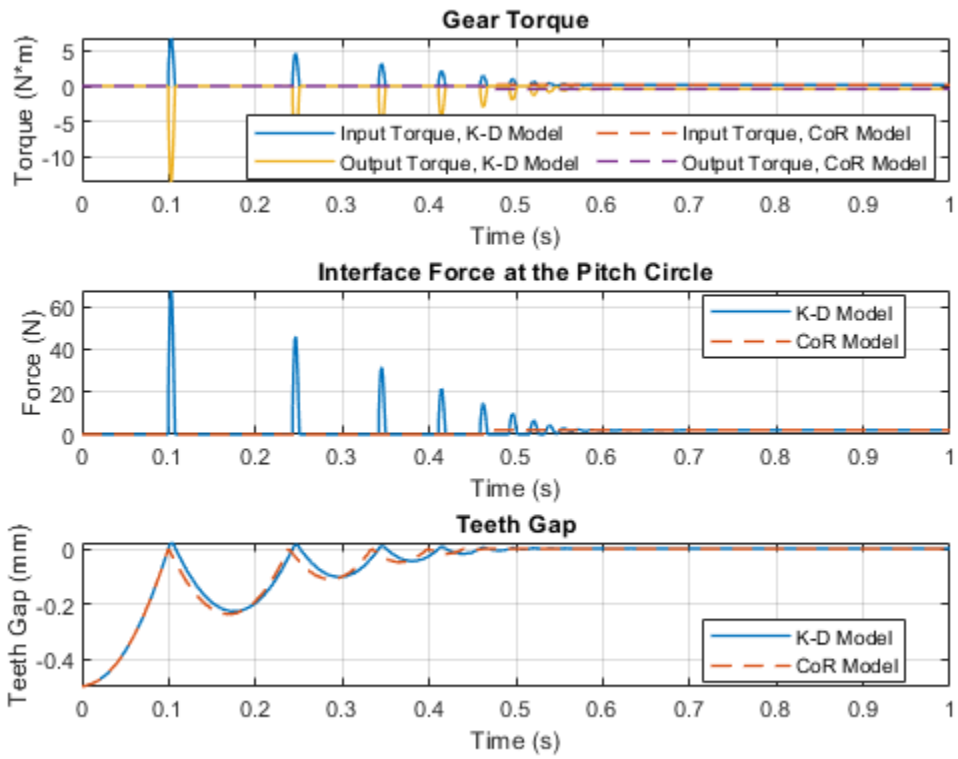
Copyright 2003-2022 The MathWorks, Inc.

Simulation Results from Simscape Logging

The plot below shows the speed of two shafts connected by gears with backlash. The input shaft is driven by a constant torque. When the simulation starts, the gear is positioned in the center of the backlash gap. As the input gear strikes the output gear, each impact accelerates the output shaft. Eventually, the shafts spin at constant accelerations in response to the torque source.



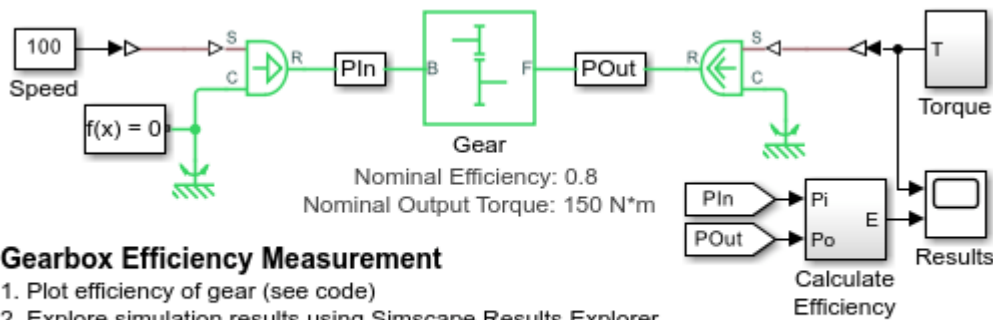
The plots below compare the gear torque, gear interface force, and teeth gap of the two backlash models. The model parameters have been tuned so that they have similar backlash oscillations. The Coefficient of Restitution backlash model uses instantaneous modes to simulate the impact between gears. While the instant impulses do not appear in the Simscape log, the model can capture cascading dynamics in connected blocks due to the impulses. The models have identical torque behavior in steady-state when the backlash oscillations end.



Gearbox Efficiency Measurement

This example shows a test for measuring gear box efficiency. The input shaft is driven at a fixed speed and a variable torque is applied to the output shaft. Power is measured on both the input and output shafts, and efficiency is calculated as the ratio of output to input power. The simulation results match the parameters specified for the efficiency of the gear.

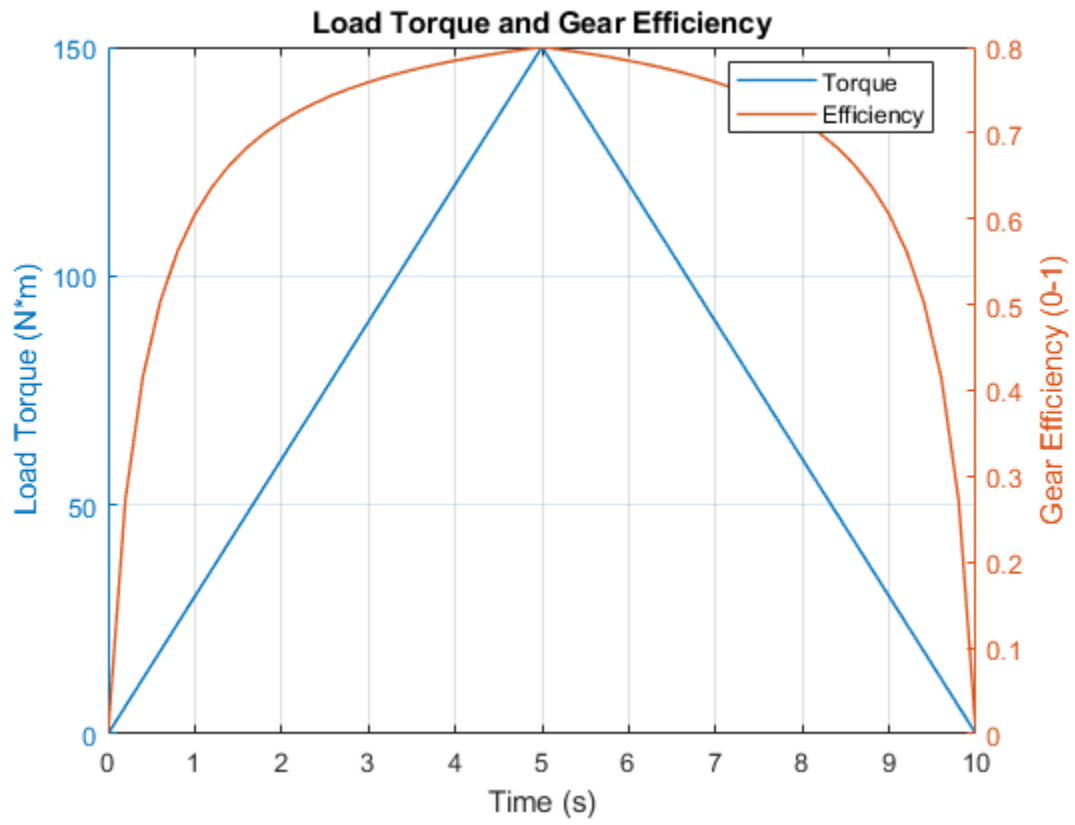
Model



Copyright 2008-2022 The MathWorks, Inc.

Simulation Results from Simscape Logging

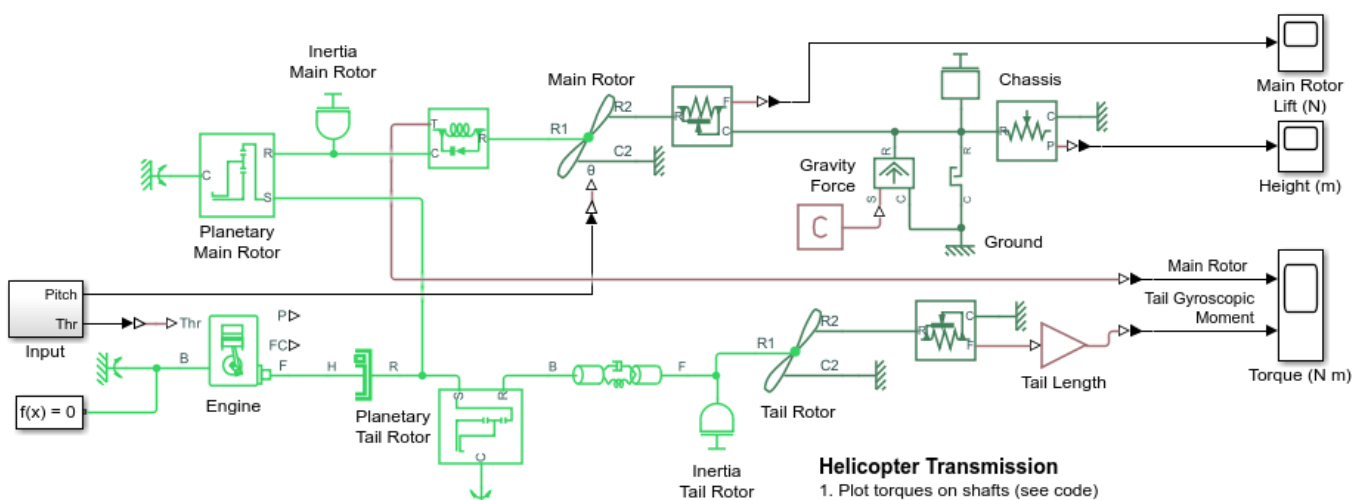
The plot below shows load-dependent efficiency in a gear model. A gear is driven at a constant speed as the load torque is varied. The efficiency at the nominal point exactly matches the parameter values in the block.



Helicopter Transmission

This example shows a fixed helicopter transmission testbed. The gasoline engine provides power that is sent to the main and tail rotors through planetary gearsets. The main rotor shaft is considered rigid, and flexibility is modeled in the longer, thinner tail rotor shaft. Blade pitch angle is set directly via a physical signal. Swash plate dynamics can be added using the appropriate blocks in Simscape™ Fluids™. The engine stalls after six seconds, but lift is maintained for some time through the unidirectional clutch. This scheme also allows autorotation of the main rotor in a descending helicopter.

Model



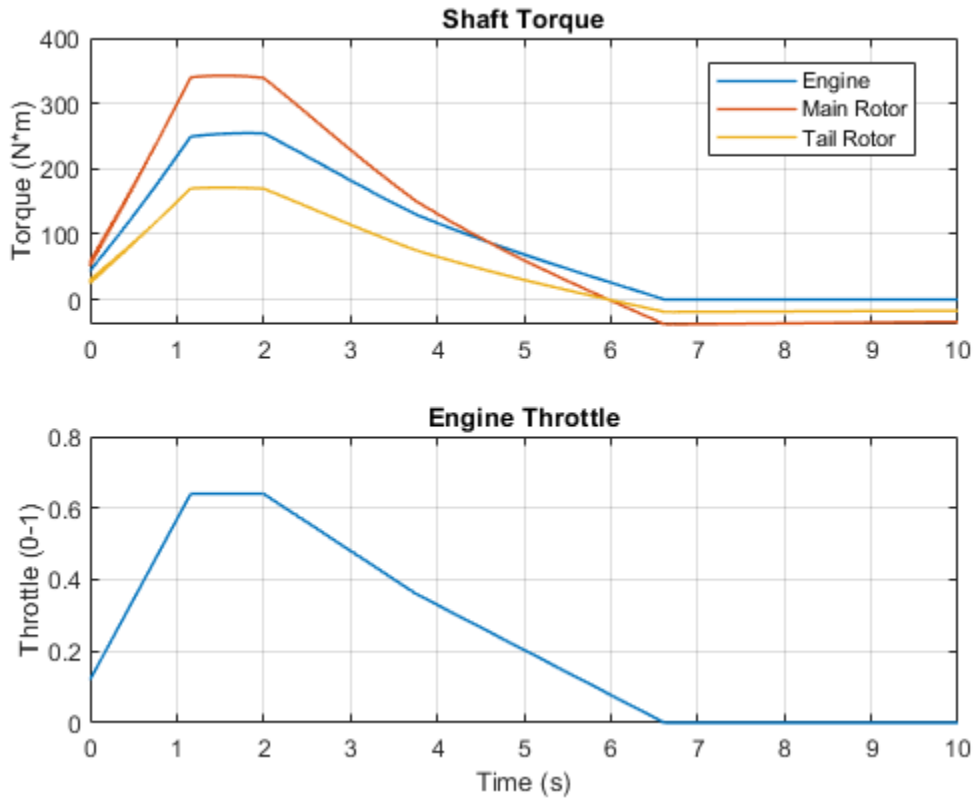
Helicopter Transmission

1. Plot torques on shafts (see code)
2. Plot main rotor coefficients (see code)
3. Plot tail rotor coefficients (see code)
4. Explore simulation results using Simscape Results Explorer
5. Learn more about this example

Copyright 2013–2022 The MathWorks, Inc.

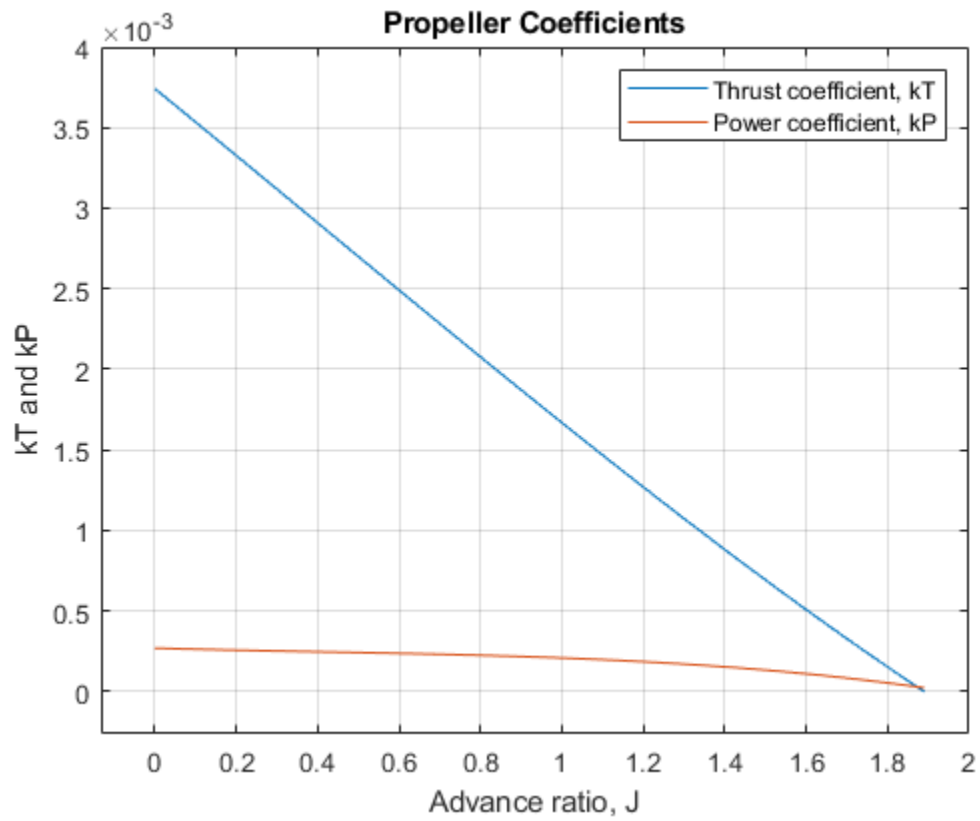
Simulation Results from Simscape Logging

The plots below show the torque supplied by the engine and the torque applied to the rotor shafts. The engine is permitted to stall after 6 seconds. Induced and profile drag on the spinning rotors act to slow them down.



Thrust and Torque Curves based on Block Parameters

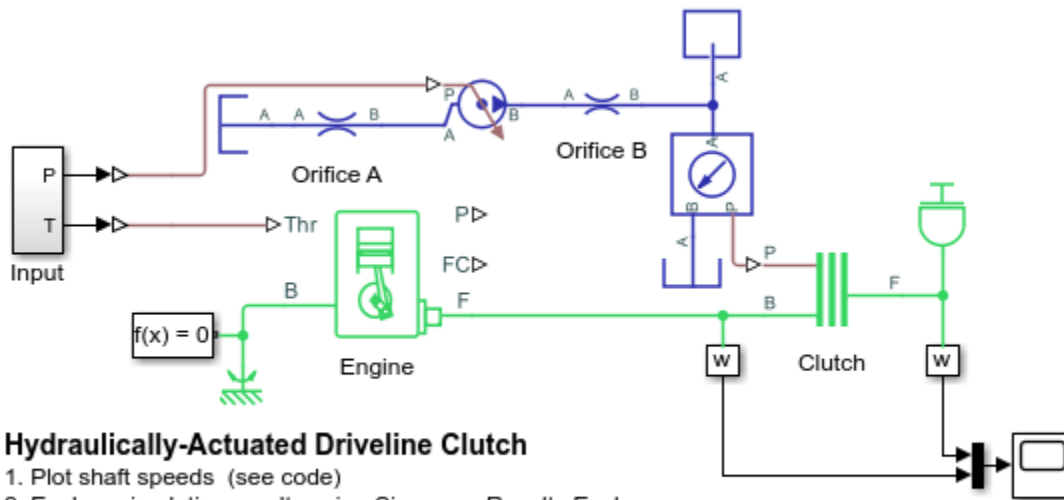
These plots show the main (Top) and tail (Bottom) rotor thrust and torque coefficients versus advance ratio defined in the block property inspector.



Hydraulically-Actuated Driveline Clutch

This example shows an engine driving an inertial load via a hydraulically-controlled clutch. It shows how Simscape™ Driveline™ can be used in conjunction with Foundation Library isothermal liquid blocks to model hydraulically-actuated drivelines.

Model



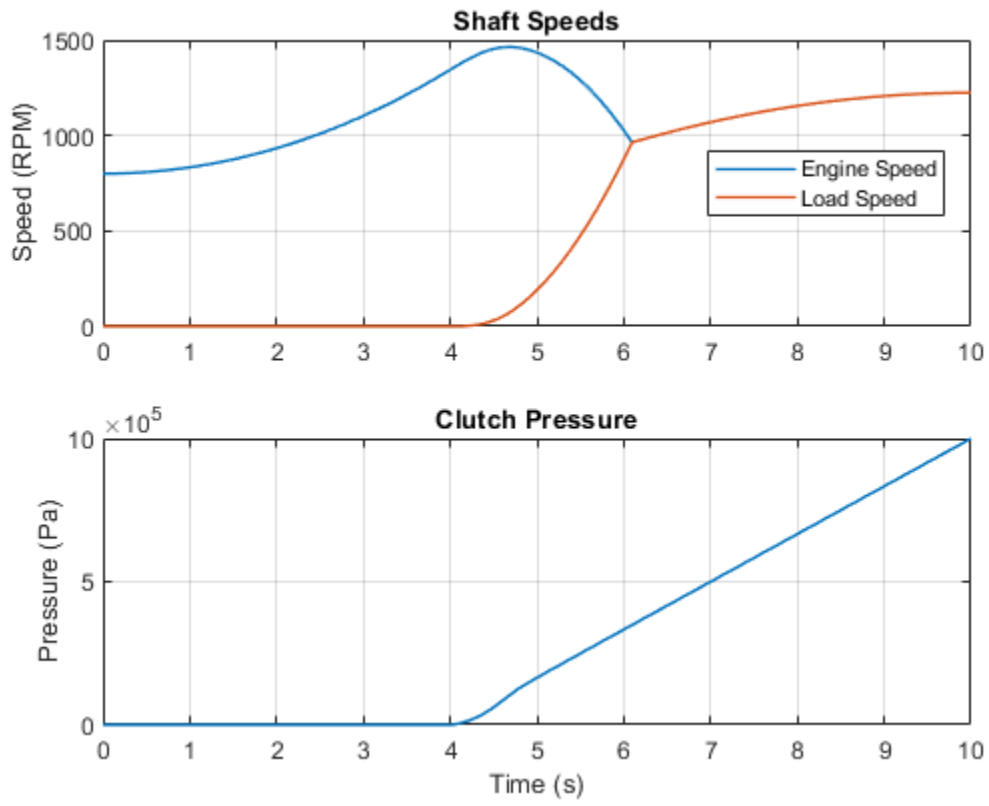
Hydraulically-Actuated Driveline Clutch

1. Plot shaft speeds (see code)
2. Explore simulation results using Simscape Results Explorer
3. Learn more about this example

Copyright 2006-2022 The MathWorks, Inc.

Simulation Results from Simscape Logging

The plot below shows the speeds of an engine and load as a clutch is engaged. Once the clutch is fully engaged the shafts rotate at the same speed. The clutch actuator is modeled as a hydraulic system.

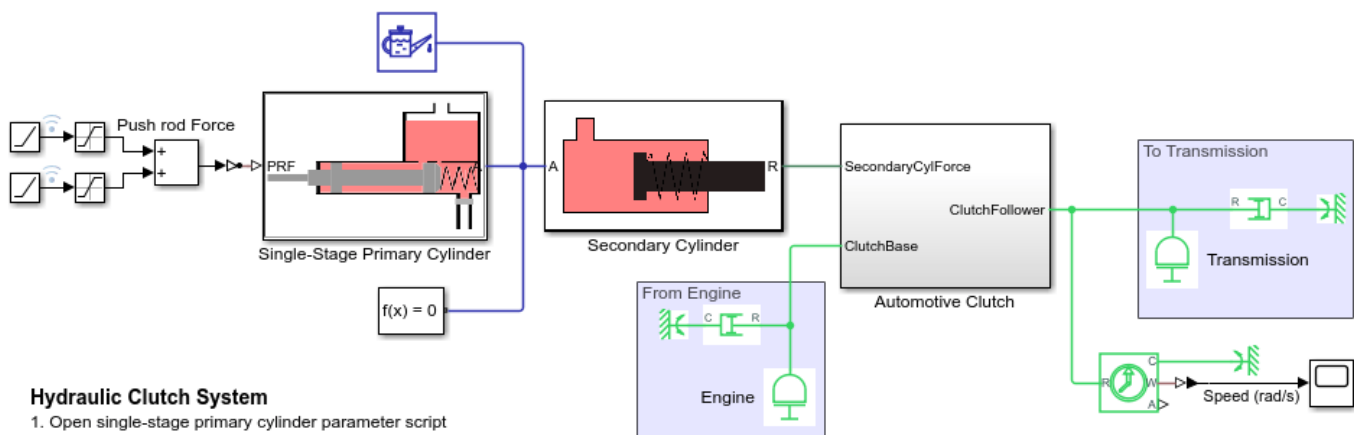


Hydraulic Clutch System

This example shows how to model, parameterize, and test a hydraulic clutch system. The model is used to generate the plot of the engine and the transmission system speeds during a declutching and clutch re-engaging scenario.

Model

The following figure shows the model of a hydraulic clutch system in a test harness. The output of the model is a plot of speeds versus time of the engine and the transmission systems during the declutching and clutch re-engaging scenario.



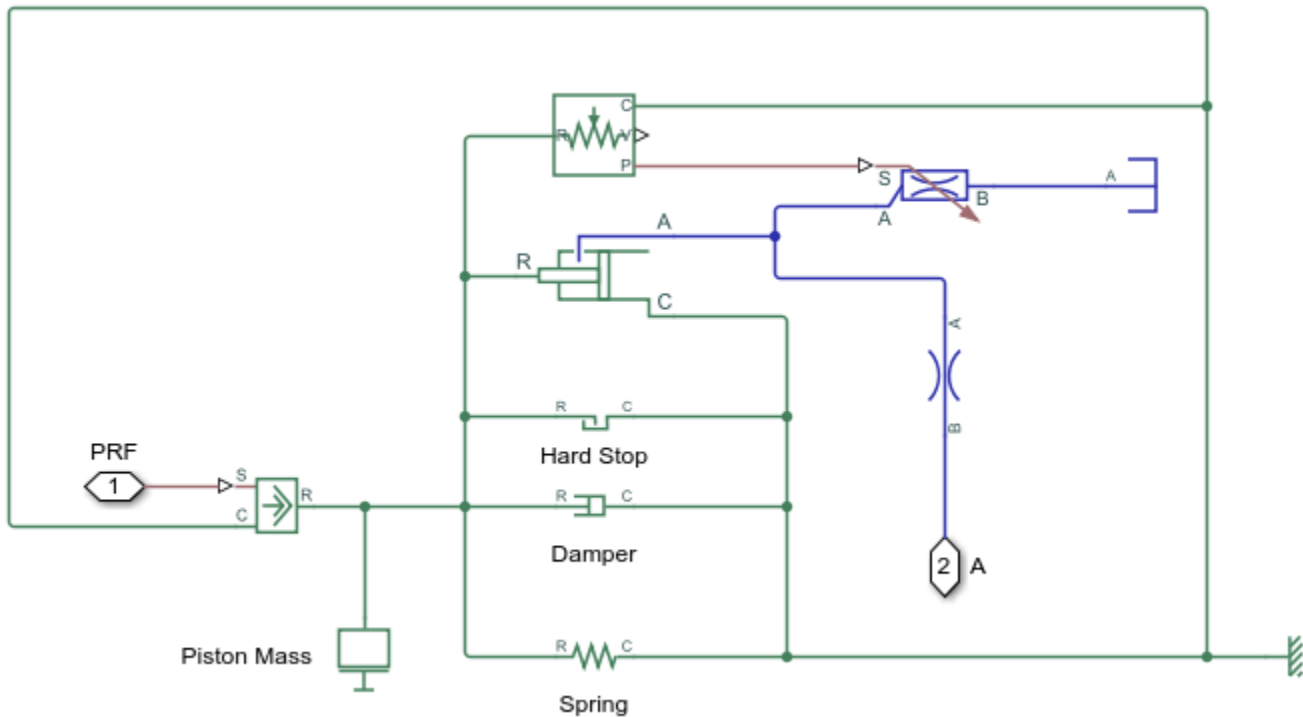
Hydraulic Clutch System

1. Open single-stage primary cylinder parameter script
2. Open secondary cylinder parameter script
3. Open automotive clutch parameter script
4. Plot function diagram (see code)
5. Explore simulation results using Simscape Results Explorer
6. Learn more about this example

Copyright 2020-22 The MathWorks, Inc.

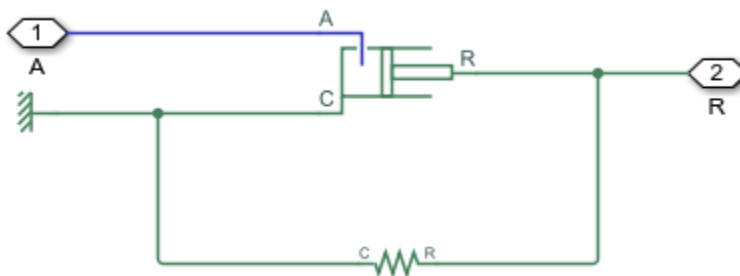
Single-Stage Primary Cylinder Subsystem

The following figure shows the modeling part of the single-stage primary cylinder.



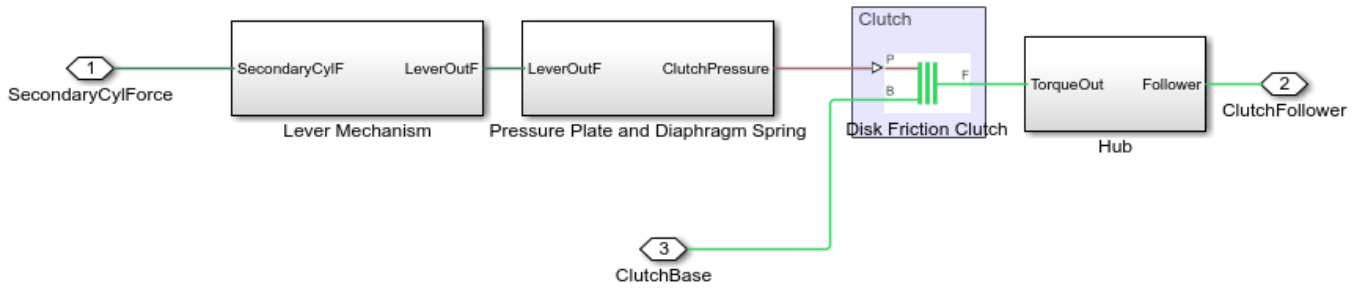
Secondary Cylinder Subsystem

The following figure shows the modeling part of the secondary cylinder.



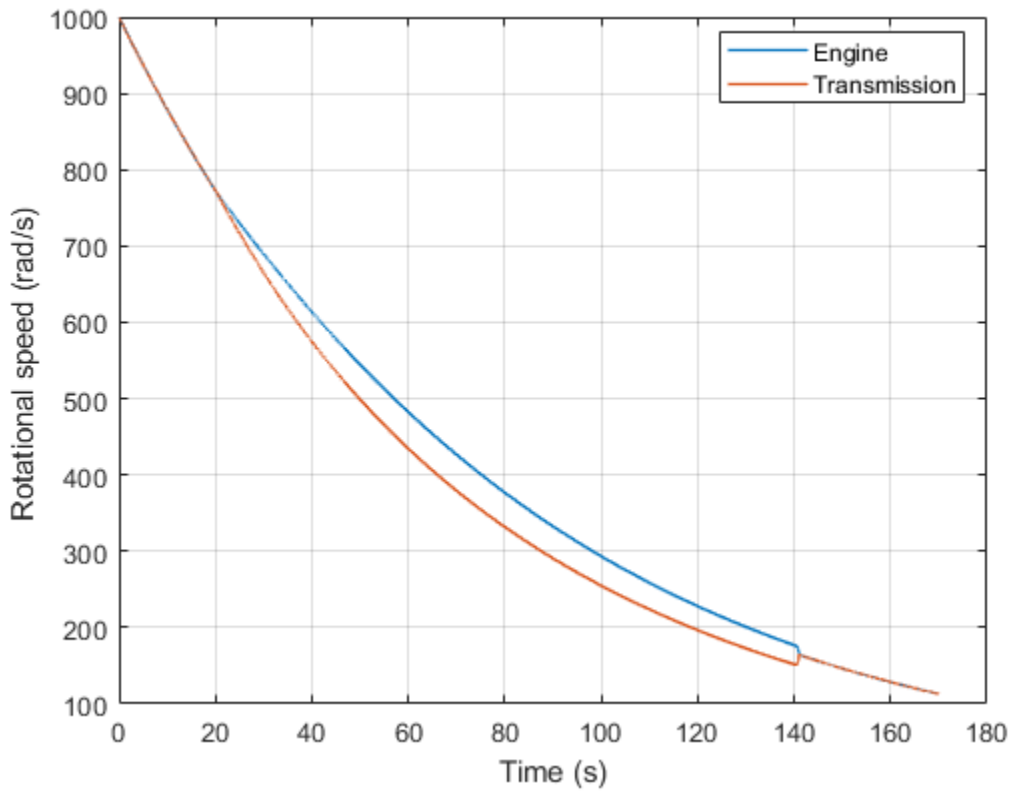
Automotive Clutch Subsystem

This subsystem shows how the clutch is modeled. The clutch consists of a lever mechanism, a diaphragm spring, a pressure plate, a set of clutch plates, and a hub. The lever transfers force to the diaphragm spring when a force is applied on the other end of the lever by the secondary cylinder. The applied force on the diaphragm spring causes the pressure plate to remove the pre-applied force on the clutch plate. The declutching takes place when the force on the clutch plate is removed through the removal of the force on the pressure plate by the diaphragm spring's action. No torque is transferred from the engine to the transmission after the declutching event.



Simulation Results from Simscape™ Logging

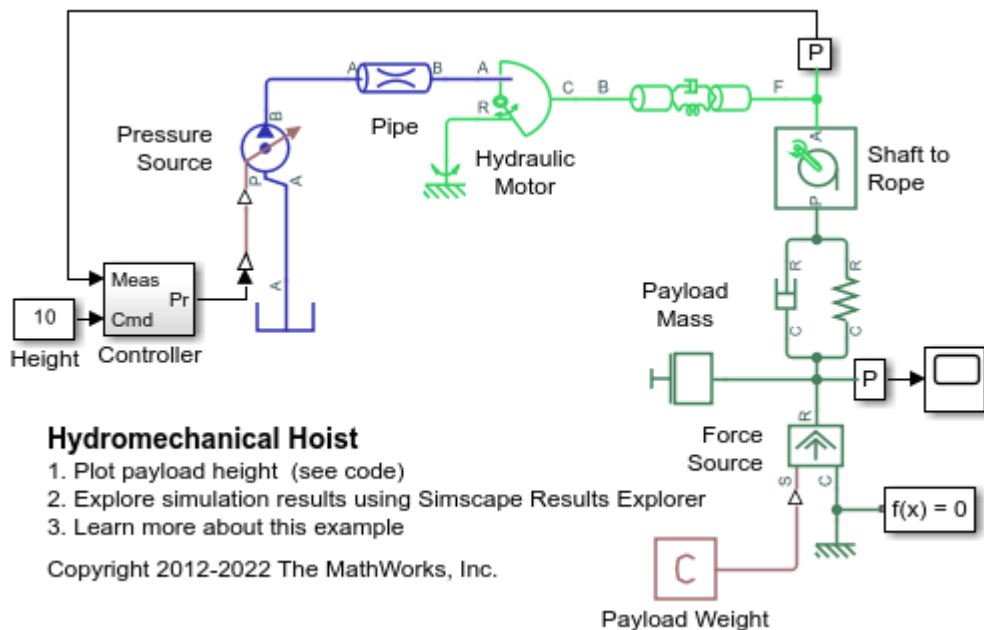
This model generates a plot of speeds versus time of the engine and the transmission systems during the declutching and clutch re-engaging scenario.



Hydromechanical Hoist

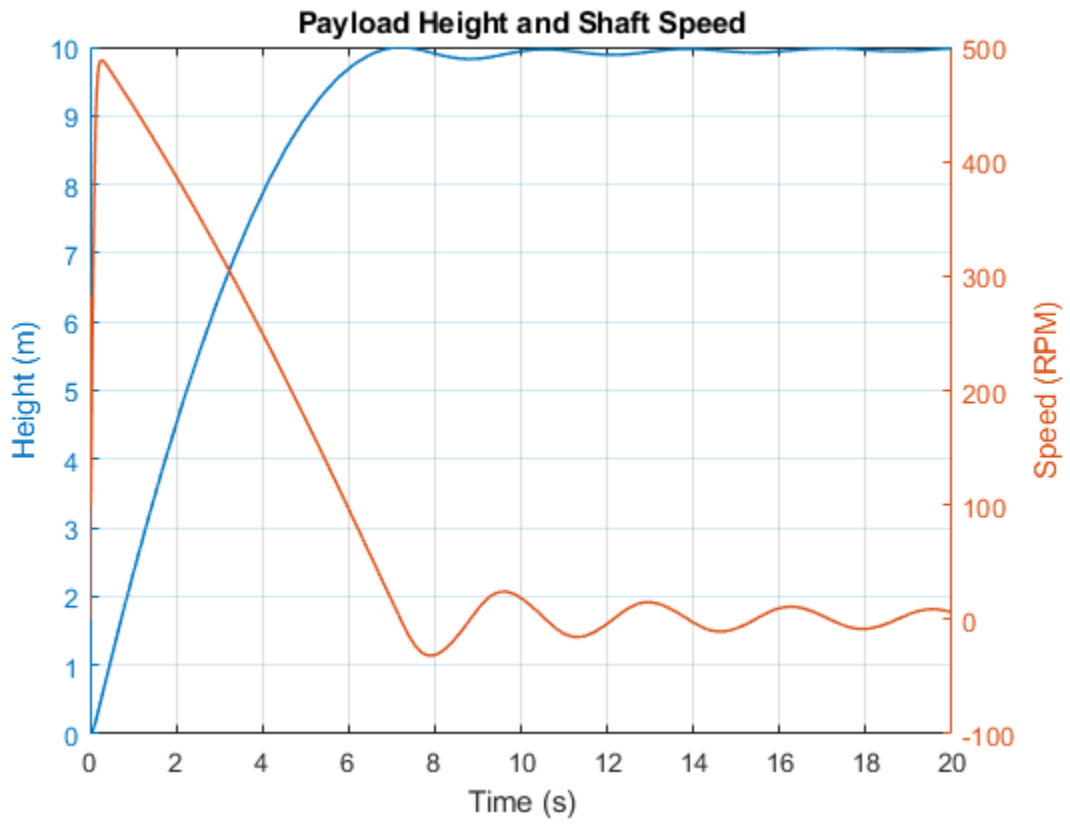
This example shows a hydromechanical hoist lifting a payload. An ideal hydraulic motor is driven by the pressure commanded by a controller. The controller estimates the current height of the payload by reading the angular velocity from the motor's flexible shaft. The rope is represented as a spring and damper, and the payload is modeled as a mass with gravitational force.

Model



Simulation Results from Simscape Logging

The plot below shows the height of a payload lifted by a hydromechanical hoist. The torque on the hoist shaft is also shown.

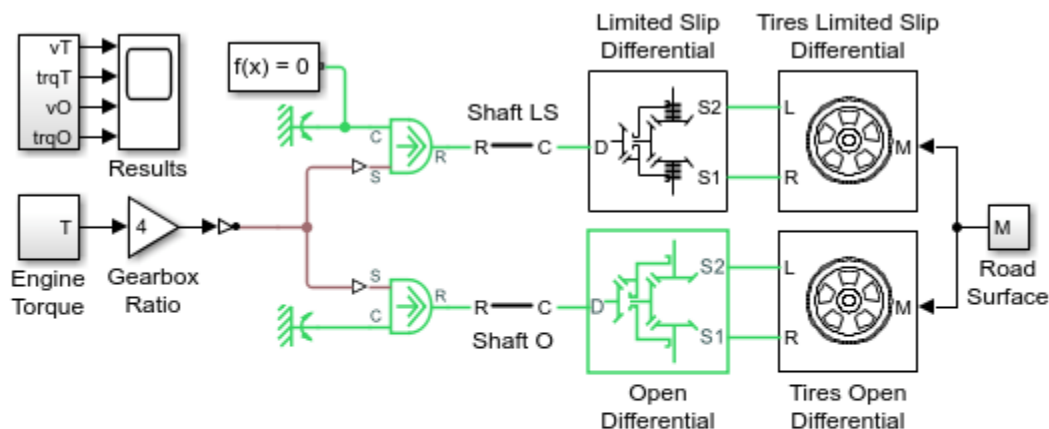


Limited Slip Differential with Clutches

This example shows a comparison between the behavior of an open differential and a limited slip differential with clutch packs. The limited slip differential is modeled using components from the Gears library and Clutches library in Simscape™ Driveline™. Wheel slip is limited by clutches that engage when the torque applied to the input of the differential exceeds a threshold. The clutches lock the differential so that the output shafts of the differential spin at the same speed.

The test surface includes an icy patch under the left wheel. This effect is introduced using the variable-friction coefficient variant of the Tire (Magic Formula) block. Comparing the two differentials on the same test surface shows that the limited slip differential locks up under the split surface road condition.

Model



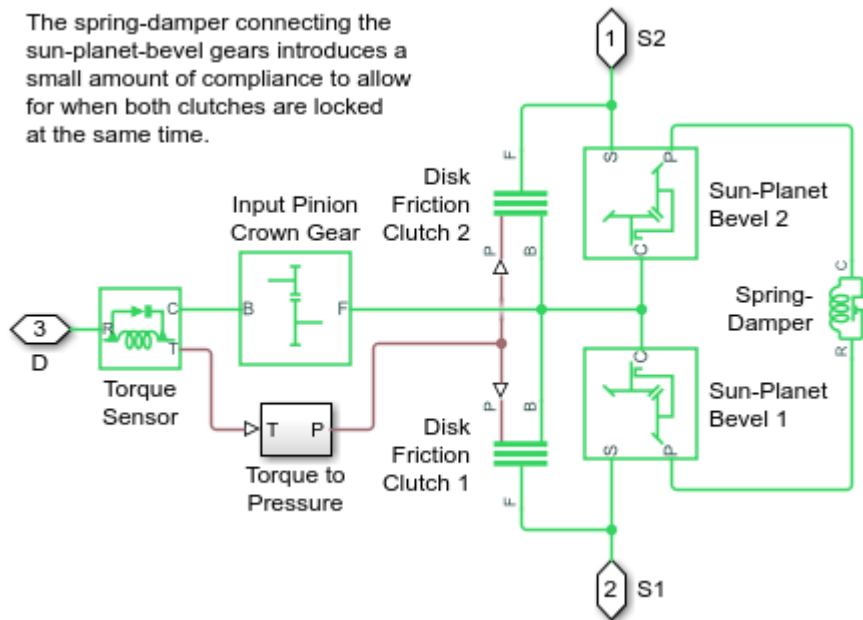
Limited Slip Differential with Clutches

1. Plot wheel speeds for both differentials (see code)
2. Plot shaft torques for both differentials (see code)
3. Explore simulation results using Simscape Results Explorer
4. Learn more about this example

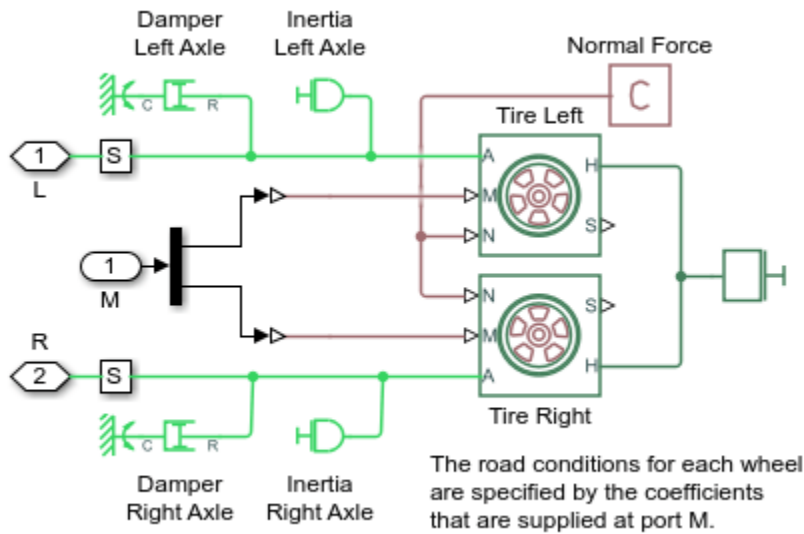
Copyright 2006-2022 The MathWorks, Inc.

Limited Slip Differential Subsystem

The spring-damper connecting the sun-planet-bevel gears introduces a small amount of compliance to allow for when both clutches are locked at the same time.

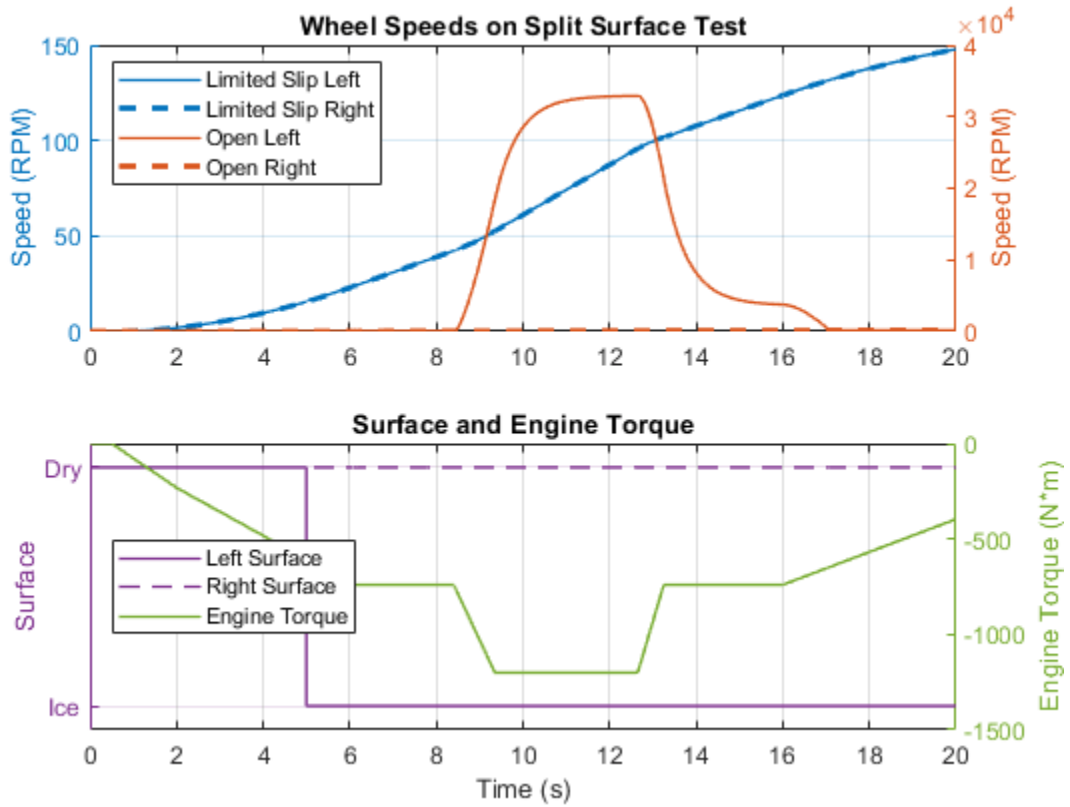


Tires Limited Slip Differential Subsystem

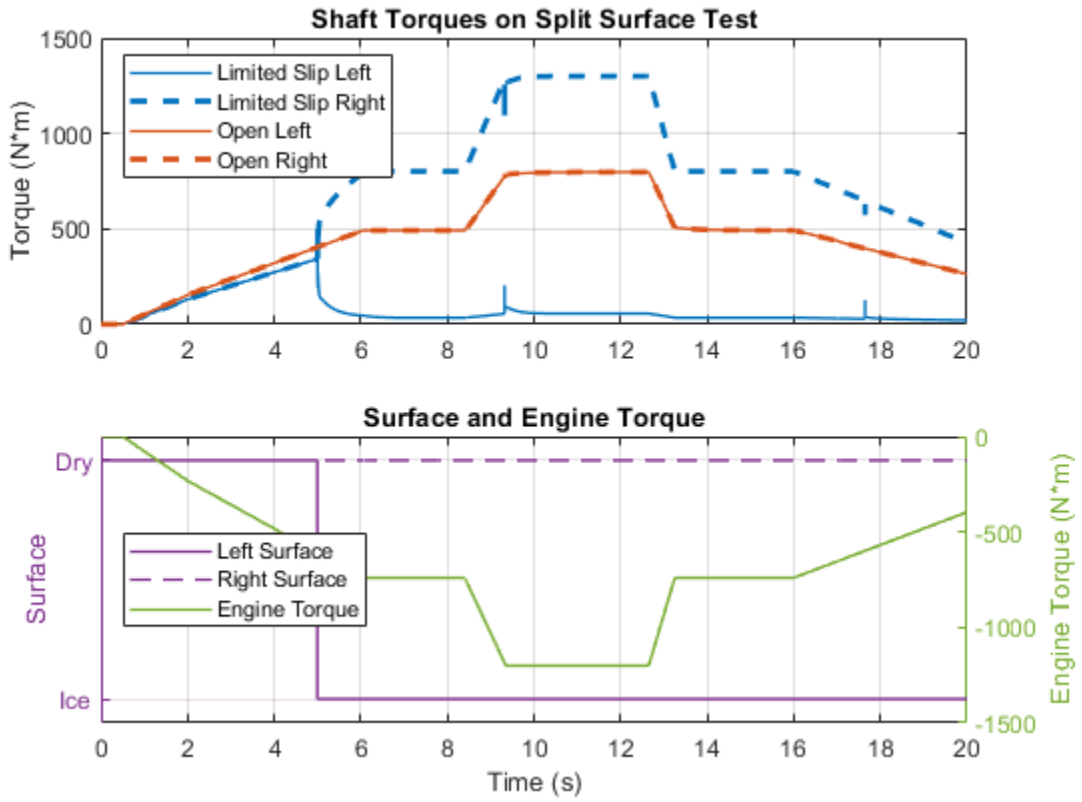


Simulation Results from Simscape Logging

The plot below shows the performance of an open and a limited slip differential on a split surface (ice and tarmac). The limited slip differential locks when the left wheel encounters the icy patch, and the left and right wheel speeds remain the same. The open differential does not lock and applies the same torque to both shafts. The result is that the left wheel loses traction on the icy patch and slips.



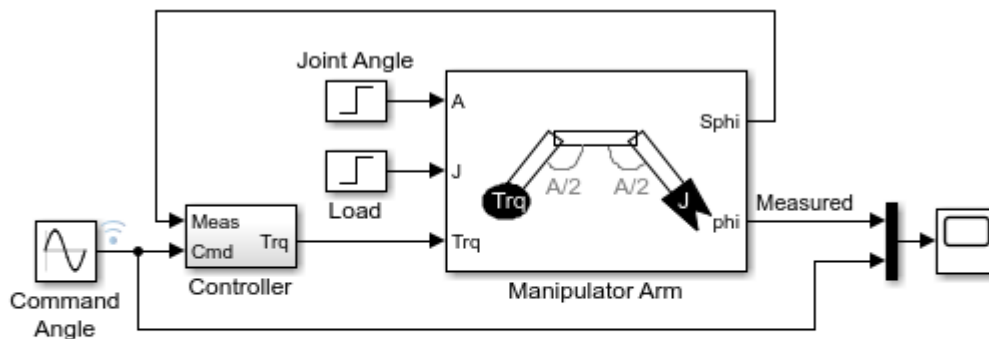
The plot below shows the performance of open and a limited slip differential on a split surface (ice and tarmac). The limited slip differential locks instantly when the left wheel encounters the icy patch. As a result, the wheels turn at the same speed and more torque is applied to the wheel on the high friction surface. The open differential does not lock. The same torque is applied to both wheels, resulting in the left wheel slipping extensively on the icy patch.



Manipulator with Unbalanced Load

This model shows a manipulator controlling the orientation of an end effector through an unbalanced arm. The motor is represented as a torque source using simple proportional control. The load on the end effector increases sharply when the end effector secures the load. Noise is introduced at each sensor to measure its effect on controller performance.

Model



Manipulator with Unbalanced Load

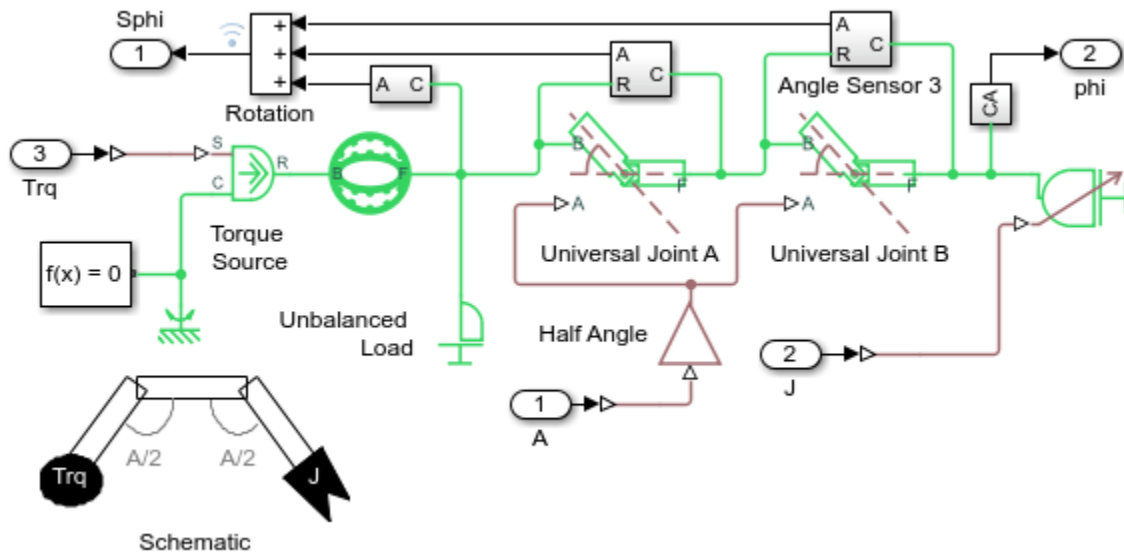
1. Plot arm rotation (see code)
2. Explore simulation results using Simscape Results Explorer
3. Learn more about this example

Copyright 2015-2022 The MathWorks, Inc.

Manipulator Arm Subsystem

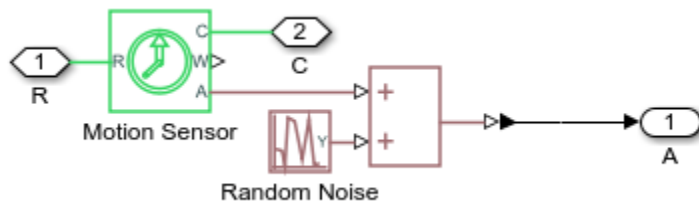
The manipulator arm consists of a harmonic drive, two universal joints assembled to form a double-cardan joint, and a load. An ideal torque source rotates the arm. The controller needs to deal with varying angles in the double-cardan joint, changes to the inertial load (Variable Inertia), and a shaft whose inertia varies with rotational angle (Unbalanced Load).

The control signal for the sensor is the sum of three sensors, each measuring the rotational angle of a portion of the arm.



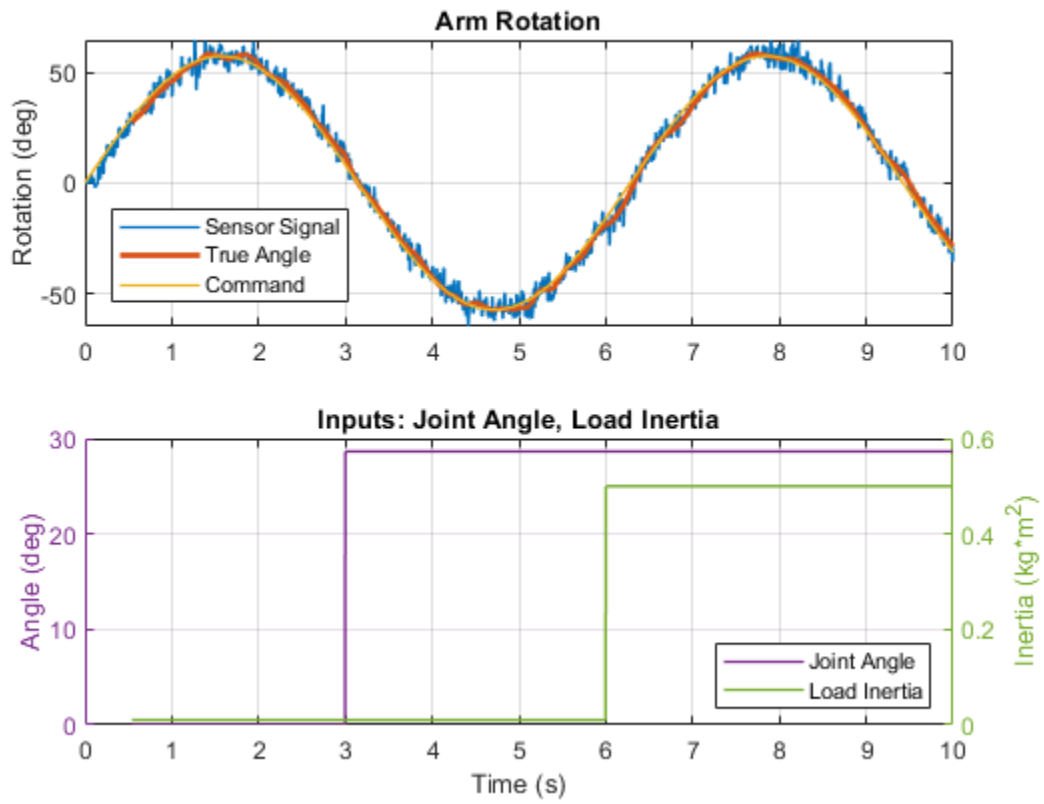
Angle Sensor 3 Subsystem

Random noise is added to each of the three sensors that provide the signal for the controller.



Simulation Results from Simscape Logging

The plot below shows the position of a rotating arm following a command signal. The system is exposed to two step disturbances: changes to the angles in a double-cardan joint and the inertial load at the end of the arm. Additional challenges for the controller include a shaft that has an inertia that varies with angle and noise added to each sensor signal.

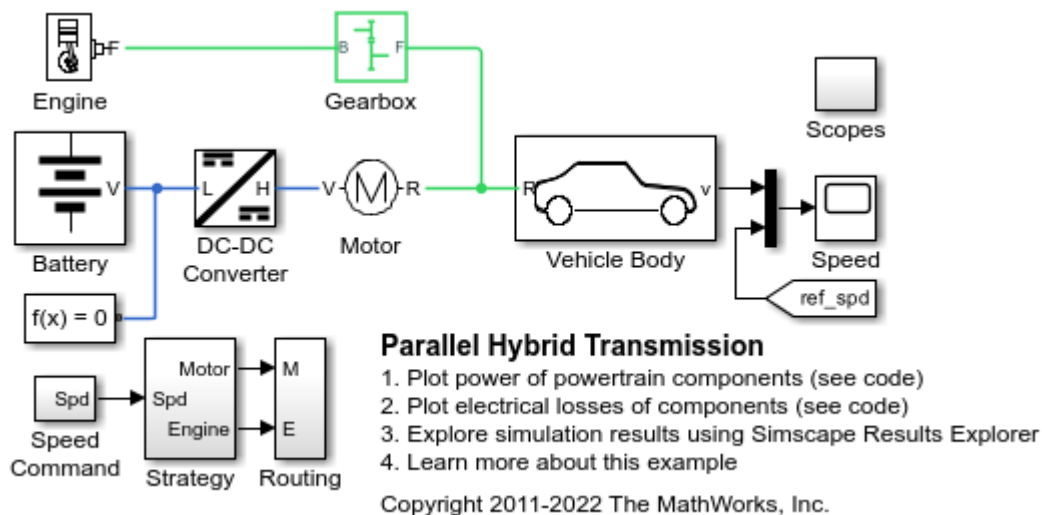


Parallel Hybrid Transmission

This example shows the basic architecture of a parallel hybrid transmission. Electrical power is applied in parallel with the combustion engine power. The electrical torque is applied at the wheel axle, but it could also be applied to the engine flywheel. In this test, the vehicle accelerates, maintains the faster speed, and then decelerates back to the original speed. The power management strategy uses just electrical power to effect the maneuver, the combustion engine only delivering the power required to maintain the original speed.

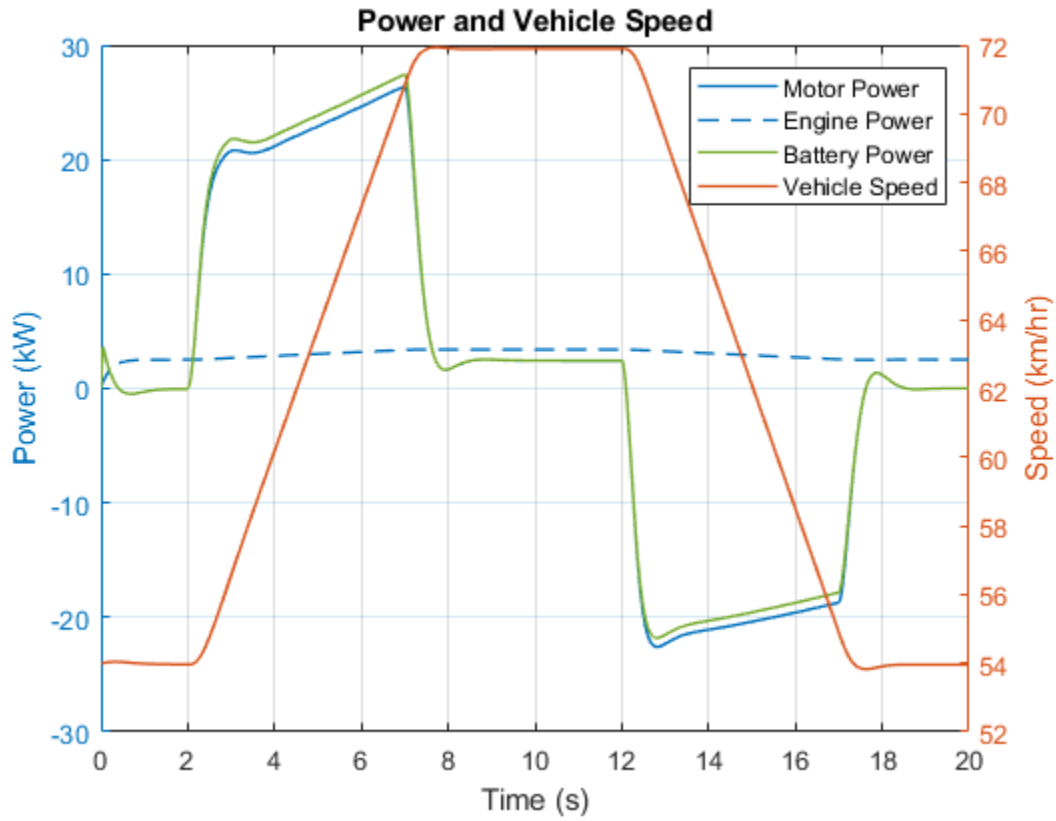
Losses for the motor, battery and gearbox are modeled. You can use this system-level model to gain understanding of system performance, and to support design of the power management strategy. The example can be directly compared with the power-split hybrid example `sdl_hybrid_power_split` and the series hybrid example `sdl_hybrid_series`.

Model

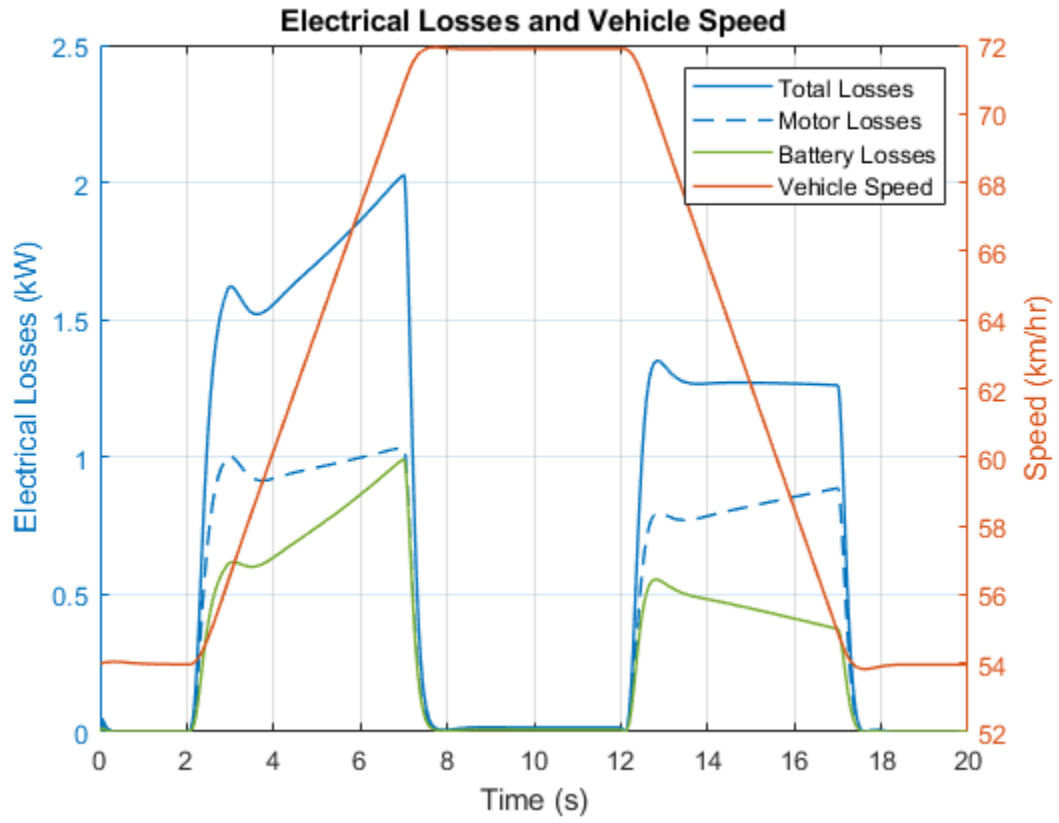


Simulation Results from Simscape Logging

The plot below shows the flow of power from the engine, motor, and battery as the vehicle accelerates and decelerates. The engine supplies just enough power to maintain the original speed. The acceleration and deceleration is done using electrical power. The motor draws enough power from the battery to accelerate the vehicle and then uses regenerative braking to feed that power back to the battery.



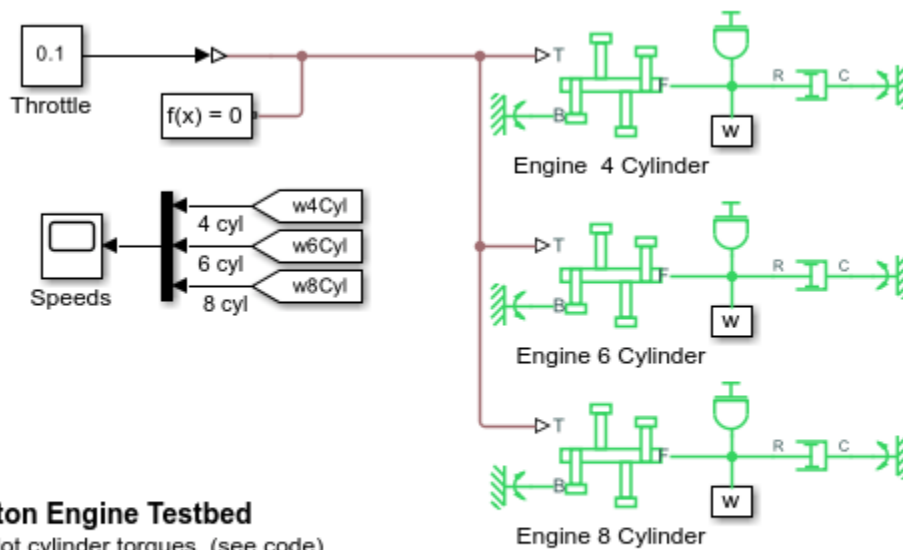
The plot below shows the electrical losses from the motor and battery as the vehicle accelerates and decelerates.



Piston Engine Testbed

This model shows the effect of varying the number of cylinders in a piston engine. Four, six, and eight cylinder engines are included with firing offsets evenly distributed about their four-stroke cycles. Piston pressures are normalized by the number of cylinders to emphasize the effect on output vibration.

Model



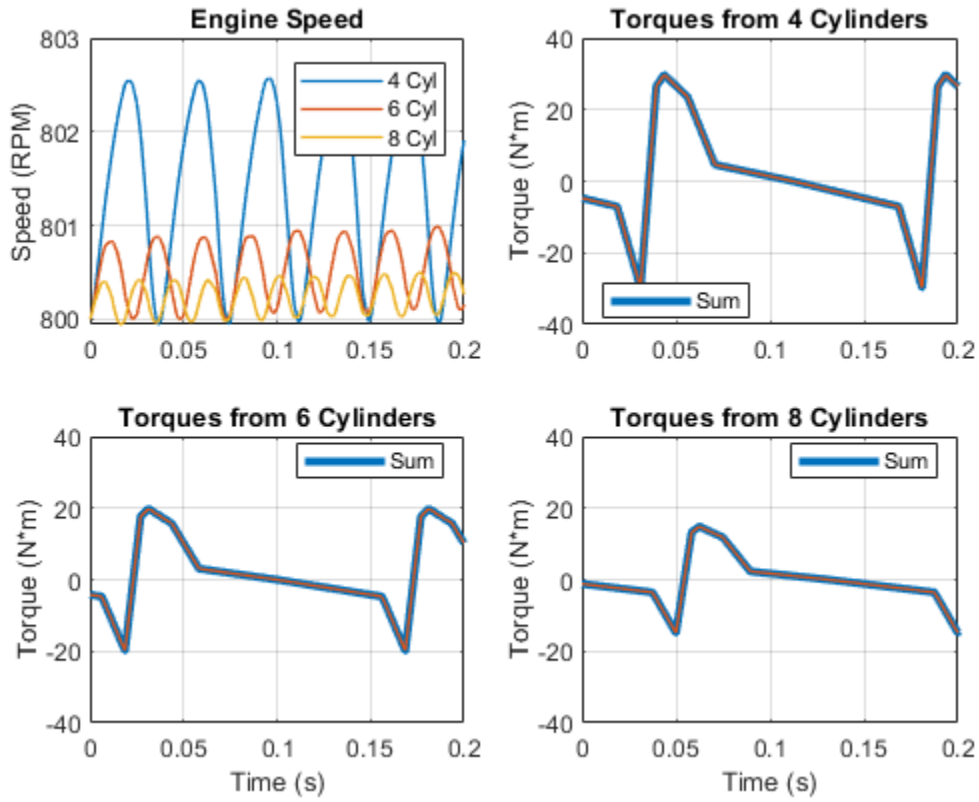
Piston Engine Testbed

1. Plot cylinder torques (see code)
2. Explore simulation results using Simscape Results Explorer
3. Learn more about this example

Copyright 2004-2022 The MathWorks, Inc.

Simulation Results from Simscape Logging

The plots below show engine speeds and cylinder torques for 4, 6, and 8 cylinder engines. All engines are acting against the same viscous friction load. The sum of torques produced by all cylinders in each engine is also plotted to illustrate the effect of more cylinders on engine torque.

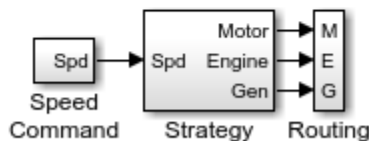
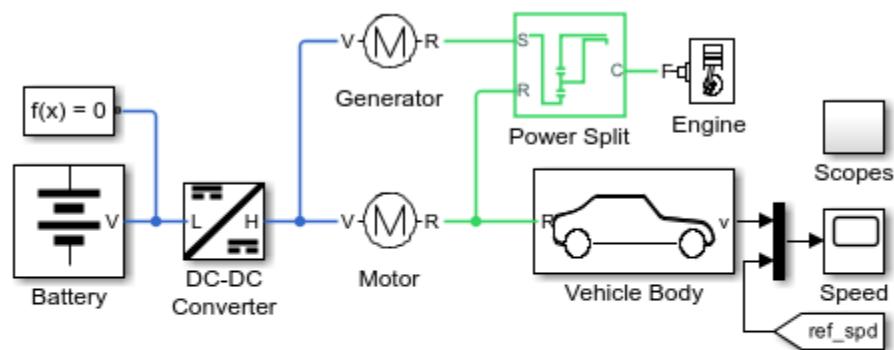


Power-Split Hybrid Transmission

This example shows the basic architecture of a power-split hybrid transmission. The planetary gear, along with the motor and generator, acts like a variable ratio gear. In this test, the vehicle accelerates, maintains the faster speed, and then decelerates back to the original speed. The power management strategy uses just stored electrical power to effect the maneuver, the combustion engine only delivering the power required to maintain the original speed.

Losses for the motor, generator, battery and planetary gear are modeled. You can use this system-level model to gain understanding of system performance, and to support design of the power management strategy. The example can be directly compared with the parallel hybrid example `sdl_hybrid_parallel` and the series hybrid example `sdl_hybrid_series`.

Model



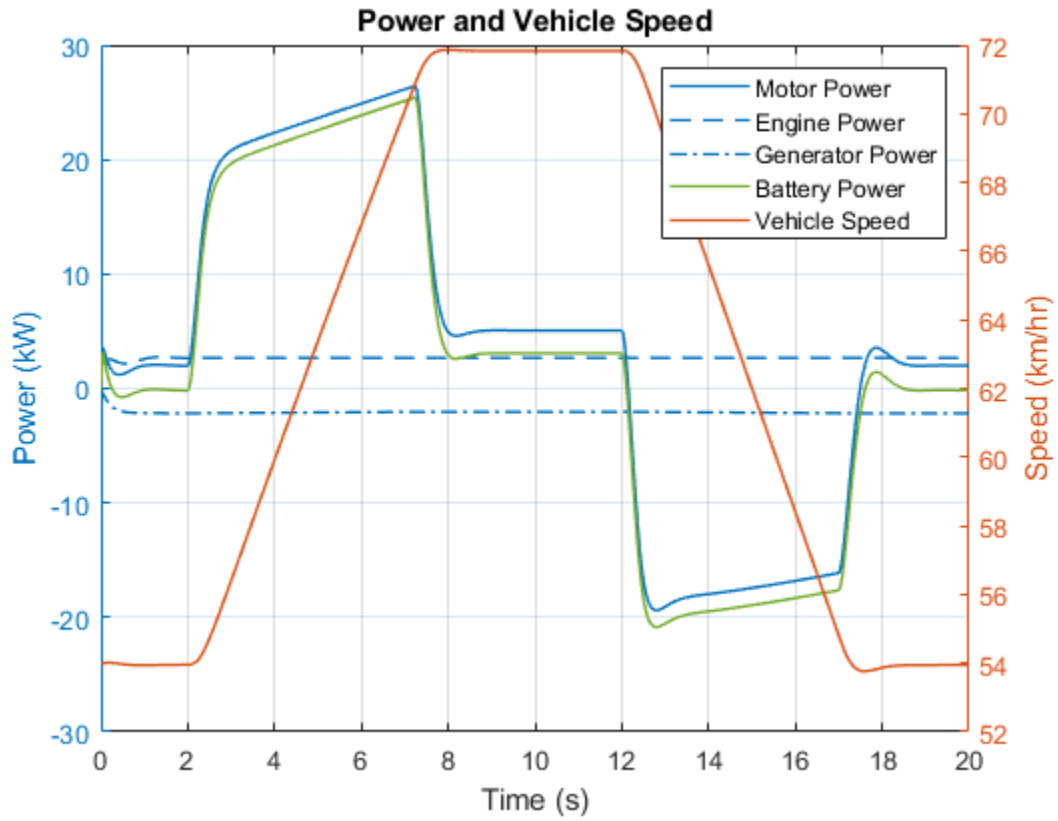
Power-Split Hybrid Transmission

1. Plot power of powertrain components (see code)
2. Plot electrical losses of components (see code)
3. Explore simulation results using Simscape Results Explorer
4. Learn more about this example

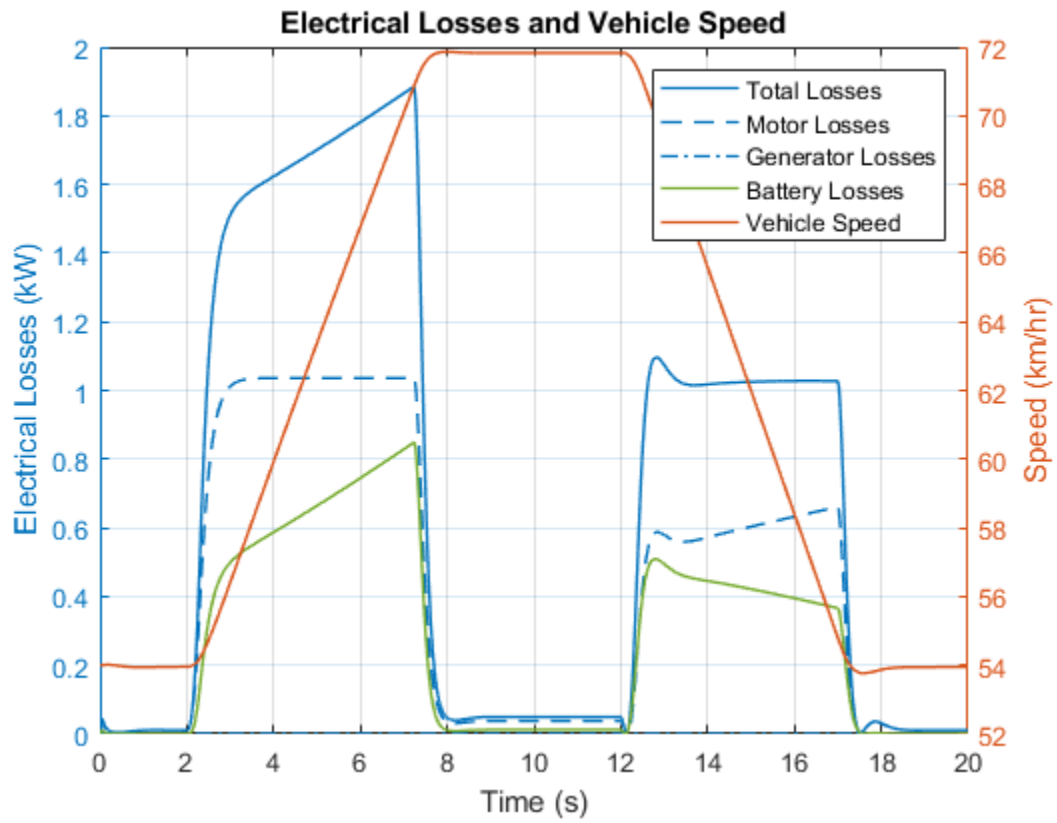
Copyright 2011-2022 The MathWorks, Inc.

Simulation Results from Simscape Logging

The plot below shows the flow of power from the engine, motor, and generator as the vehicle accelerates and decelerates. The generator supplies the DC network with a constant flow of power drawn from the engine. The motor draws power from the battery to accelerate the vehicle and then uses regenerative braking to feed that power back to the battery.



The plot below shows the electrical losses from the motor, generator, and battery as the vehicle accelerates and decelerates. The largest losses come from the motor and the battery.

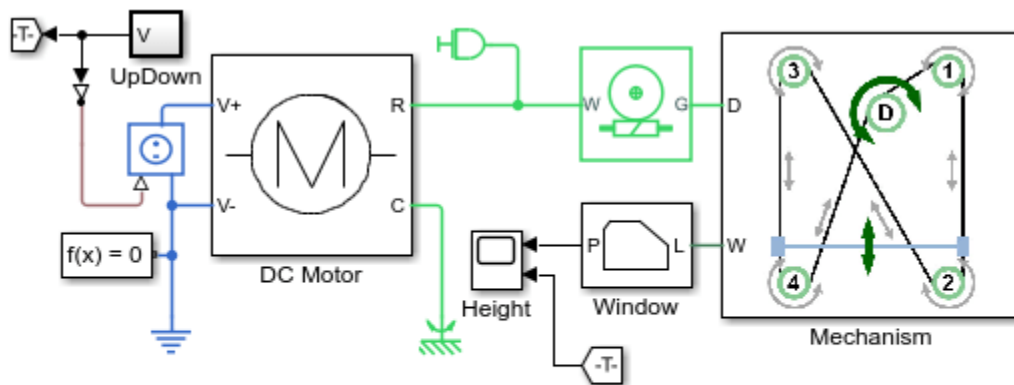


Power Window System

This example shows a motor-driven power window system. A DC motor drives the power window mechanism via a self-locking worm gear with the ratio 1 : 50. The power window mechanism consists of a cable drum and four pulleys all connected by a cable. The window is attached to the cable at two points by the lift plate. This ensures both sides of the window move at the same speed and in the same direction, keeping the window level. The model also includes viscous friction in the guide rails.

Pre-tensioned cable segments are added between the pulleys. Each segment is simulated as a damper and preloaded spring using a Simscape™ Driveline™ Rope block. When modeling systems using pulleys and drums, it is important to remember that slack cables and belts cannot transmit force, hence the need for pre-tensioned cables and belts in the system.

Model



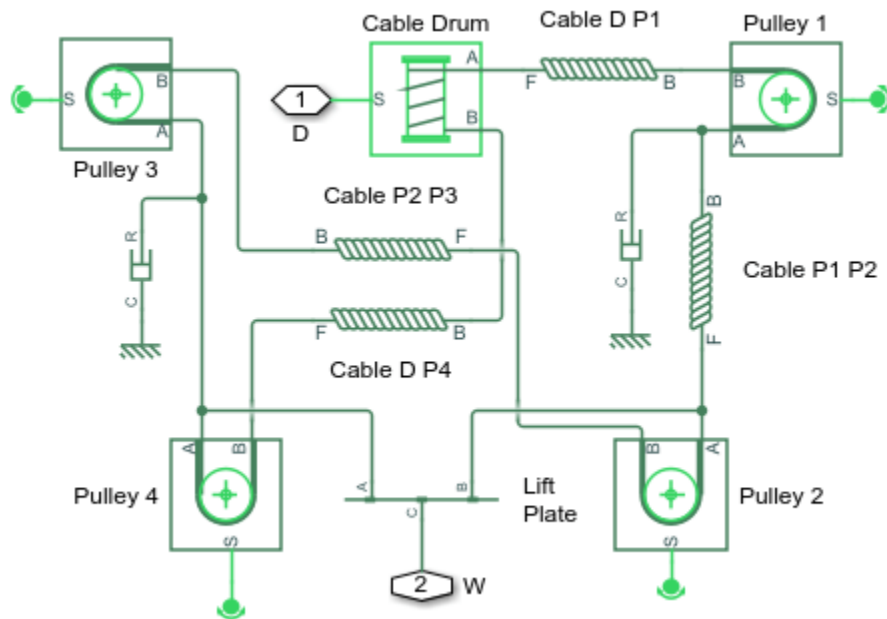
Power Window System

1. Plot motor torque and cable drum speed (see code)
2. Explore simulation results using Simscape Results Explorer
3. Learn more about this example

Copyright 2008-2022 The MathWorks, Inc.

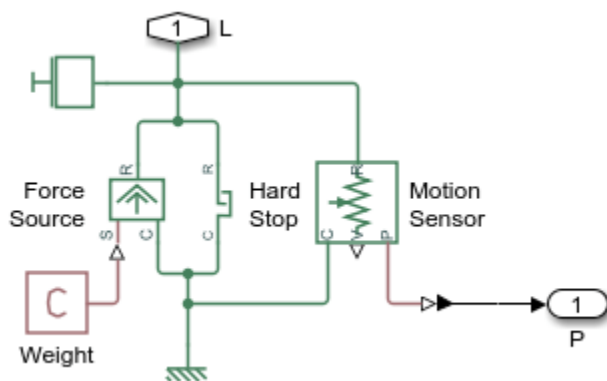
Mechanism Subsystem

The four rollers in the system are modeled using Belt Pulley blocks. They are connected via Rope blocks, which act as preloaded springs and dampers to ensure the cable is kept in tension. The power from the motor is transmitted to the cable assembly via the cable drum. The lift plate is modeled using a Lever block, and represents the two connection points of the window to the cable. The viscous friction in the guide rails is modeled using translational dampers.



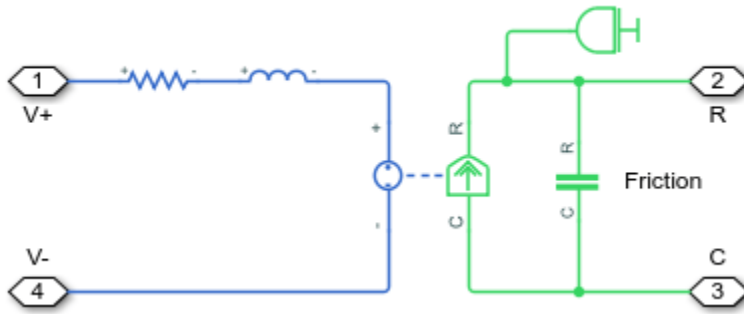
Window Subsystem

The window is modeled as a mass subjected to a gravitational force. The Force Source is necessary to include the gravitational force. The hard stop represents the limit of travel in the guide rails.



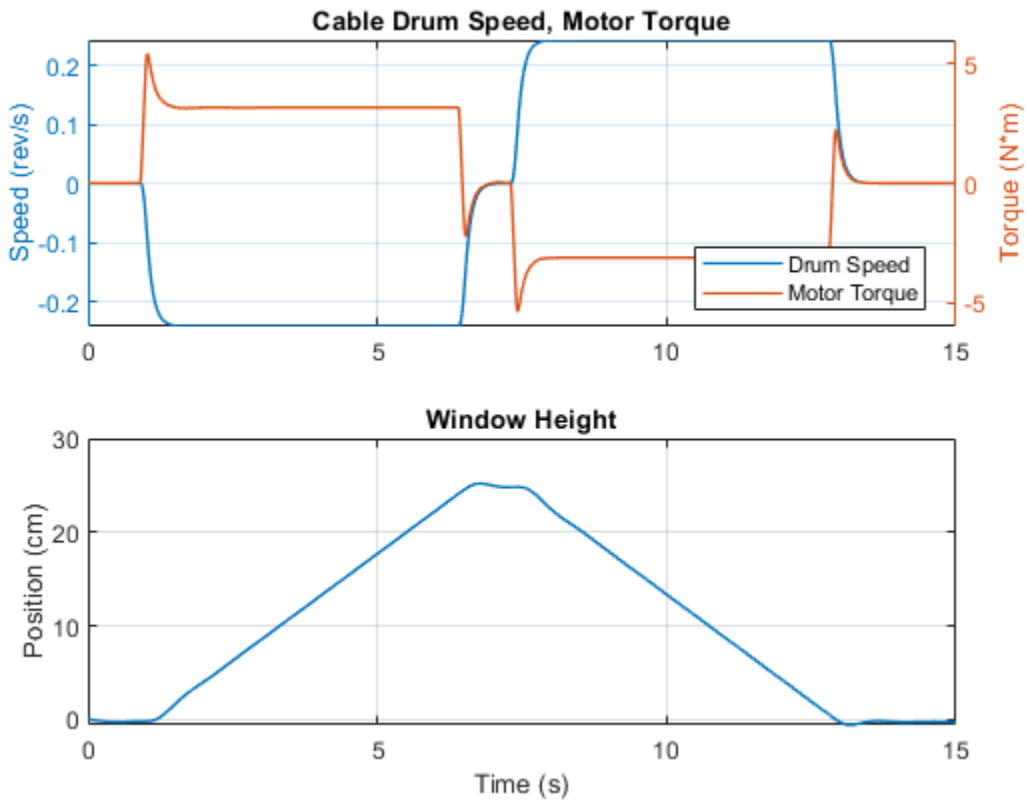
DC Motor Subsystem

A simple DC motor is modeled using electrical and mechanical components.



Simulation Results from Simscape Logging

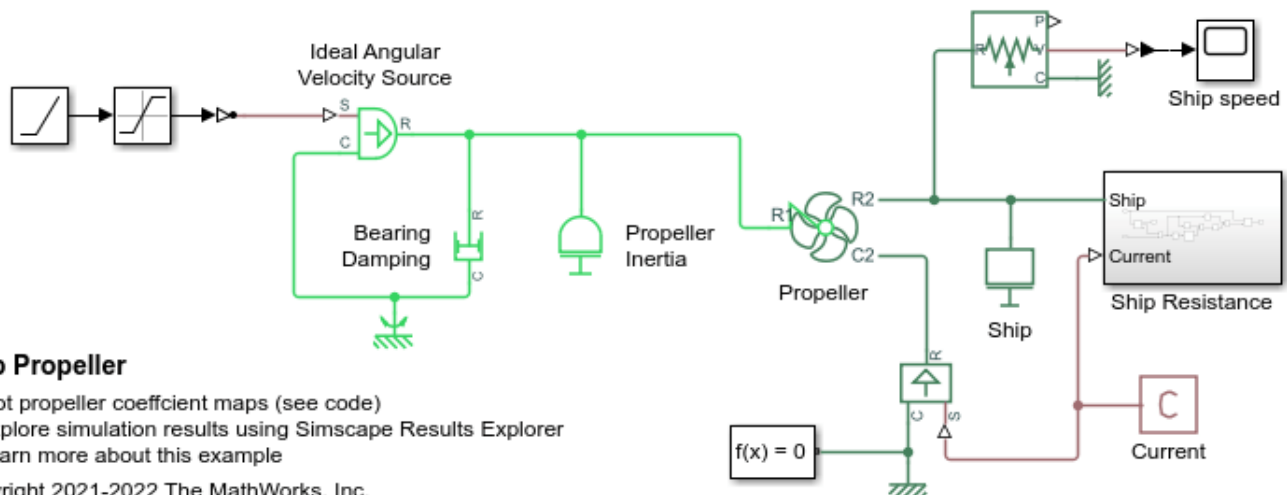
The plot below shows the torque required to raise and lower the window. The rotational speed of the cable drum is also shown.



Ship Propeller

This example shows a marine propeller represented by a Propeller block from the Simscape™ Driveline™ library that propels a ship. The propeller starts from rest while the current pushes the ship forward. The ship accelerates as the Ideal Angular Velocity Source starts up the propeller. After 500 seconds, the propeller spins at a constant speed, and the ship approaches a constant speed. Ship motion is resisted by a drag force that increases quadratically with ship speed.

Model

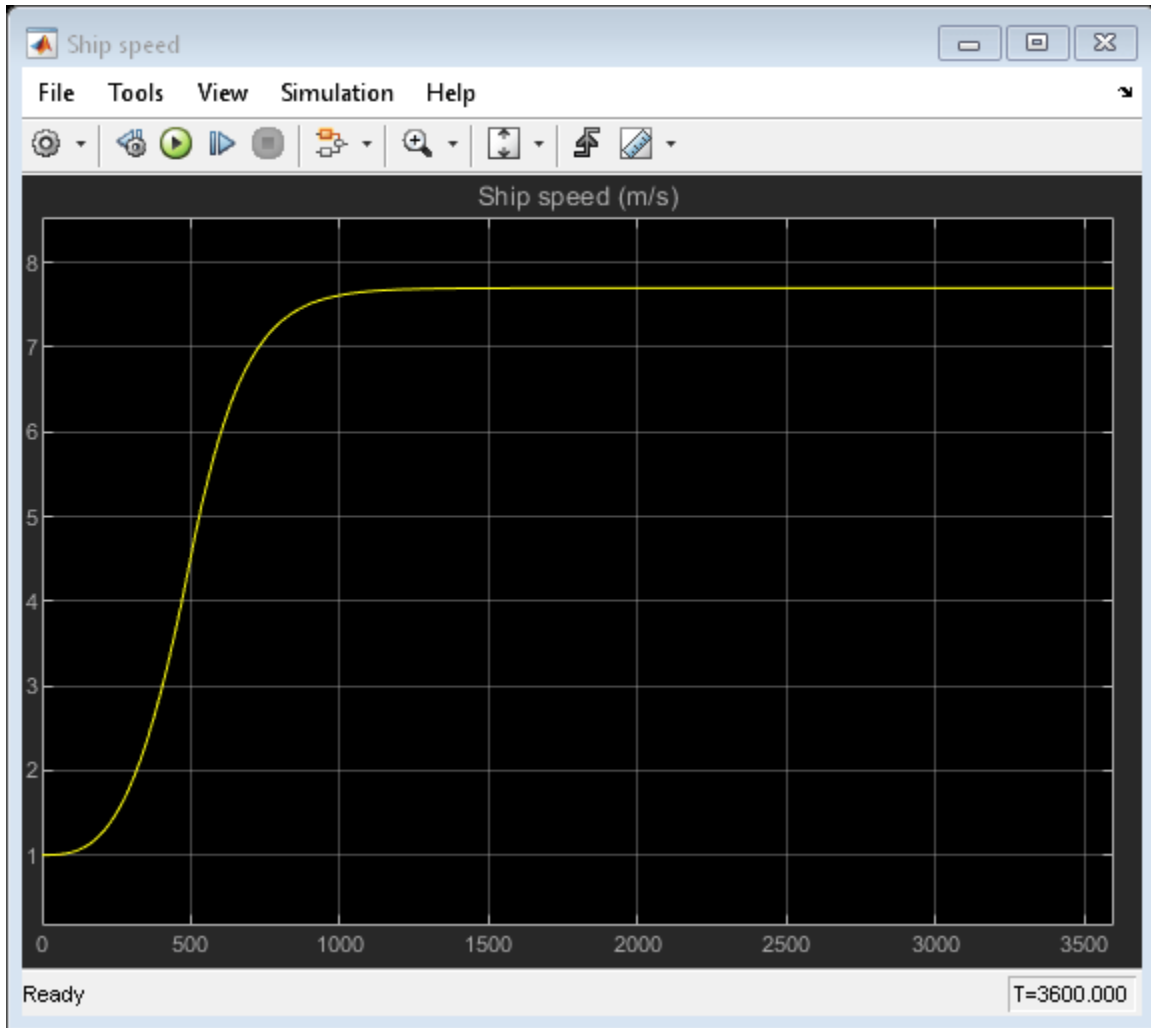


Ship Propeller

1. Plot propeller coefficient maps (see code)
2. Explore simulation results using Simscape Results Explorer
3. Learn more about this example

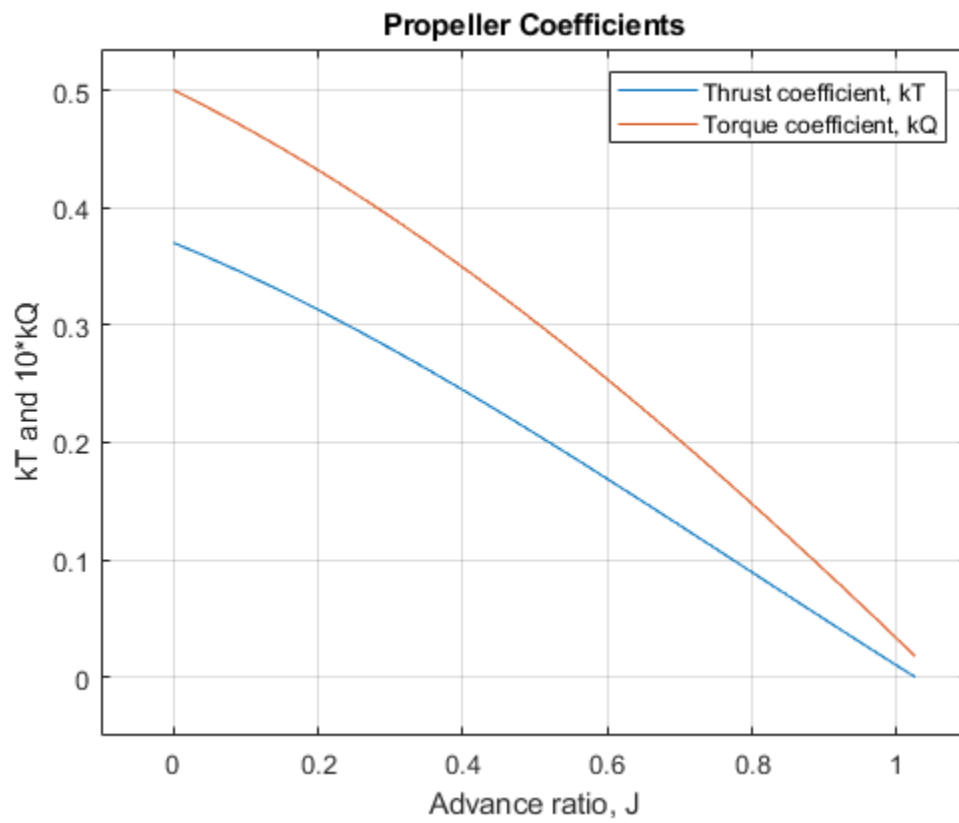
Copyright 2021-2022 The MathWorks, Inc.

Simulation Results from Scope



Thrust and Torque Curves based on Block Parameters

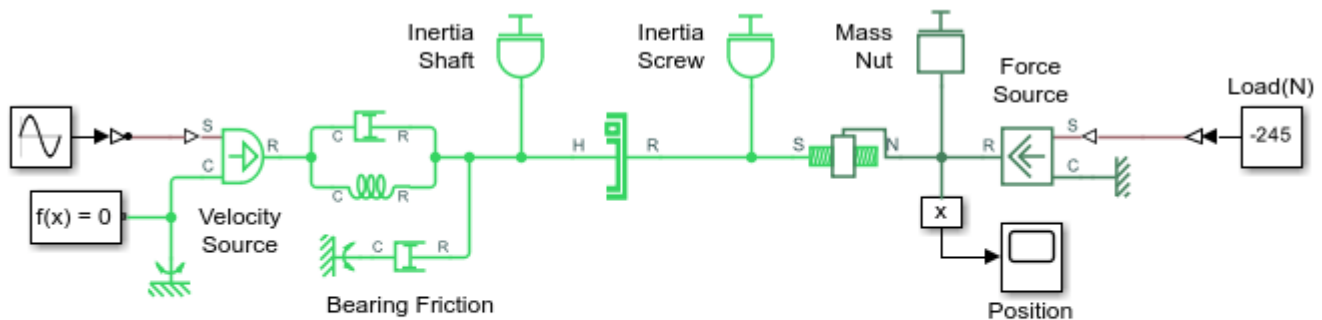
This plot shows the propeller thrust and torque coefficients versus advance ratio defined in the block property inspector.



Ratchet Leadscrew Mechanism

This example shows a stepping mechanism constructed from a self-locking leadscrew and a unidirectional clutch. The input shaft of the unidirectional clutch oscillates with velocity amplitude 2 rad/s and frequency 0.5 rad/s. The unidirectional clutch drives the load via the leadscrew. The lead angle of the screw is small enough to ensure self-locking operation due to geometry and thread friction. As a result, the load maintains its position while the clutch input shaft rotates in the opposite direction and the load moves in incremental steps of about 25 mm.

Model



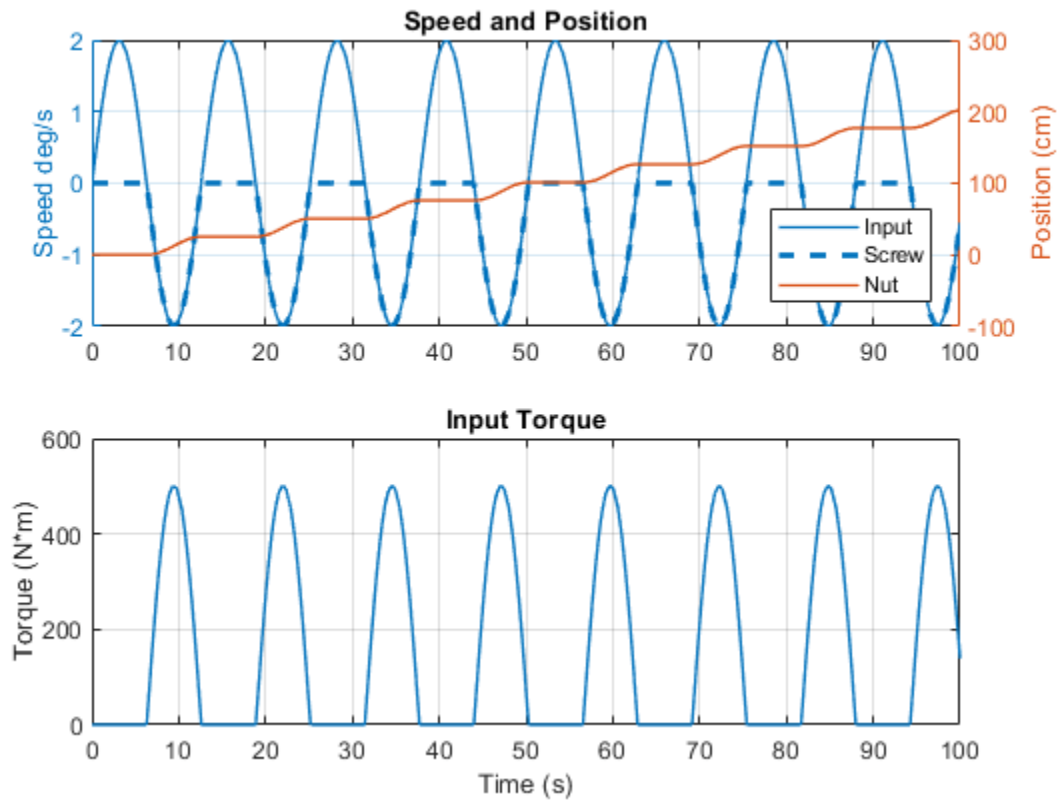
Ratchet Leadscrew Mechanism

1. Plot motion of mechanism (see code)
2. Explore simulation results using Simscape Results Explorer
3. Learn more about this example

Copyright 2006-2022 The MathWorks, Inc.

Simulation Results from Simscape Logging

The plot below shows the speed of the input shaft, leadscrew shaft, and the motion of the nut. The self-locking behavior of the leadscrew plus the unidirectional clutch enables the cyclical input motion to move the nut in a single direction.

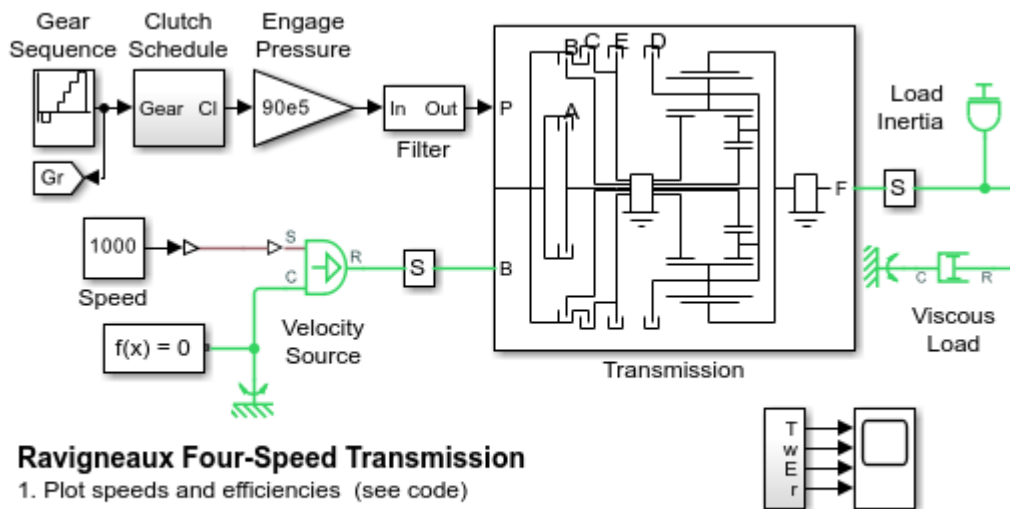


Ravigneaux Four-Speed Transmission

This model shows a test harness for a Ravigneaux 4-speed transmission. The subsystem uses the Ravigneaux Gear block and five friction clutches to implement four forward ratios plus reverse. Meshing losses are enabled in the Ravigneaux gear. The clutch states required to implement the different ratios can be viewed by opening the Clutch Schedule block.

The test executed in this model drives the input shaft to the transmission at a constant speed while cycling through all gears, including neutral. The shaft speeds and power are measured so that the gear ratios and overall efficiency can be calculated.

Model

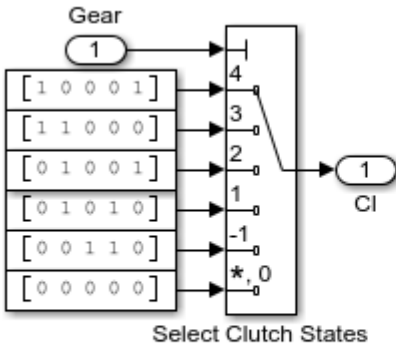


Ravigneaux Four-Speed Transmission

1. Plot speeds and efficiencies (see code)
2. Explore simulation results using Simscape Results Explorer
3. Learn more about this example

Copyright 2007-2022 The MathWorks, Inc.

Clutch Schedule Subsystem

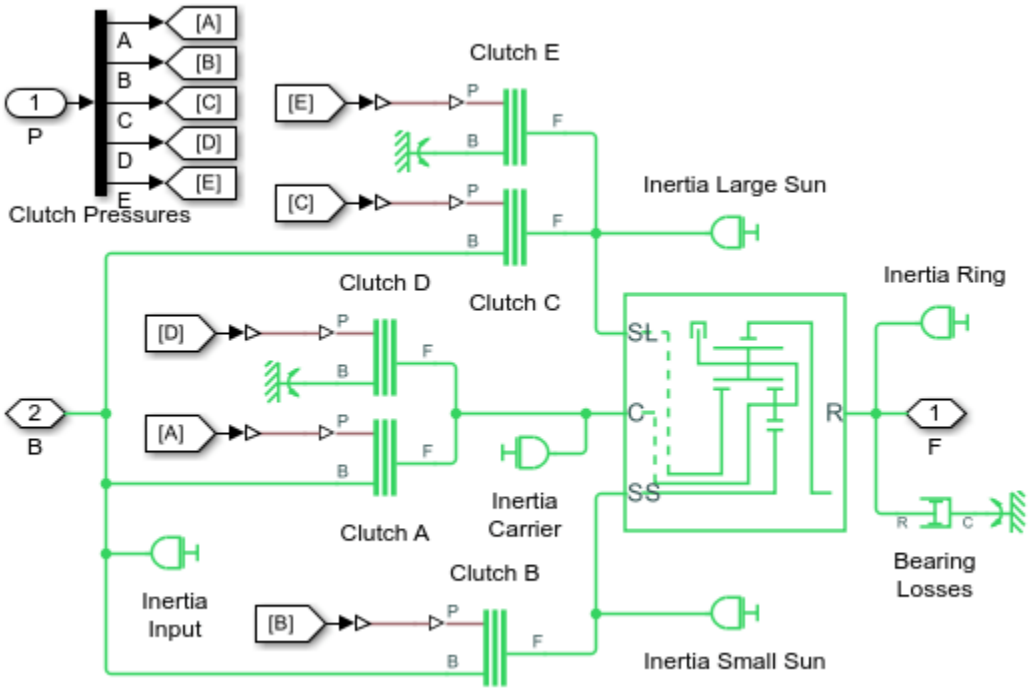


Clutch Schedule

Gear	A	B	C	D	E	Ratio
R	0	0	1	1	0	-g1
1	0	1	0	1	0	g2
2	0	1	0	0	1	$(g1+g2)/(1+g1)$
3	1	1	0	0	0	1
4	1	0	0	0	1	$g1/(1+g1)$

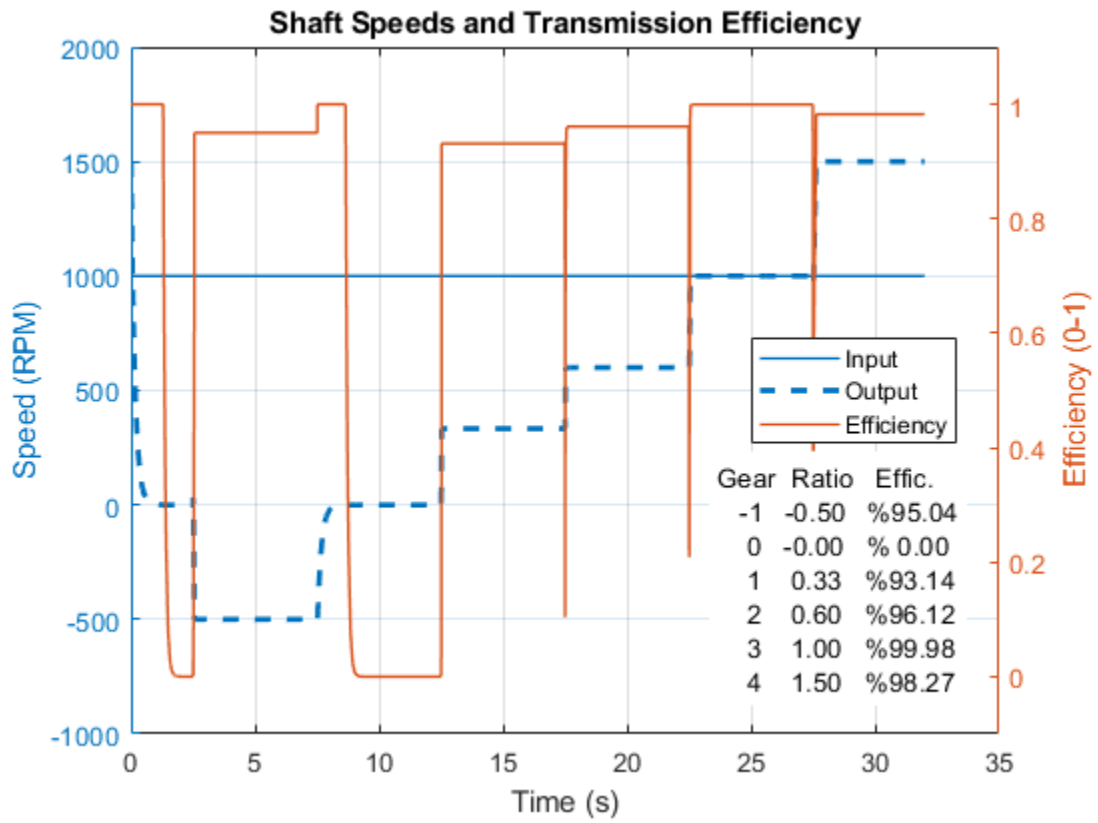
0 - Disengaged, 1 - Engaged
 g1: Ravigneaux ring (R)/large sun (SL) gear ratio
 g2: Ravigneaux ring (R)/small sun (SS) gear ratio

Transmission Subsystem



Simulation Results from Simscape Logging

The plot below shows shaft speeds and overall transmission efficiency as the transmission is cycled through each of its gears. The input shaft is turned at a constant speed. The table in the lower right shows the steady-state values for gear ratio and efficiency for each gear.

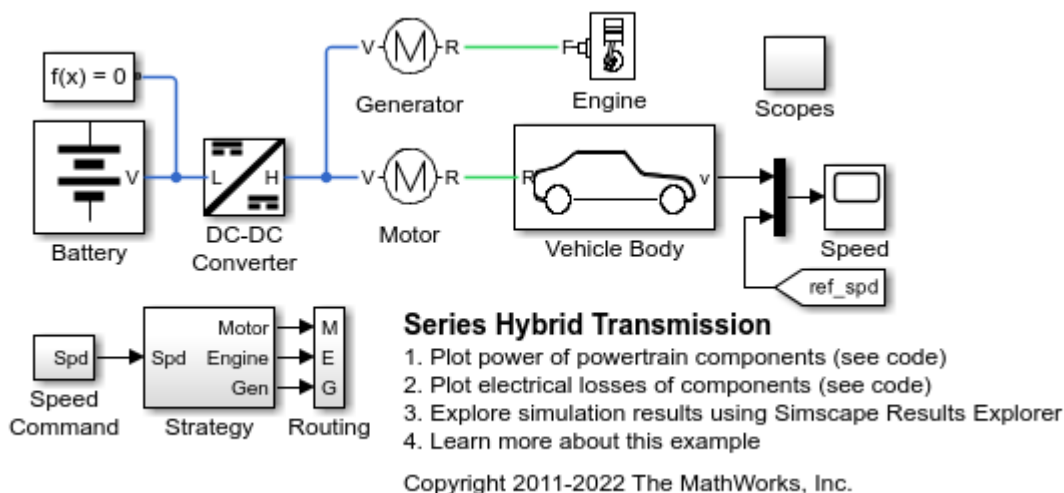


Series Hybrid Transmission

This example shows the basic architecture of a series hybrid transmission. All mechanical power from the engine is converted to electrical power via the generator. In this test, the vehicle accelerates, maintains the faster speed, and then decelerates back to the original speed. The power management strategy uses just stored electrical power to effect the maneuver, the combustion engine only delivering the power required to maintain the original speed.

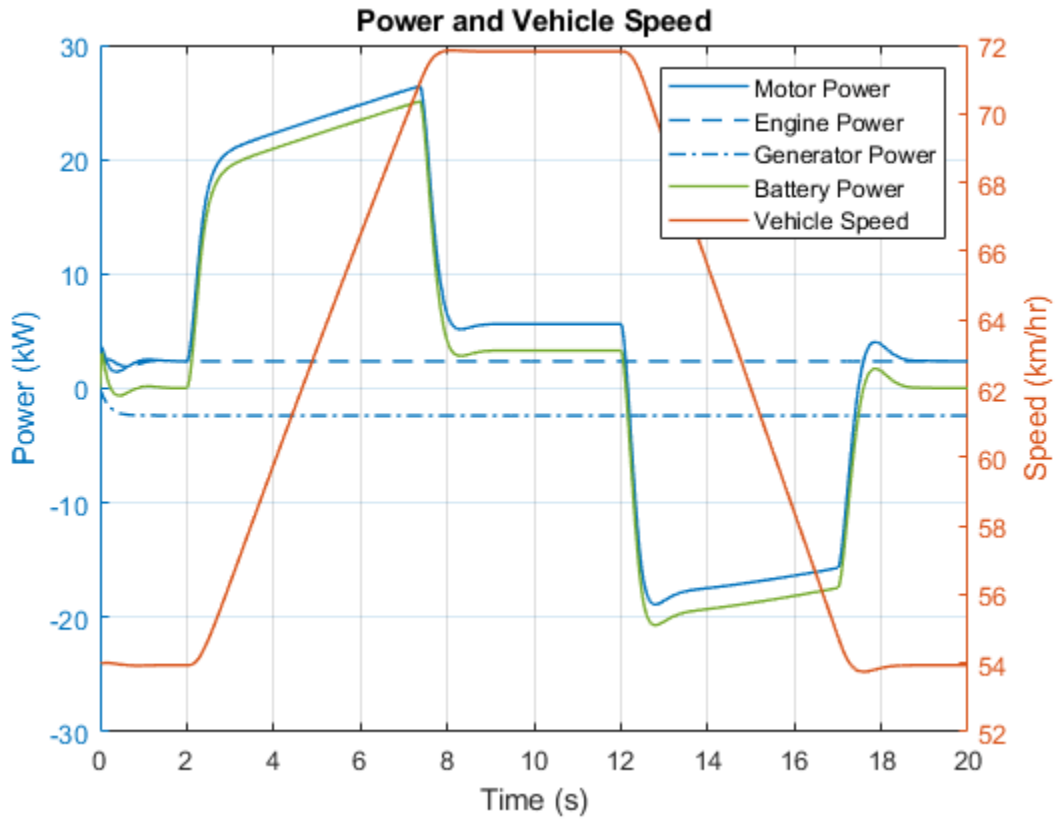
Losses for the motor, generator, and battery are modeled. You can use this system-level model to gain understanding of system performance, and to support design of the power management strategy. The example can be directly compared with the parallel hybrid example `sdl_hybrid_parallel` and the power-split hybrid example `sdl_hybrid_power_split`.

Model

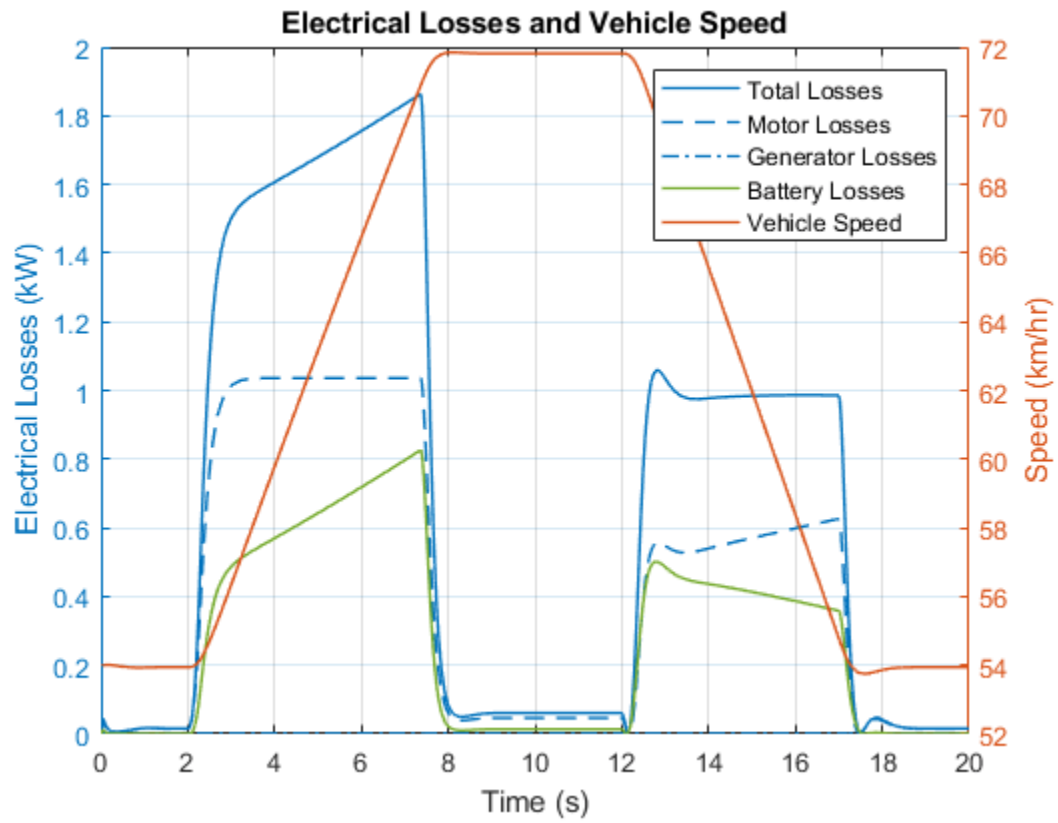


Simulation Results from Simscape Logging

The plot below shows the flow of power from the engine, motor, and generator as the vehicle accelerates and decelerates. The generator supplies the DC network with a constant flow of power drawn from the engine. The motor draws power from the battery to accelerate the vehicle and then uses regenerative braking to feed that power back to the battery.



The plot below shows the electrical losses from the motor, generator, and battery as the vehicle accelerates and decelerates. The largest losses come from the motor and the battery.



Sheet Metal Feeder

This example shows a feeding mechanism of a sheet metal cutter. Two slitted rolls are driven by a torque motor through two mechanical drivetrains. Each drivetrain consists of a spur gear train, worm gear, and a chain drive. The sprocket of the chain drive is connected to its respective slitted roll. The interaction between the rolls and the sheet metal is simulated by the Loaded Contact Translational Friction blocks.

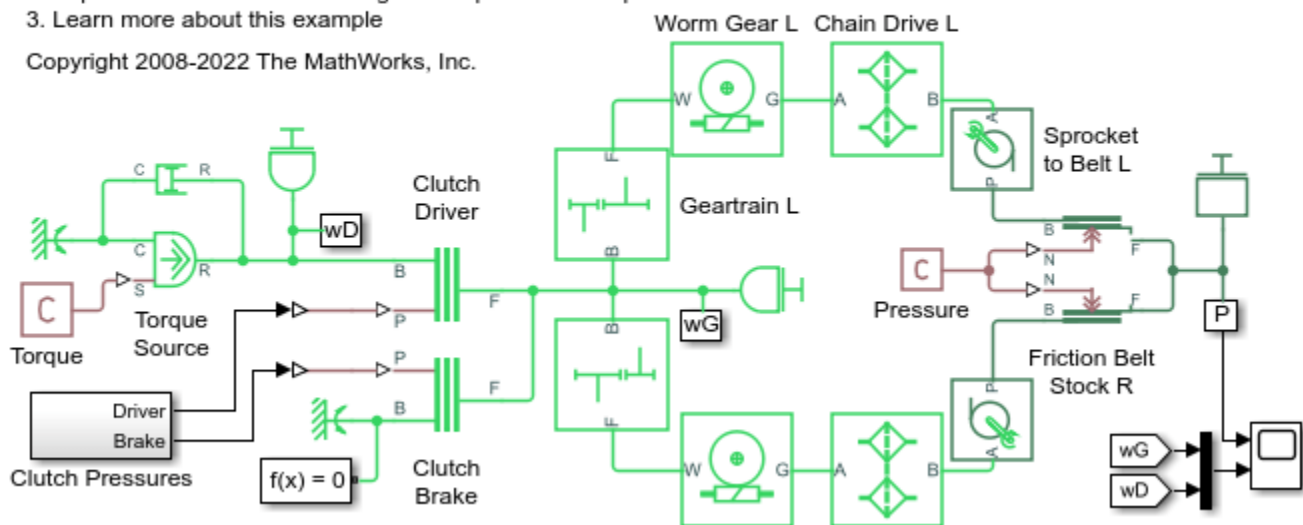
The driver uses a torque motor power unit and two disk friction clutches. The drive clutch periodically connects the flywheel to the transmissions to advance the stock. The brake clutch engages during idle intervals to fix the rollers in intermediate positions.

Model

Sheet Metal Feeder

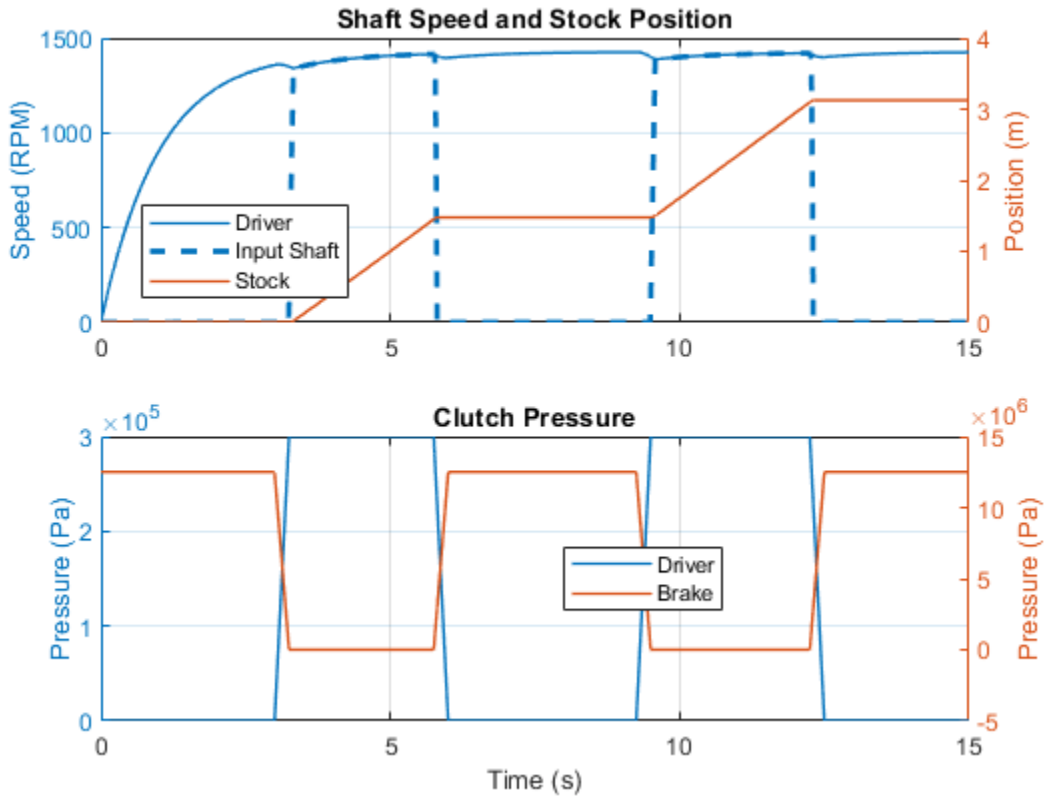
1. Plot speed and position (see code)
2. Explore simulation results using Simscape Results Explorer
3. Learn more about this example

Copyright 2008-2022 The MathWorks, Inc.



Simulation Results from Simscape Logging

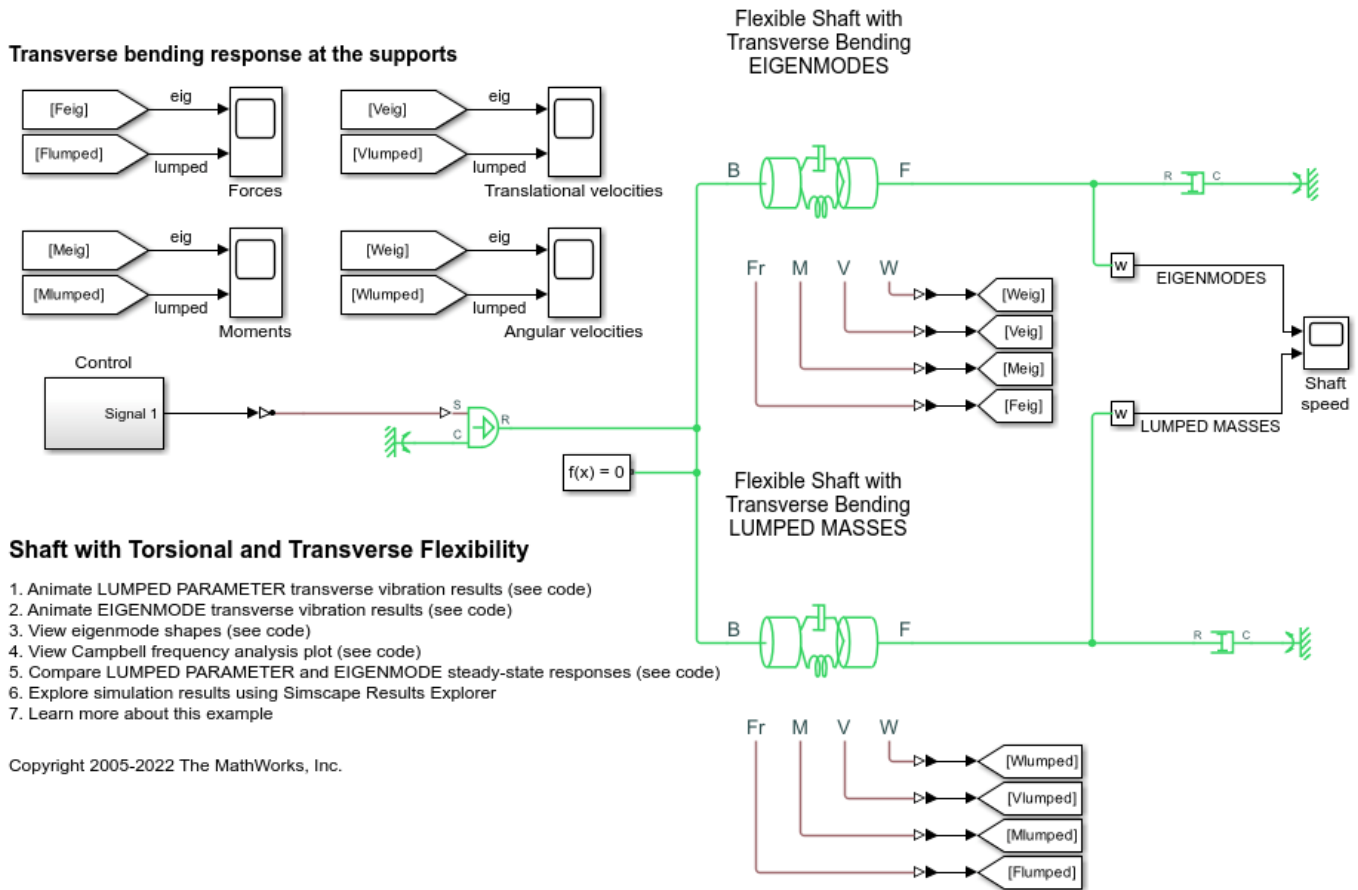
The plot below shows the shaft speeds of the sheet metal feeder and the position of the stock. When the driver clutch is closed, the stock advances. When the brake clutch is closed, the stock remains still.



Shaft with Torsional and Transverse Flexibility

This example shows two flexible shafts that have torsional and transverse flexibility. One shaft computes transverse motion using computed eigenmodes, and the other shaft uses a lumped parameter approach. The boundary conditions on the shafts can be changed on the block masks.

Model



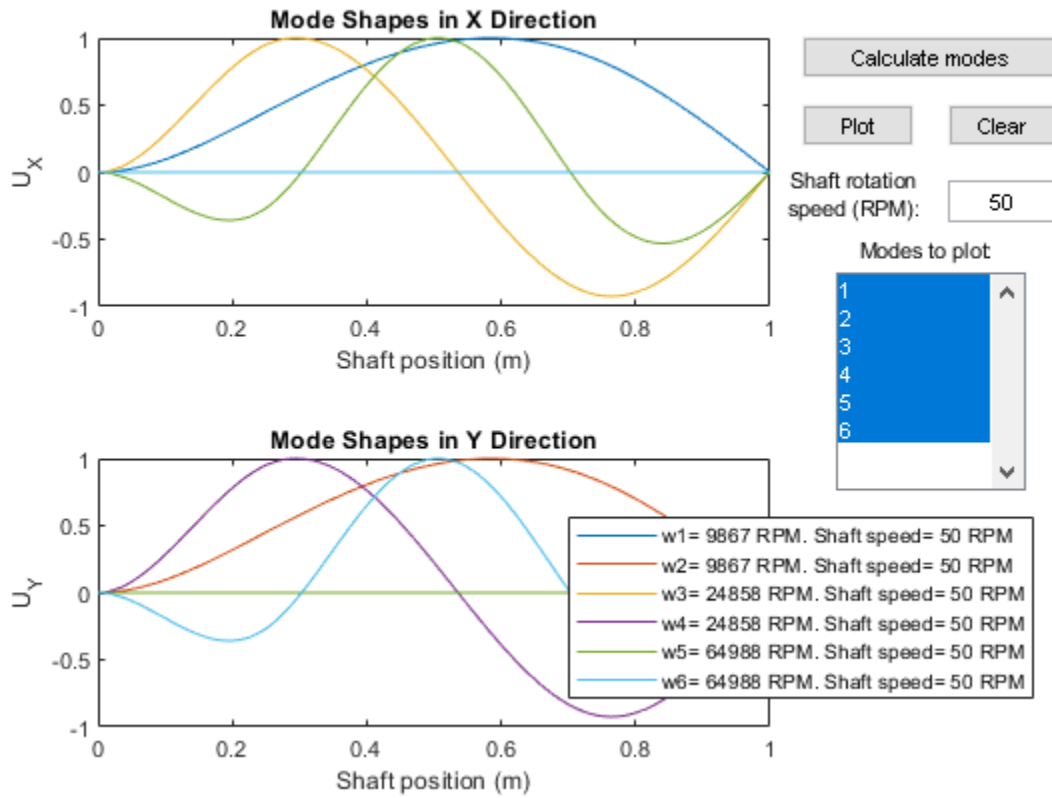
Shaft with Torsional and Transverse Flexibility

1. Animate LUMPED PARAMETER transverse vibration results (see code)
2. Animate EIGENMODE transverse vibration results (see code)
3. View eigenmode shapes (see code)
4. View Campbell frequency analysis plot (see code)
5. Compare LUMPED PARAMETER and EIGENMODE steady-state responses (see code)
6. Explore simulation results using Simscape Results Explorer
7. Learn more about this example

Copyright 2005-2022 The MathWorks, Inc.

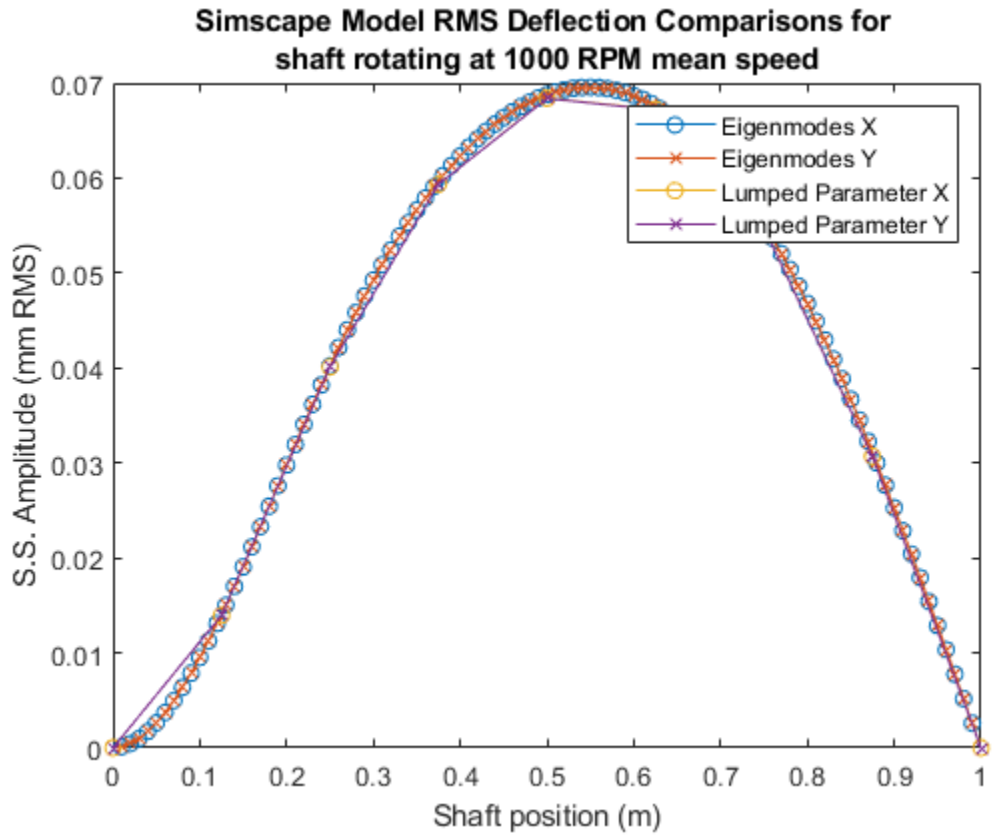
Computed Eigenmodes

The plot below shows the eigenmodes computed automatically from the physical parameters of the Flexible Shaft. Alternatively, precomputed eigenmodes can be entered directly as block parameters.



Simulation Results from Simscape Logging

The plots below show a comparison of the results from the lumped mass and eigenmode approaches.

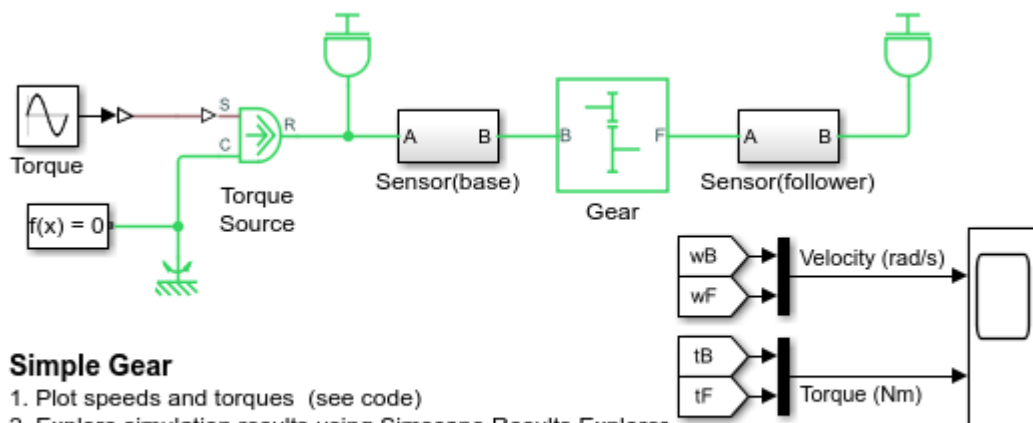


Simple Gear

This example shows a simple gear coupling two inertias (shafts). The gear ratio between the follower (F) and base (B) is 2:1. Thus the angular velocity of the follower shaft is half the angular velocity of the base shaft. The follower shaft torque is twice the base shaft torque.

An alternative to adding sensors to your model is to use Simscape™ data logging. This example has data logging enabled, passing results to the workspace variable 'simlog_SimpleGear'. For example, entering 'plot(simlog_SimpleGear.Gear.B.w)' at the command line plots the angular velocity at port B of the Gear block. See the Data Logging Example in the Simscape User Guide for further details.

Model



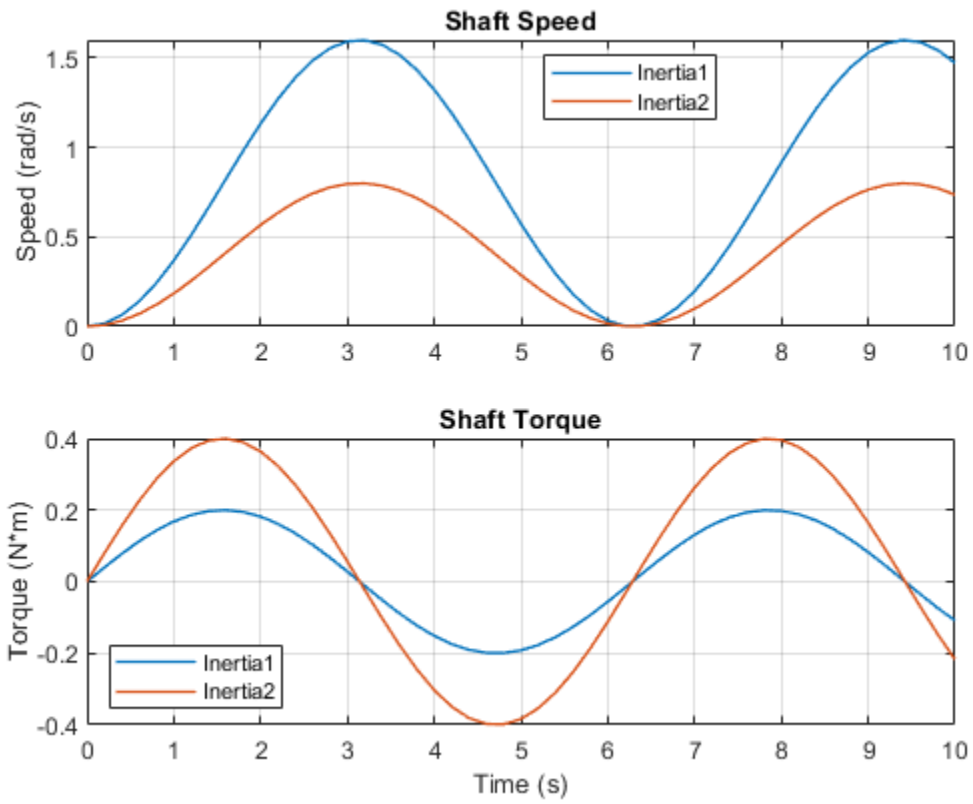
Simple Gear

1. Plot speeds and torques (see code)
2. Explore simulation results using Simscape Results Explorer
3. Learn more about this example

Copyright 2003-2022 The MathWorks, Inc.

Simulation Results from Simscape Logging

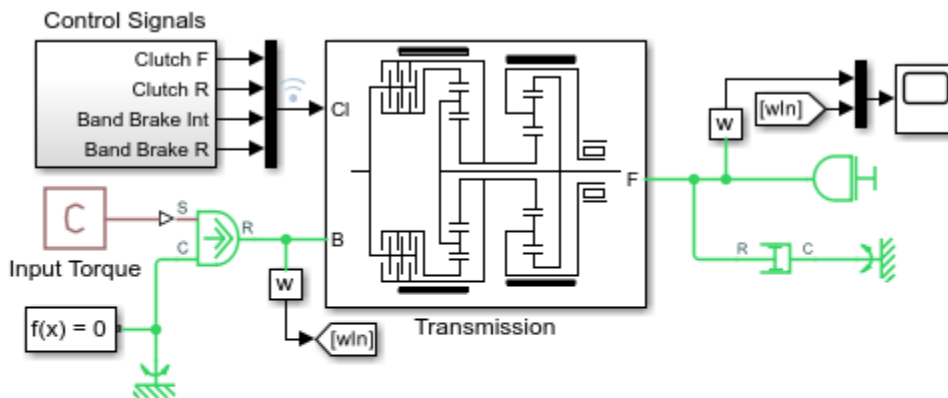
The plots below show the speeds and torques applied to two shafts connected by a gearbox.



Simpson Three-Speed Transmission

This example shows a Simpson transmission with three forward speeds and one reverse speed. Two planetary gear sets are linked with a common sun gear. Two disk friction clutches and two band brakes determine which components can rotate relative to one another which determines the final gear ratio. The Clutch Schedule subsystem shows which clutches and brakes should be locked for each forward or reverse speed.

Model

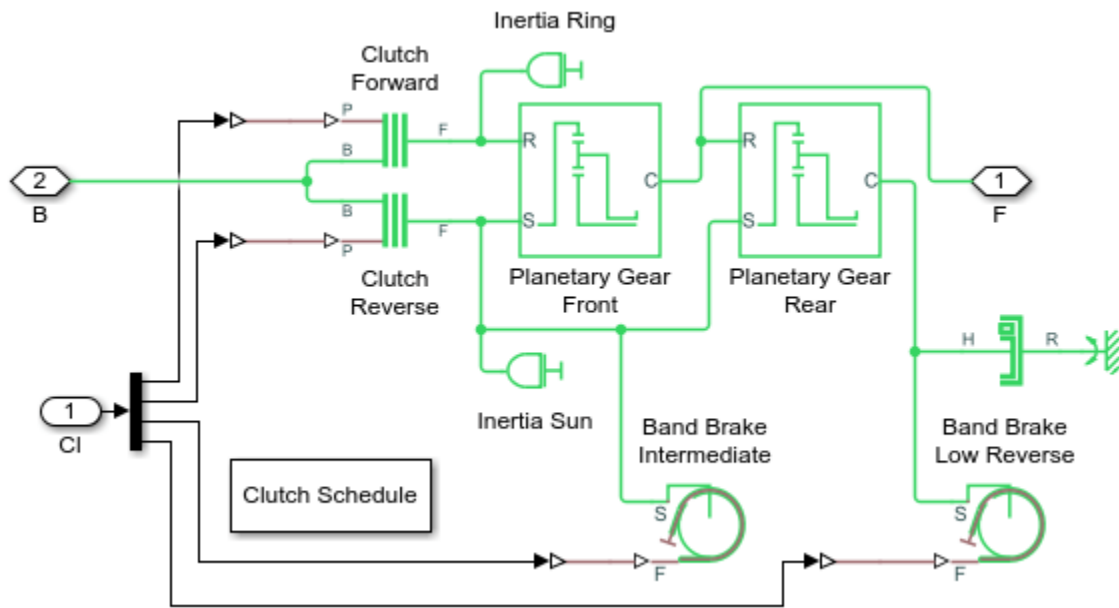


Simpson Three-Speed Transmission

1. Plot speeds of input and output shafts (see code)
2. Explore simulation results using Simscape Results Explorer
3. Learn more about this example

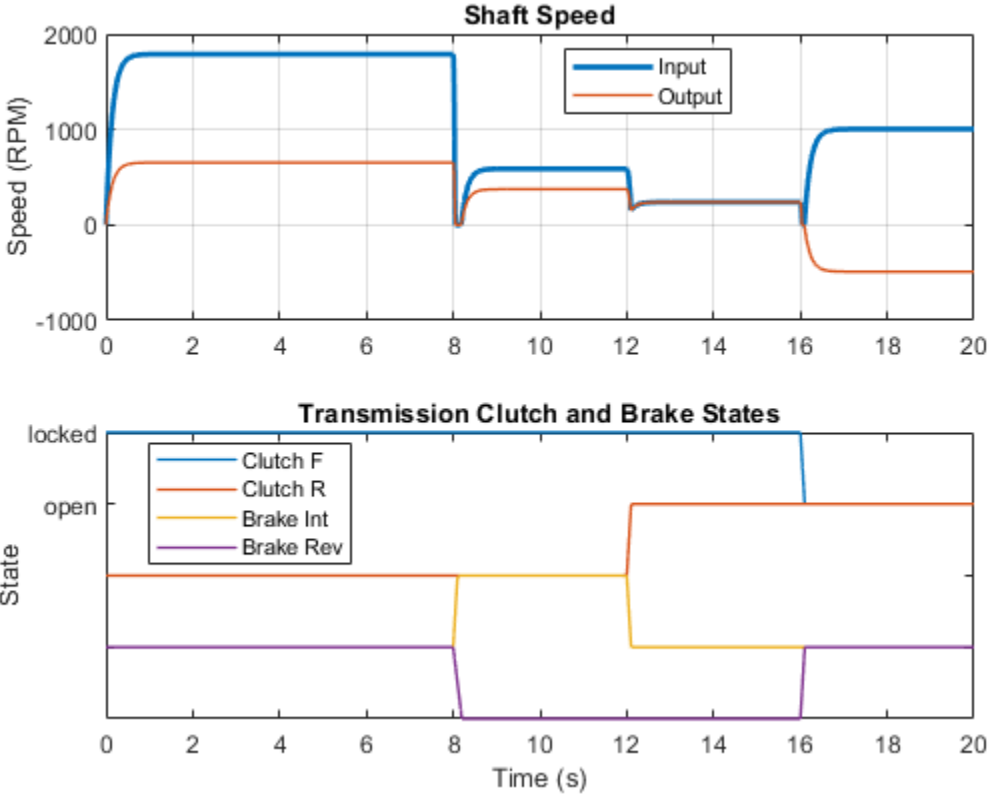
Copyright 2012-2022 The MathWorks, Inc.

Transmission Subsystem



Simulation Results from Simscape Logging

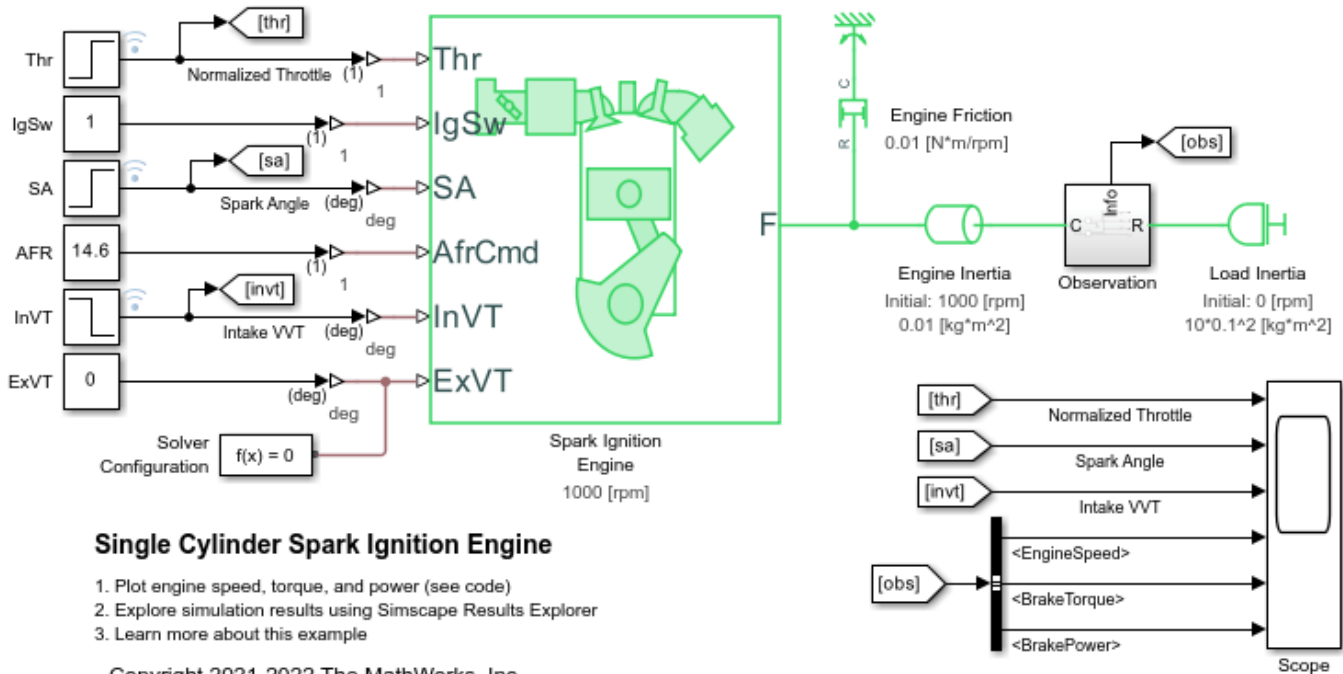
The plot below shows the speed of the input and output shafts of the Simpson 3-speed transmission. This test cycles through four gear ratios, three for forward and one for reverse. The states of the two clutches and two band brakes are also shown.



Single Cylinder Spark Ignition Engine

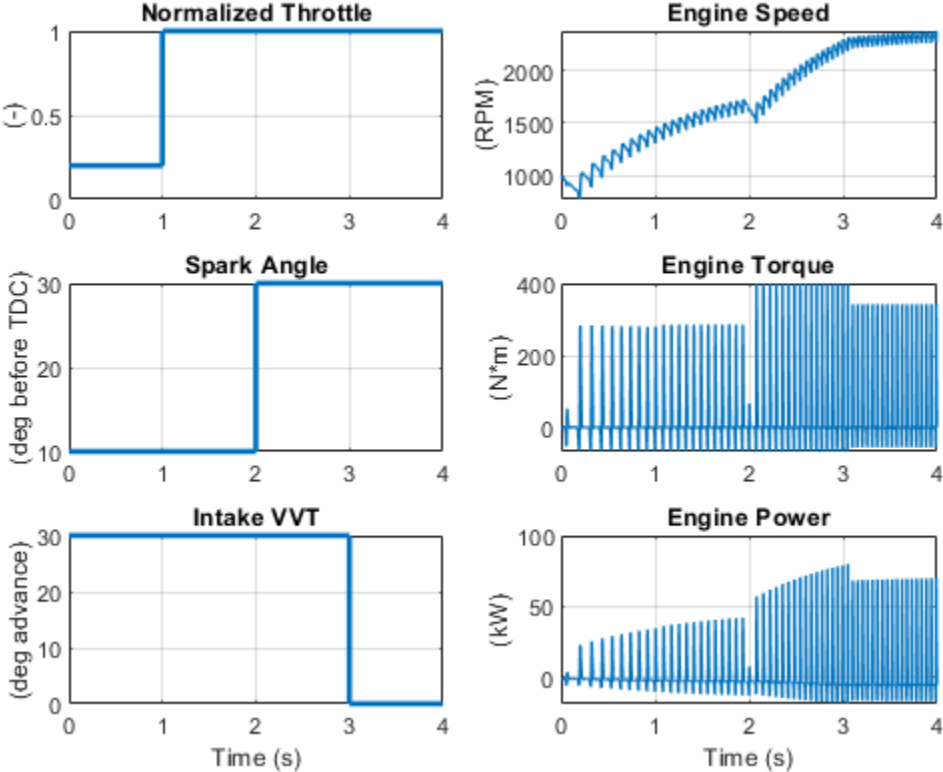
This example shows a crank-angle-resolved, naturally aspirated, spark ignited single cylinder engine. This example varies control inputs including the spark angle and the intake VVT during the simulation.

Model



Simulation Results from Simscape Logging

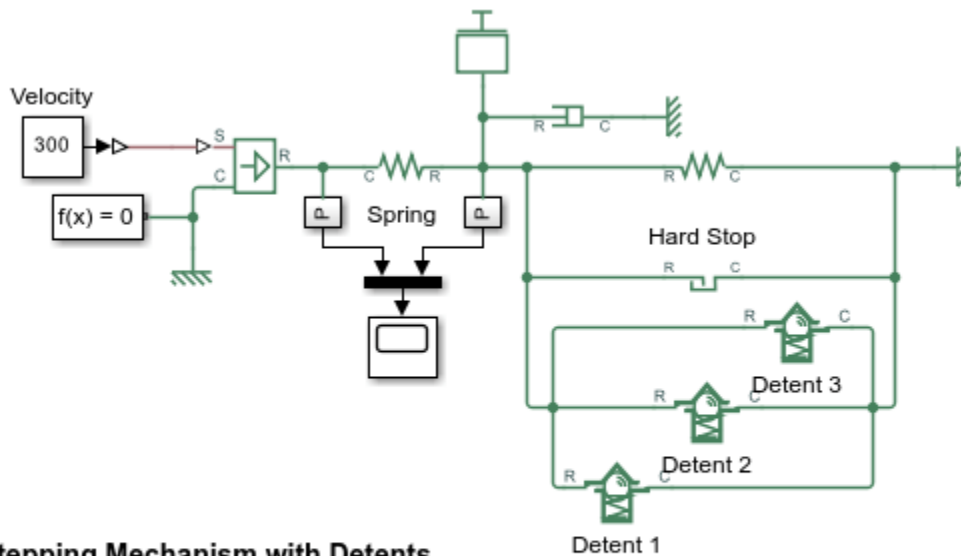
The plots below show the throttle, spark angle, and intake VVT inputs to the engine and the speed, torque, and power responses.



Stepping Mechanism with Detents

This example shows a mass moving along a slider and held at set points with translational detents. The mass is suspended between two springs and is subject to viscous friction. Three detents hold the load at 50, 75, and 125 mm. The peak force of these detents is 20, 30, and 20 N respectively. To move the load past each detent, the spring pushing the load must deform enough to produce a force that overcomes the holding force of the detents.

Model



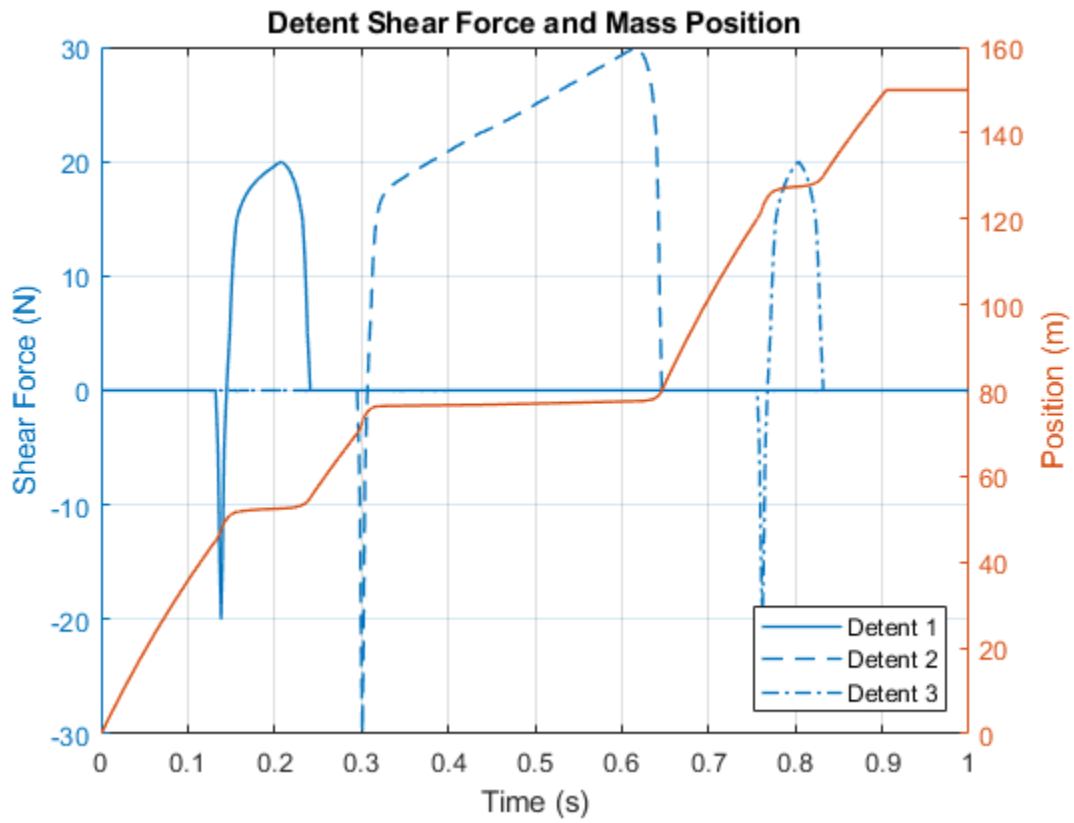
Stepping Mechanism with Detents

1. Plot forces from detents (see code)
2. Explore simulation results using Simscape Results Explorer
3. Learn more about this example

Copyright 2007-2022 The MathWorks, Inc.

Simulation Results from Simscape Logging

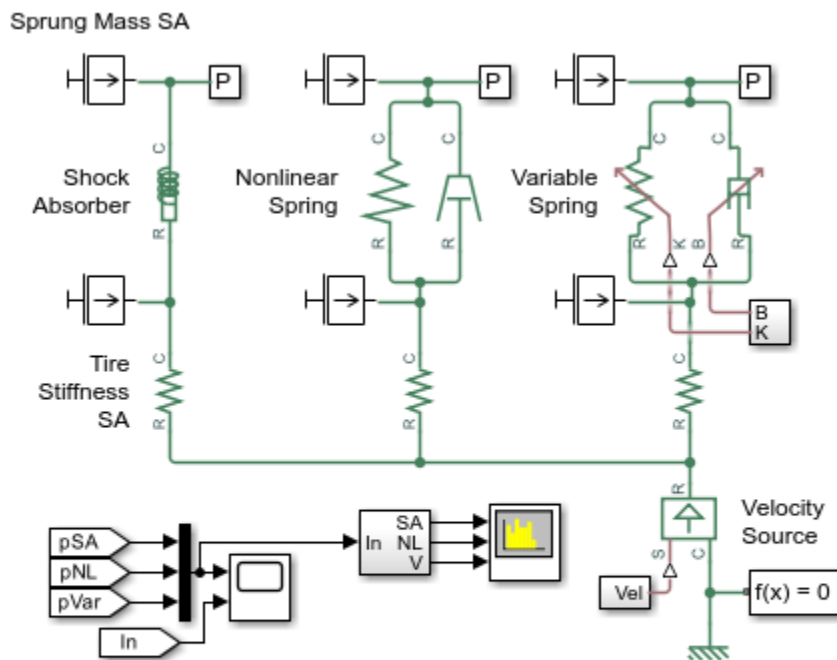
The plot below shows the shear force applied by each detent to the mass. As the force applied exceeds shear force, the detent member is pushed into its hole and the mass can continue moving until it reaches the hard stop.



Suspension System Comparison

This example shows a testbed containing three sets of springs and dampers excited by the same oscillating velocity source. The first uses the Shock Absorber block and includes linear stiffness and damping. Optional friction and hard stops are not used. The second shock absorber uses a Nonlinear Translational Spring and a Nonlinear Translational Damper specified by symmetric polynomials. The third uses a Variable Translational Spring and Variable Translational Damper. The spring constant is varied during the simulation using open loop control and the damping coefficient adjusts to ensure critical damping is achieved. Closed loop control can be added to simulate an adaptive suspension system.

Model



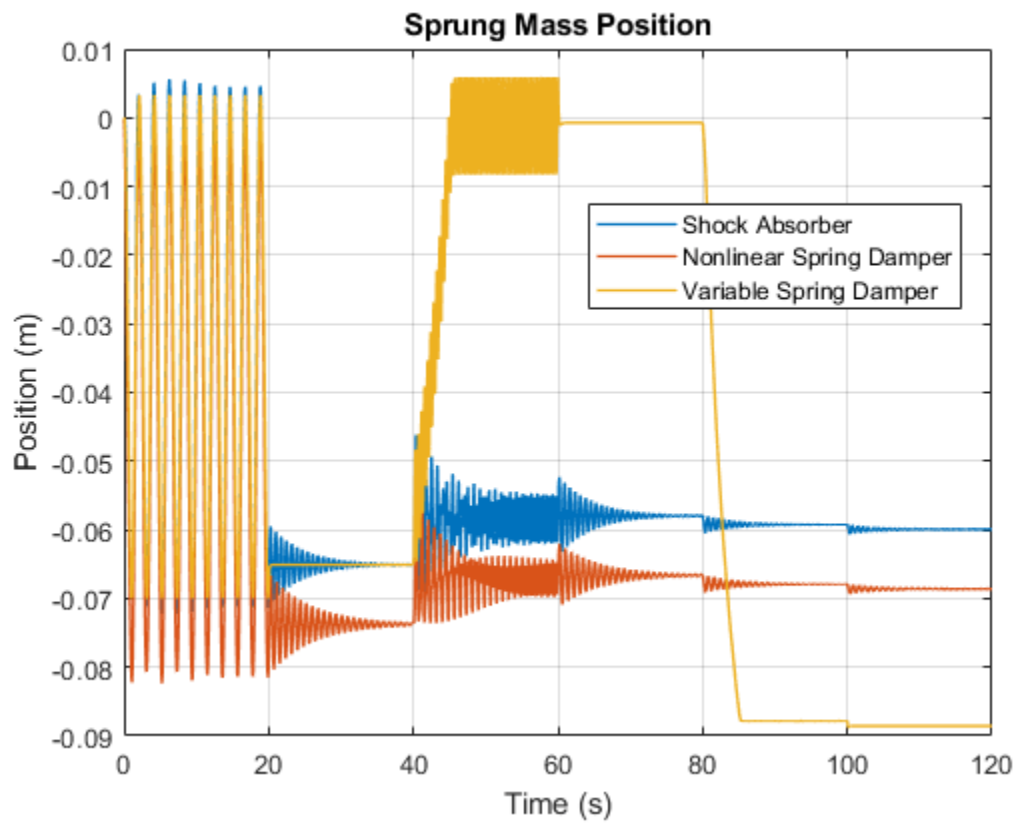
Suspension System Comparison

1. Plot position of sprung masses (see code)
2. Explore simulation results using Simscape Results Explorer
3. Learn more about this example

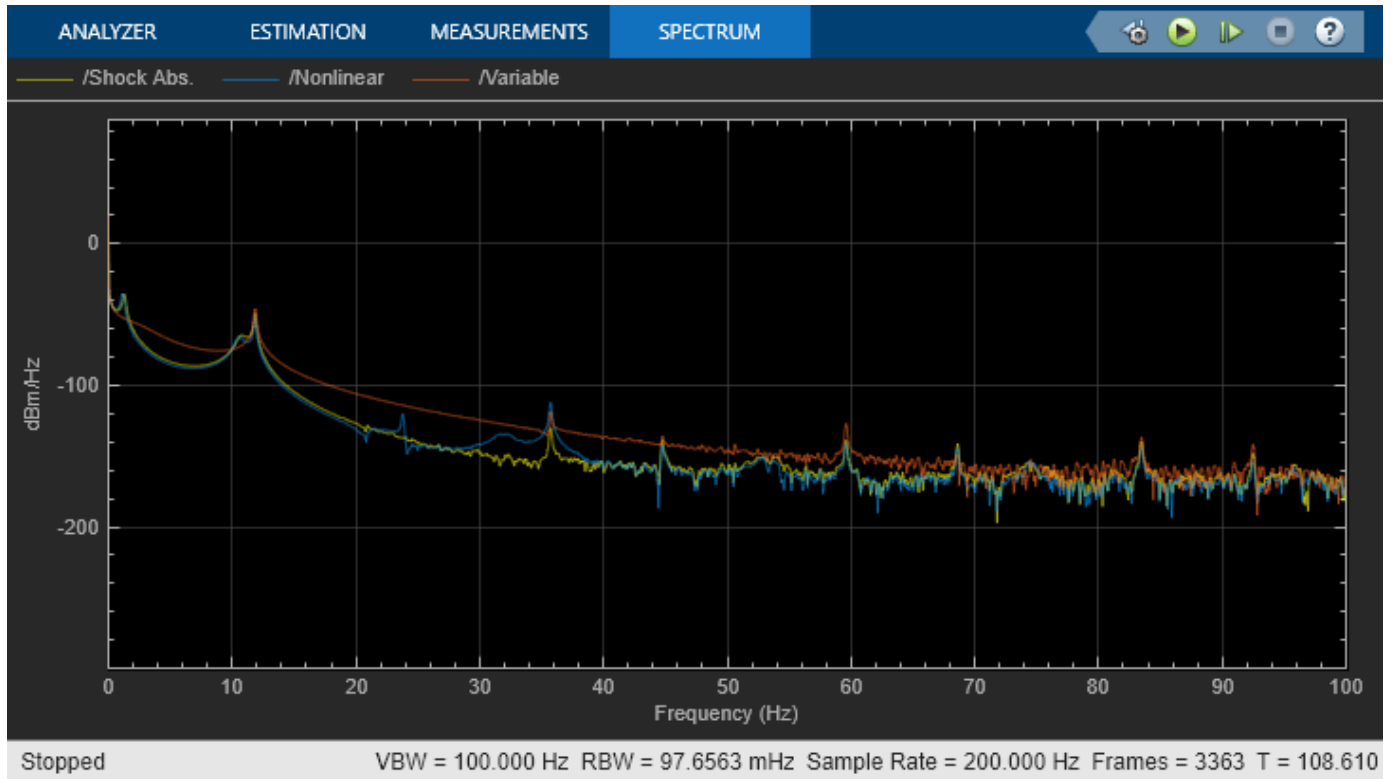
Copyright 2012-2022 The MathWorks, Inc.

Simulation Results from Simscape Logging

The plot below shows the position of the sprung masses in three different suspension models that are subjected to the same test.



Power Spectral Density

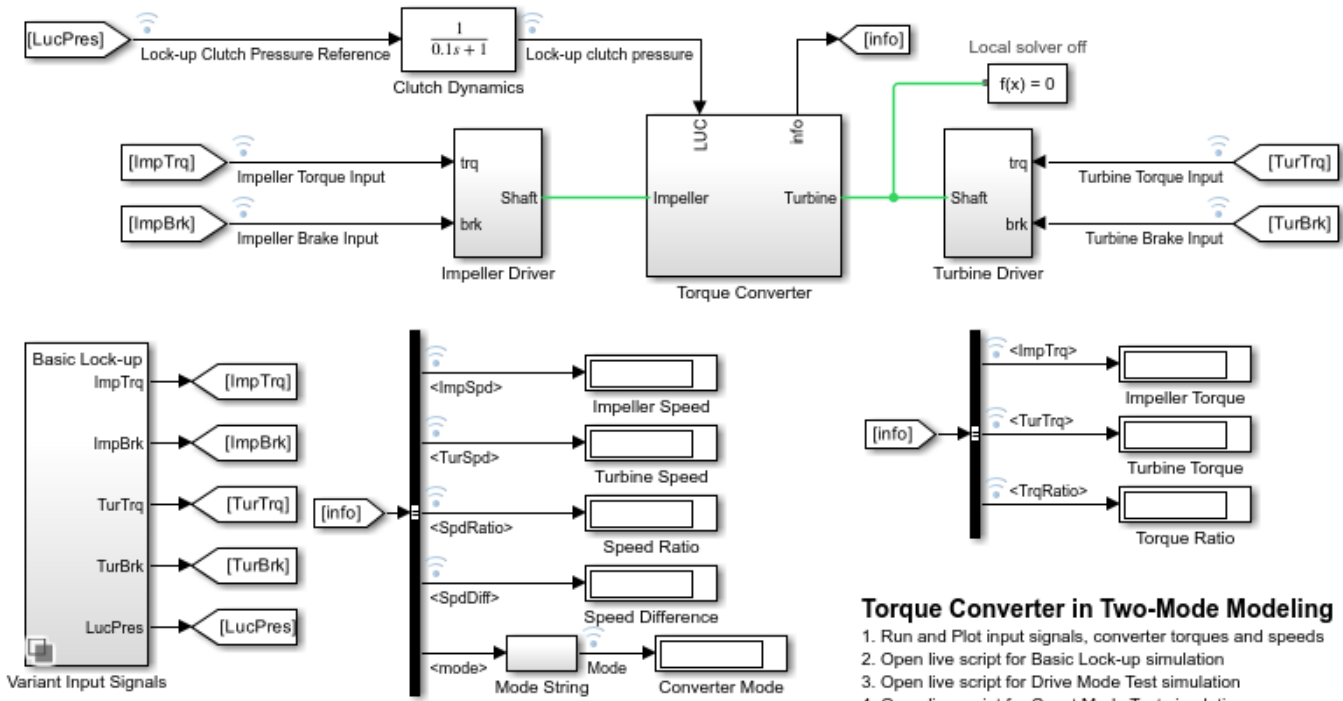


Torque Converter in Two-Mode Modeling

This example shows basic operations of a torque converter including lock-up clutch mechanism. You can select three different simulation cases - Basic Lock-up, Drive Mode Test, and Coast Mode Test.

The impeller and turbine of the converter are connected with external torque sources and brakes. The lock-up clutch is connected with an external pressure source. Input Signal block allows you to select the simulation cases. Basic Lock-up simulates elementary operating patterns of the converter in both drive mode and coast mode including lock-up. Drive Mode Test and Coast Mode Test perform speed ratio sweep and yield operating characteristics plots such as a plot of converter efficiency as a function of speed ratio.

Model

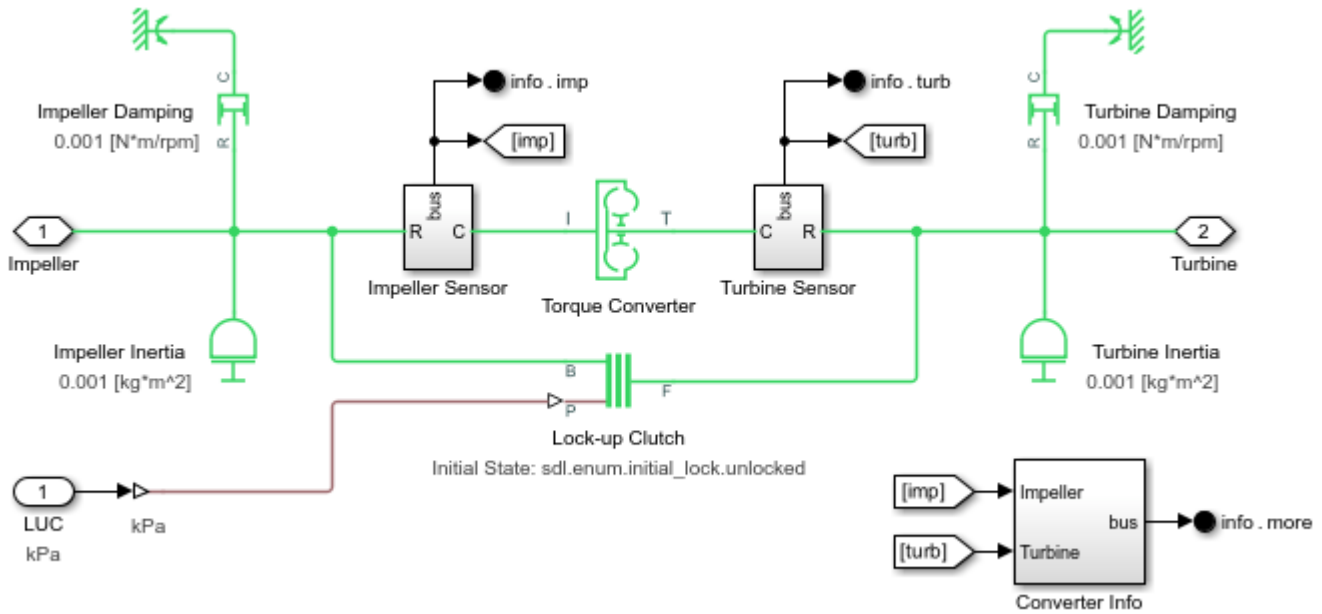


Torque Converter in Two-Mode Modeling

1. Run and Plot input signals, converter torques and speeds
2. Open live script for Basic Lock-up simulation
3. Open live script for Drive Mode Test simulation
4. Open live script for Coast Mode Test simulation
5. Explore simulation results using Simscape Results Explorer
6. Learn more about this example

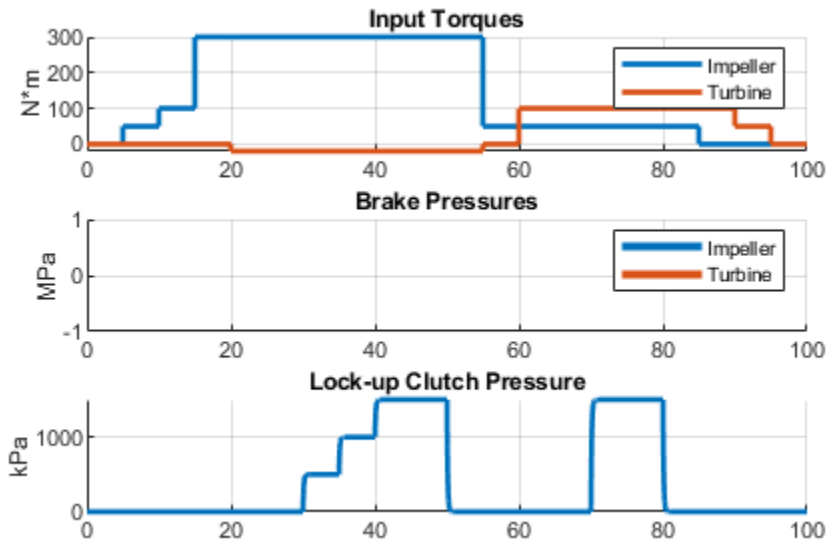
Copyright 2020-2022 The MathWorks, Inc.

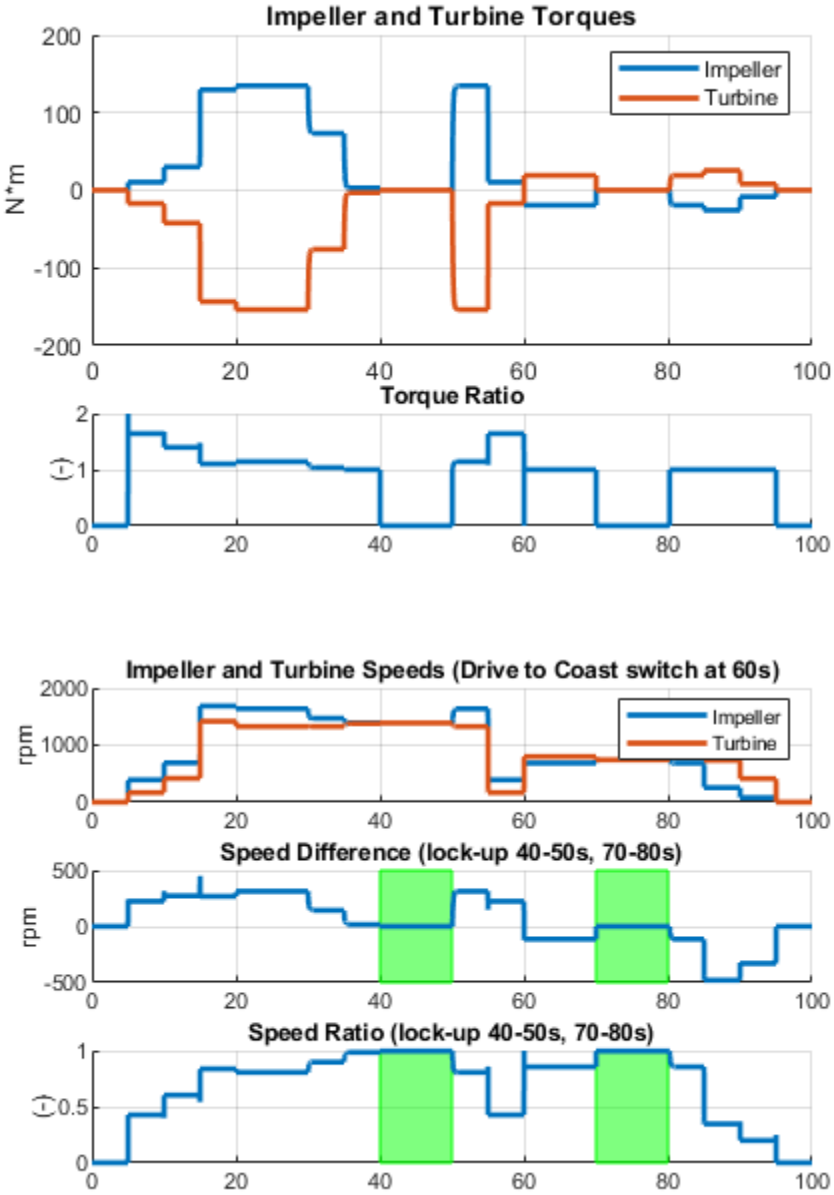
Torque Converter Subsystem



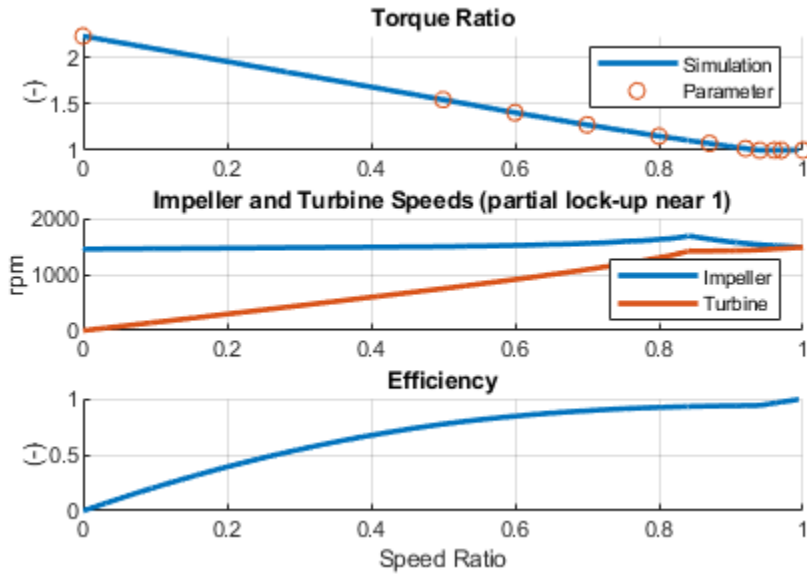
Simulation Results

The plots below show the result of Basic Lock-up simulation. When simulation starts, the torque converter goes into drive mode and the lock-up clutch is engaged. Then the converter switches over to coast mode and the lock-up is engaged again. The green background areas indicate that the lock-up clutch is active.

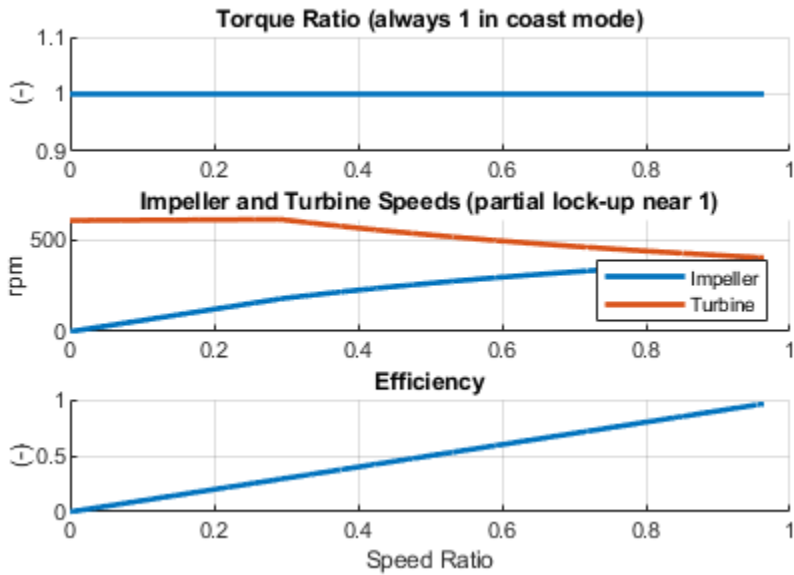




The plots below show the converter operating characteristics in drive mode.



The plots below show the converter operating characteristics in coast mode.

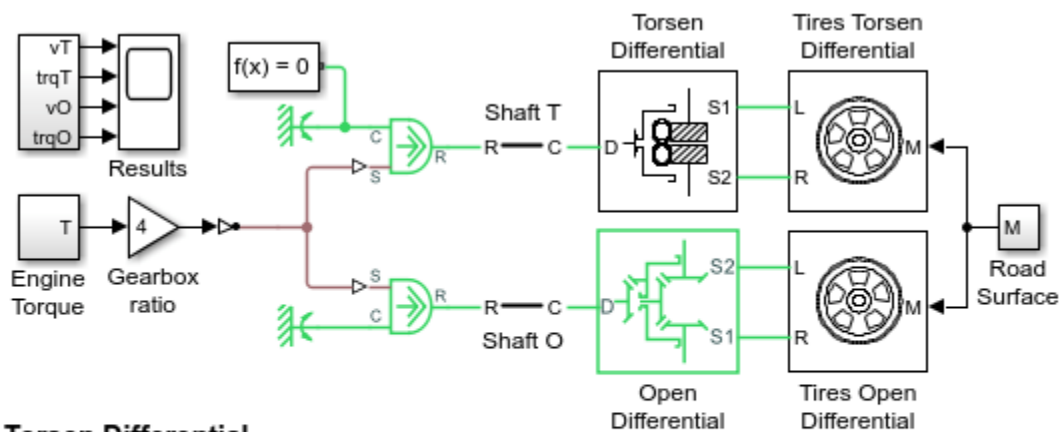


Torsen Differential

This example shows a comparison between the behavior of an open differential and a Torsen limited slip differential. The Torsen differential is modeled using components from the Gears library in Simscape™ Driveline™. Slip is limited in the Torsen differential because it uses non-backdrivable worm gears, which are modeled by Sun-Planet Worm Gear components. The result is higher torque applied to the wheel with greater traction, and identical speeds for the left and right axles.

The test surface includes an icy patch under the left wheel. This effect is introduced using the variable-friction coefficient variant of the Tire (Magic Formula) block. Comparing the two differentials on the same test surface shows that the Torsen differential locks up under the split surface road condition.

Model

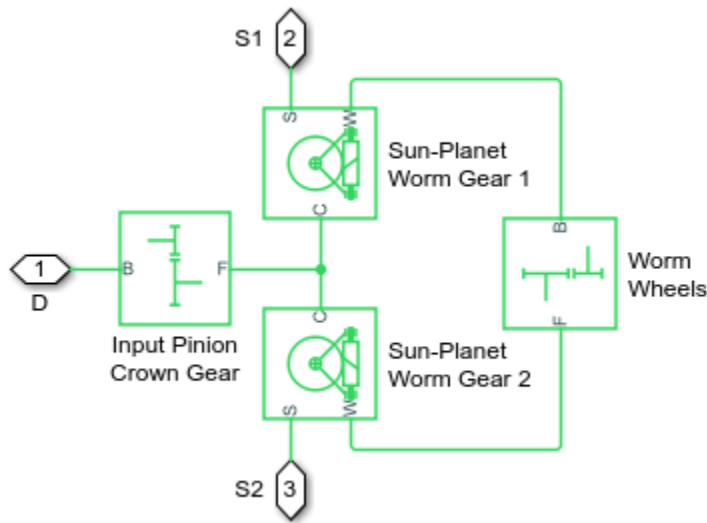


Torsen Differential

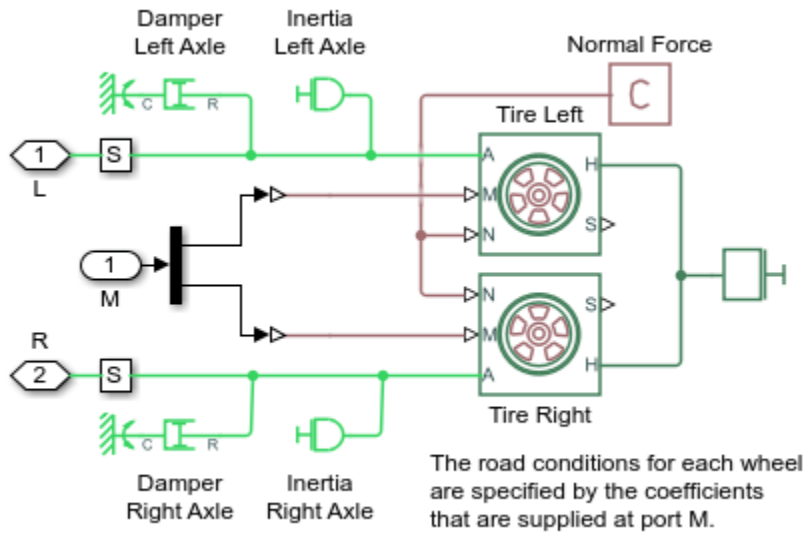
1. Plot wheel speeds for both differentials (see code)
2. Plot shaft torques for both differentials (see code)
3. Explore simulation results using Simscape Results Explorer
4. Learn more about this example

Copyright 2006-2022 The MathWorks, Inc.

Torsen Differential Subsystem

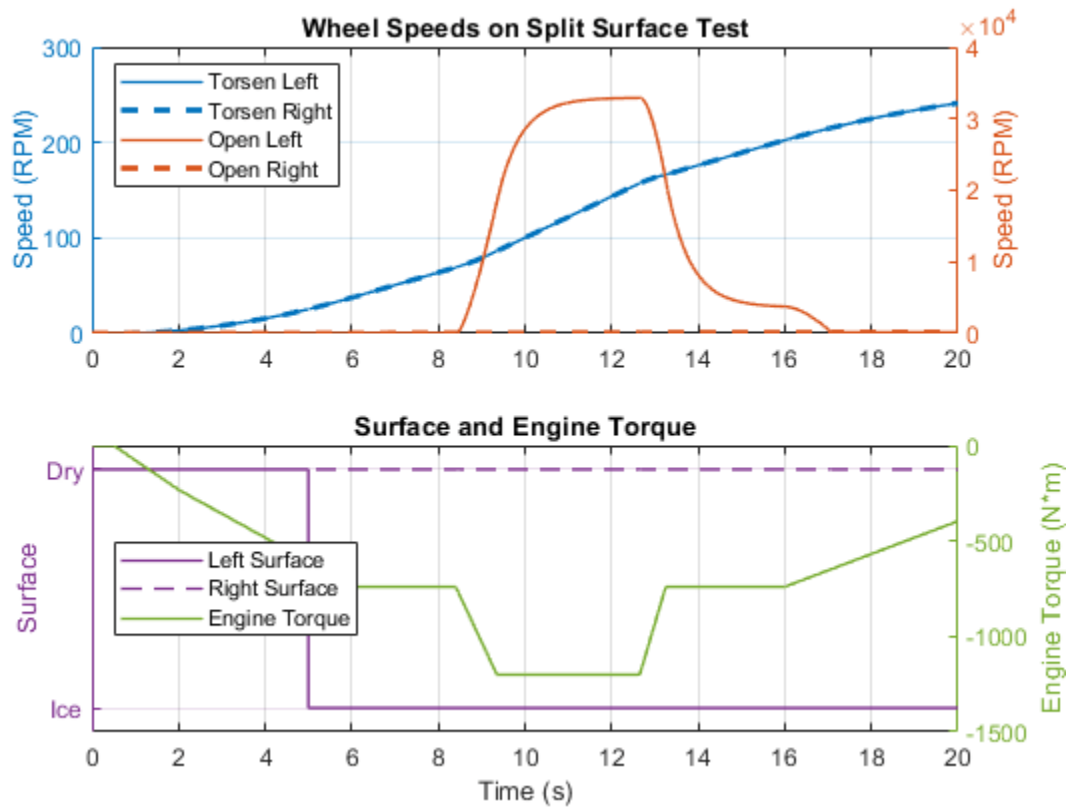


Tires Subsystem

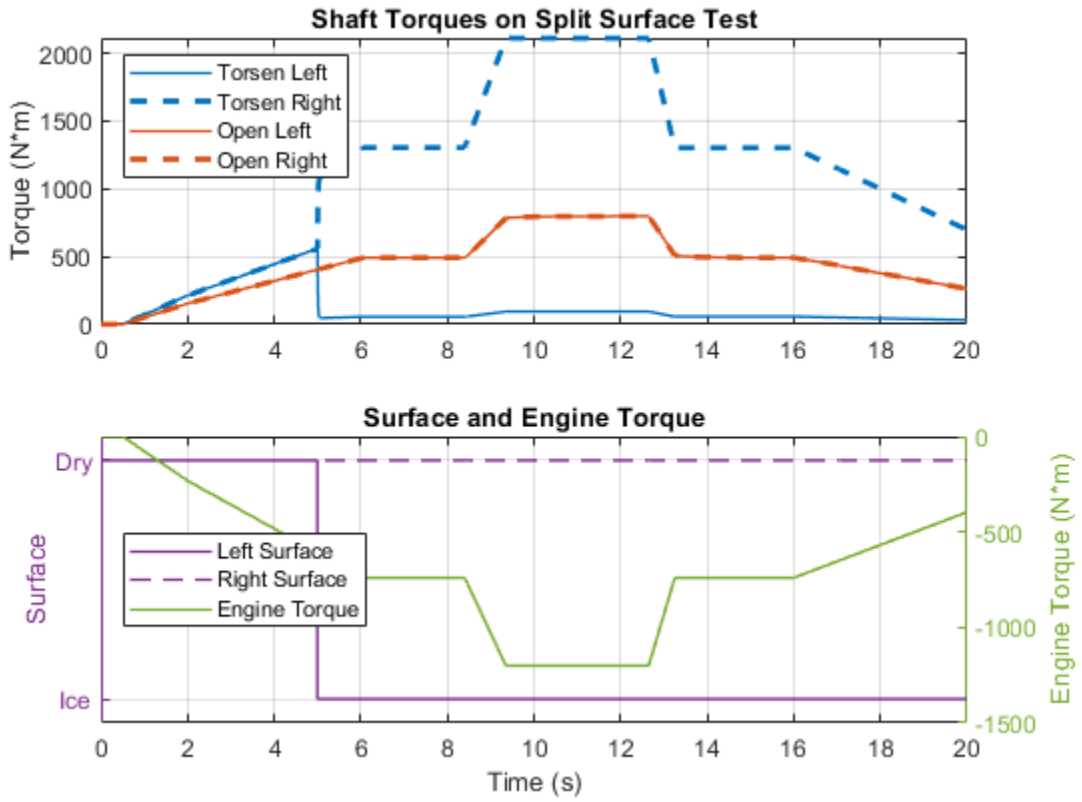


Simulation Results from Simscape Logging

The plot below shows the performance of open and Torsen differentials on a split surface (ice and tarmac). The Torsen differential locks instantly when the left wheel encounters the icy patch, and the left and right wheel speeds remain the same. The open differential does not lock and applies the same torque to both shafts. The result is that the left wheel loses traction on the icy patch and slips.



The plot below shows the performance of open and Torsen differentials on a split surface (ice and tarmac). The Torsen differential locks instantly when the left wheel encounters the icy patch. As a result, the wheels turn at the same speed and more torque is applied to the wheel on the high friction surface. The open differential does not lock. The same torque is applied to both wheels, resulting in the left wheel slipping extensively on the icy patch.

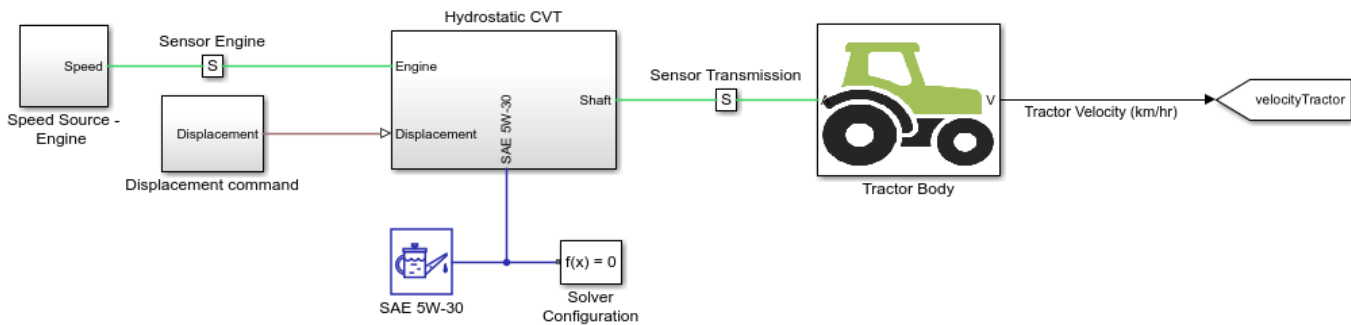


Tractor Transmission Energy Flow Chart

This example shows how to model, parameterize, and test a tractor with a hydrostatic continuously variable transmission (CVT). When you run one of the plot functions, you can view plots of the tractor velocity, engine speed, transmission speed, and pump and motor energy computations.

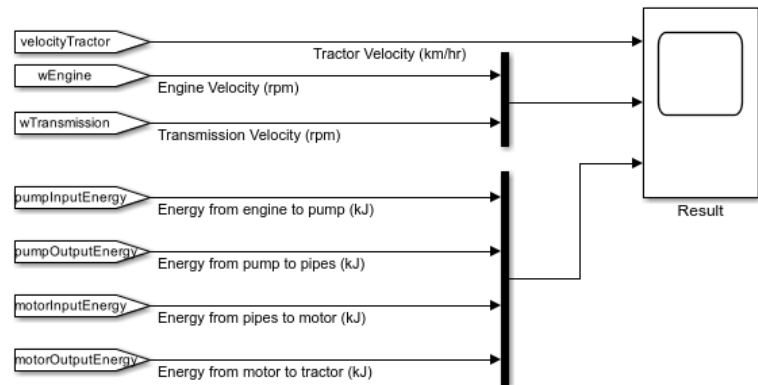
Model

The figure shows the hydrostatic CVT tractor model.



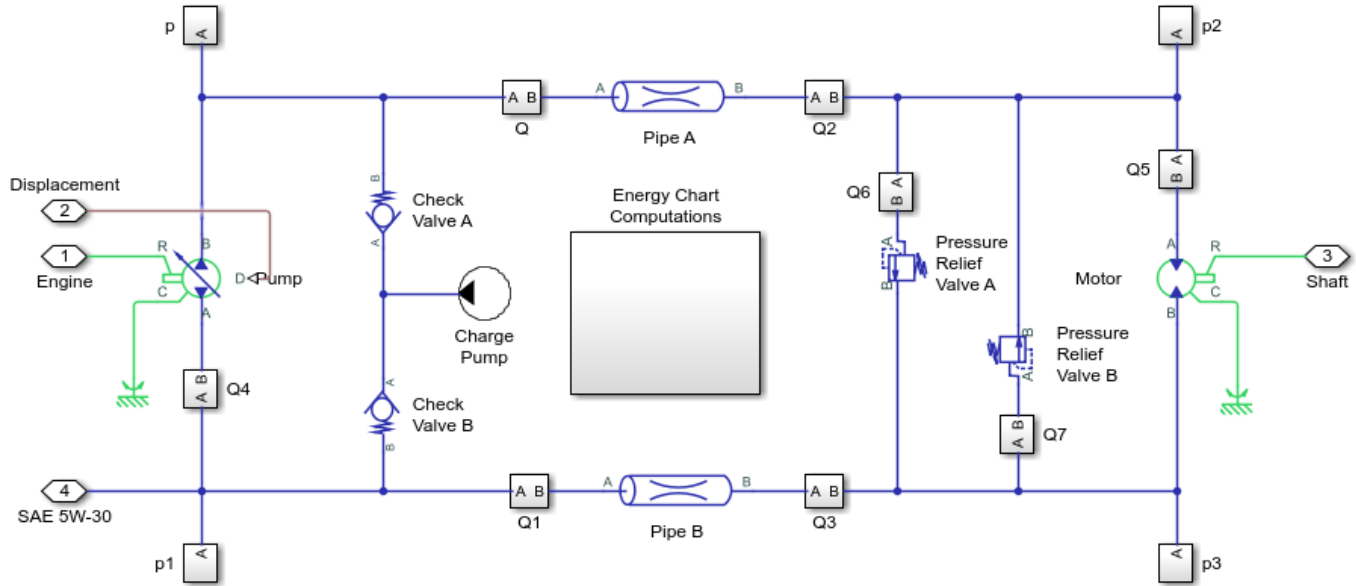
Tractor Transmission Energy Flow Chart

1. Open parameter initialization script (see code)
 2. Plot Tractor speed curve (see code)
 3. Plot pump and motor energy curves (see code)
 4. Explore simulation results using Simscape Results Explorer
 5. Learn more about this example
- Copyright 2022 The MathWorks, Inc.

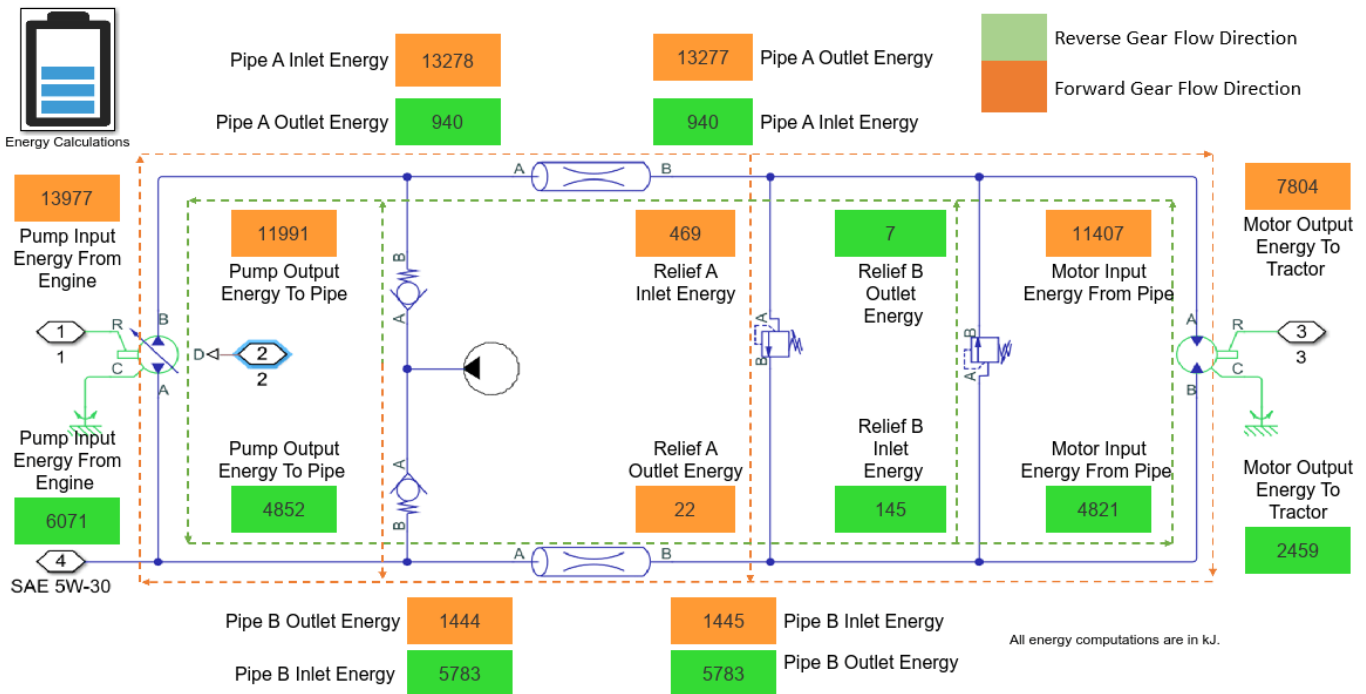


Hydrostatic CVT Subsystem

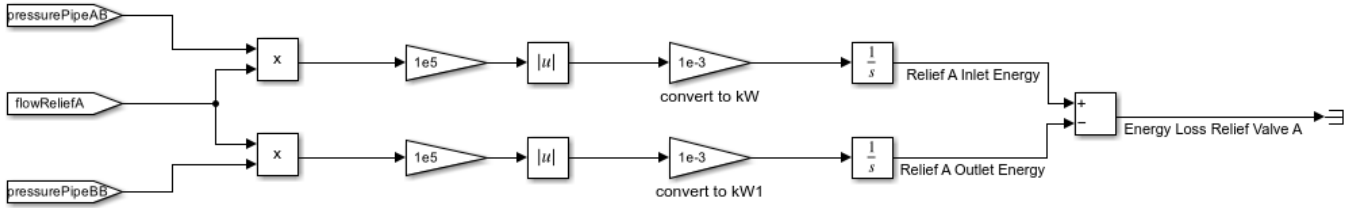
This subsystem demonstrates how to model the hydrostatic transmission. Here, connection Engine is a physical signal port associated with the speed source representing the engine. Connection Shaft is a physical signal port associated with the transmission shaft connecting with the final drive of the Tractor. Connection SAE 5W-30 is a physical signal port associated with the fluid properties of the Isothermal Liquid network that represents the transmission hydraulics. Connection Displacement is a physical signal port associated with the axial piston pump displacement command. The driver provides the swash displacement command in a tractor. The charge pump maintains the minimum pressure in the low pressure line of the circuit. The relief valves limit the maximum pressure in the circuit's high pressure line.



Energy Chart Computations: This subsystem shows the energy chart of the tractor transmission system. The green dashed line shows the fluid flow path while in reverse, and the orange line shows the fluid flowpath while in forward gears. The pump displacement is positive for the forward gear and negative for the reverse gear tractor operation. You can compute the available, useful, recirculated, wasted energies as well as the efficiency of the components from the energy flow chart.

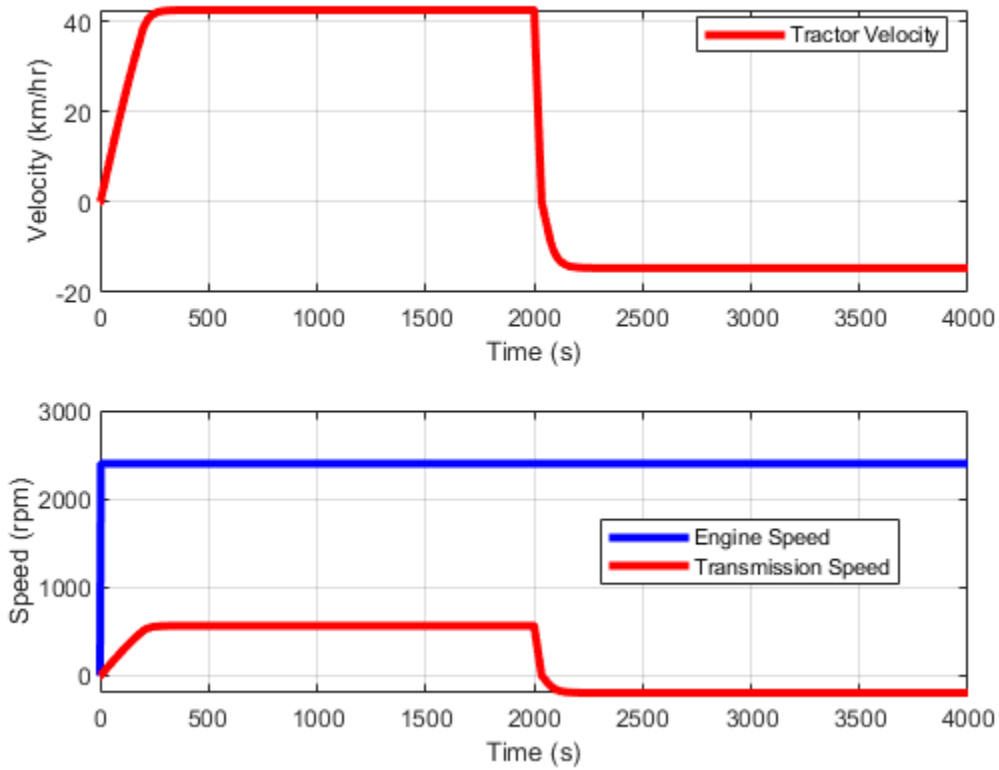


Energy Calculations: This subsystem Relief Valve A Energy Computation shows how to perform energy calculations for a relief valve.

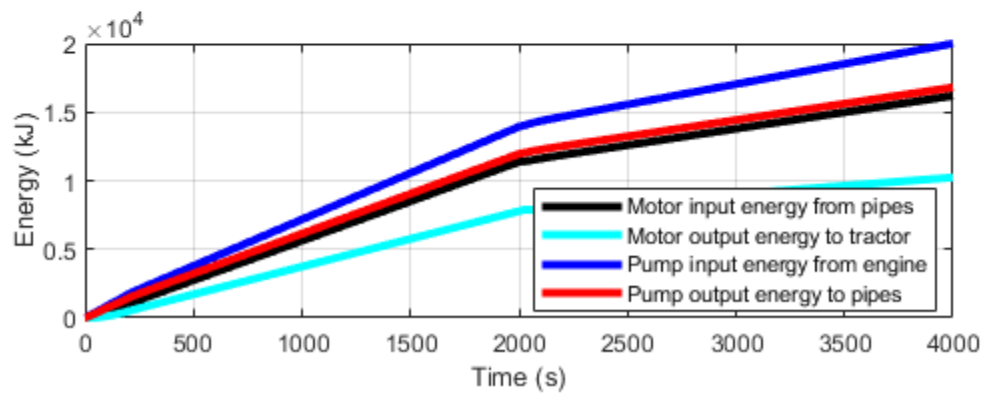
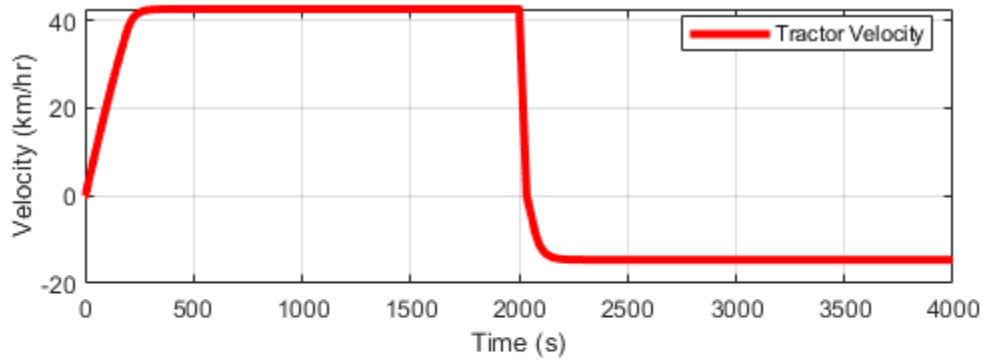


Simulation Results from Simscape™ Logging

This model generates a plot of the tractor velocity, engine speed, and transmission speed.



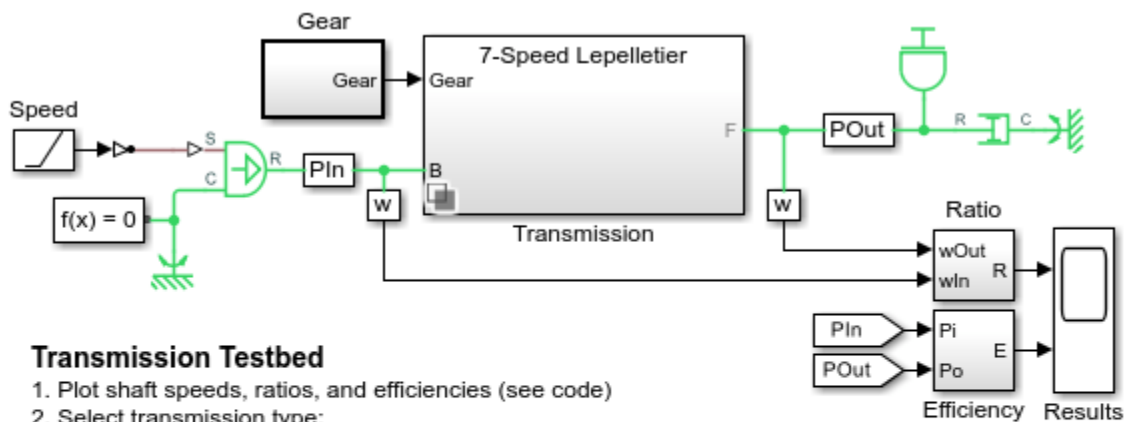
This model also generates a plot of the input and output energies of the pump and motor.



Transmission Testbed

This example shows a testbed with interchangeable transmissions. The transmission models vary from classic four speed transmissions to modern seven, eight, nine, and ten speed configurations. The efficiency and drive ratio can be adjusted by varying the components in each individual transmission configuration. The transmission choices are held in a variant subsystem and are selected by either using the hyperlinks in the model or right-clicking the Transmission subsystem, selecting Variant -> Override using, and the desired variant.

Model



Transmission Testbed

1. Plot shaft speeds, ratios, and efficiencies (see code)
2. Select transmission type:
 - 4-Speed: Ravigneaux, CR-CR
 - 6-Speed Lepelletier
 - 7-Speed Lepelletier
 - 8-Speed
 - 9-Speed
 - 10-Speed
3. Explore simulation results using Simscape Results Explorer
4. Learn more about this example

Copyright 2015-2022 The MathWorks, Inc.

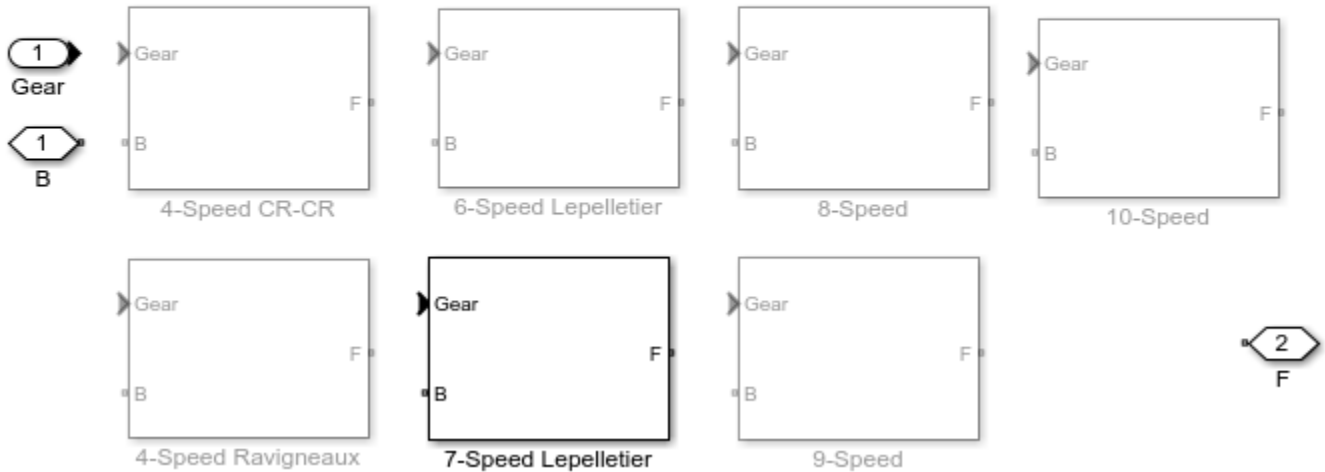
Transmission Variant Subsystem

At this level, the different transmission options are shown as variant subsystems.

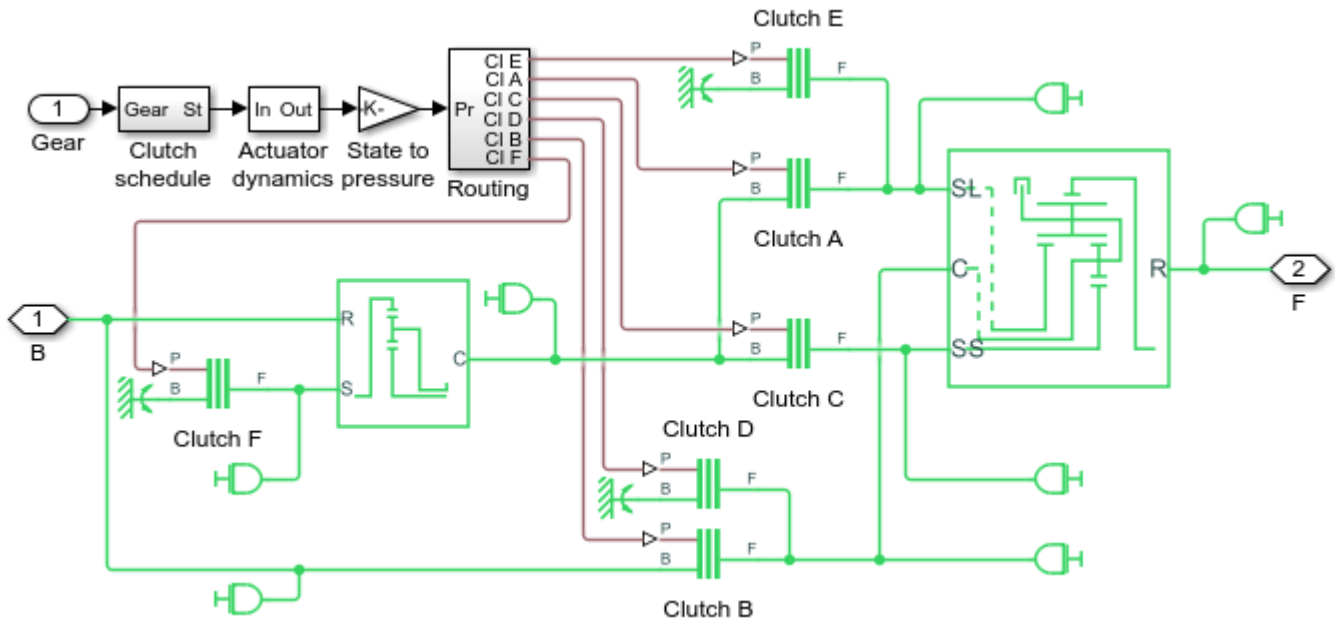
Variants Subsystem

- 1) Add [Subsystem](#) or [Model](#) blocks as valid variant choices.
- 2) You cannot connect blocks at this level.

At simulation, connectivity is automatically determined based on the active variant and port name matching.

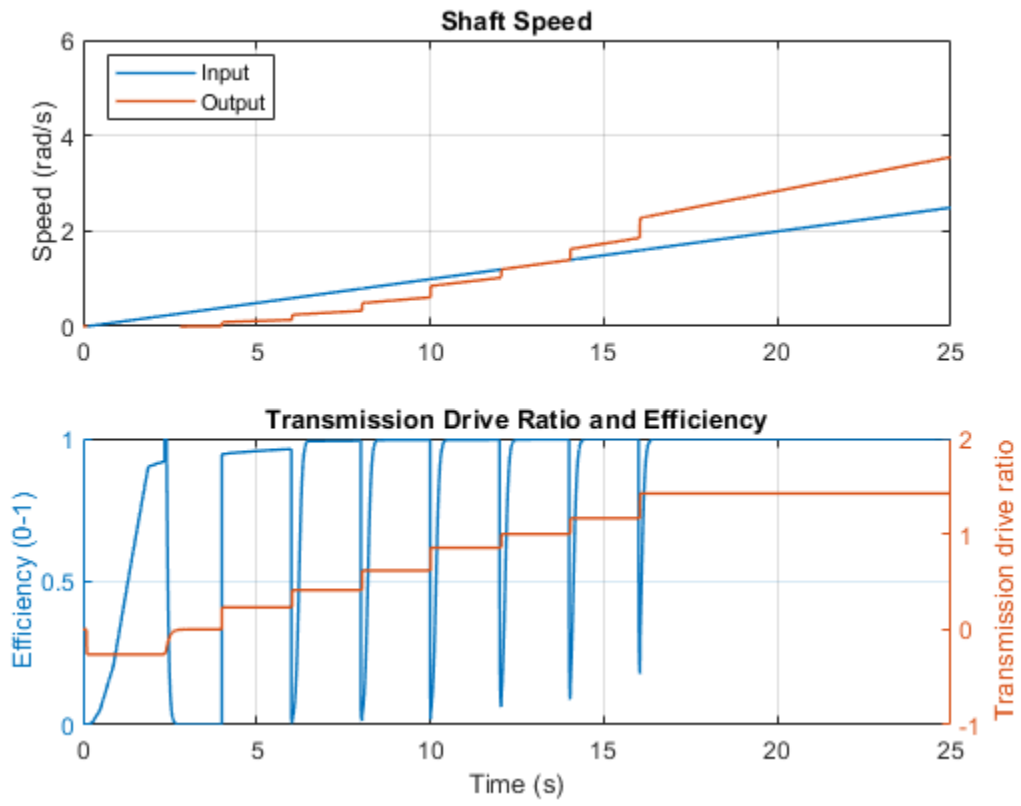


Lepelletier 7 Speed Subsystem



Simulation Results from Simscape Logging

The plots below show the input and output speed of the transmission under test. The gear ratios and overall efficiency for the transmission are shown for each gear.

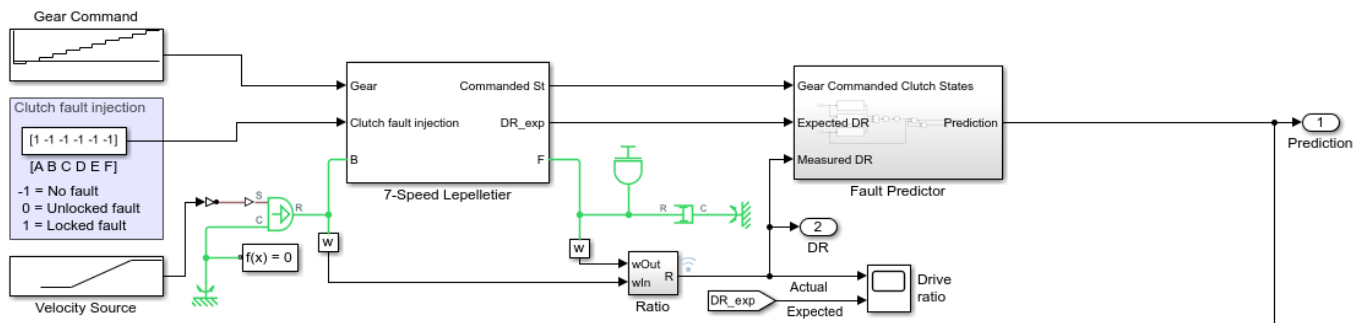


Transmission Fault Detection Harness

This example shows how to inject stuck-off and stuck-on clutch faults into a transmission system, iterate through failure scenarios to gather synthetic training data sets, apply the data to a basic fault classifier algorithm, and measure the classifier's accuracy. This example uses Fast Restart mode to iterate through many different scenarios without needing to recompile the model in between iterations. This workflow is useful for tuning and testing different transmission fault sensor configurations and detector algorithms.

The example is a test harness for a 7-speed Lepelletier transmission. A velocity source at the transmission base drives the system at a constant or increasing ramped velocity. A gear command controls the gear state of the transmission system by controlling the pressure applied to each of the six clutches. When you inject a fault into a clutch, the corresponding clutch pressure remains high or low, depending on if it is an unlocked or locked fault, respectively. The Fault Predictor outputs a probability-like score for each clutch fault type and location by sensing the drive ratio and comparing it to training data containing drive ratios and known clutch states.

Model



Transmission Fault Detection

1. Train fault predictor by collecting clutch states : drive ratio data (see code)
2. Inject a fault:
 Locked clutch: A B C D E F
 Unlocked clutch: A B C D E F
 Clear all faults
3. Generate confusion matrix (see code)
4. Explore simulation results using Simscape Results Explorer
5. Learn more about this example

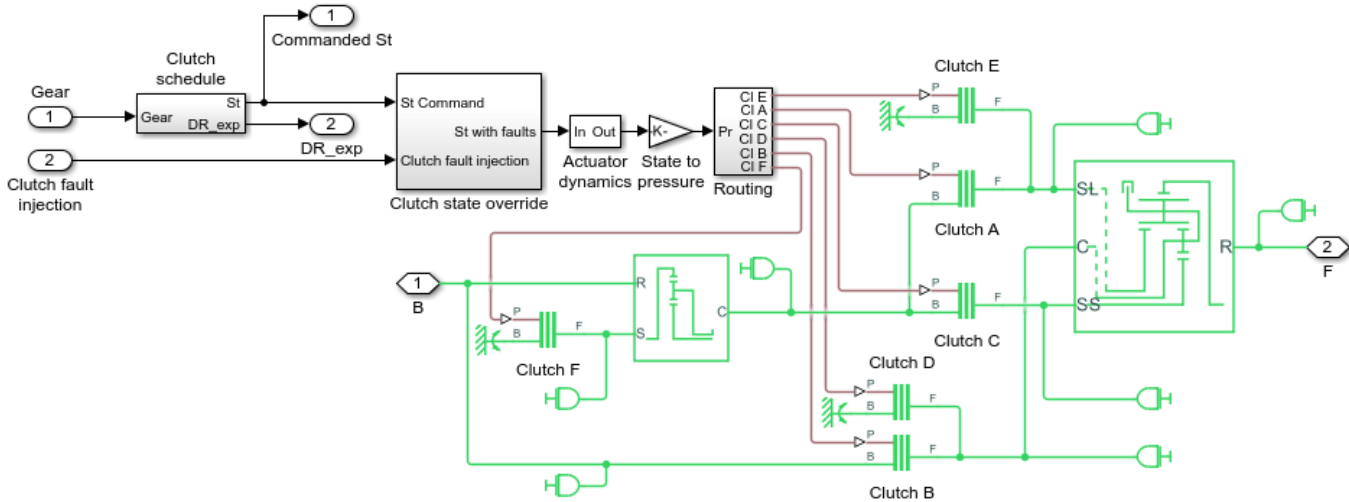
Copyright 2021-2022 The MathWorks, Inc.

Clutch Fault Prediction		A	B	C	D	E	F
Unlocked fault		0.005	0.0025	0.0025	0.1	0.3025	0.005
Locked fault		0.505	0	0.005	0	0.0025	0.0025

0- Low certainty 1- High certainty

7-Speed Lepelletier Transmission

The 7-Speed Lepelletier Transmission subsystem receives a Gear command and a Clutch fault injection signal. The Clutch fault injection signal overrides the clutch state commands for any faulted clutches, so that they differ from the expected commands for one or more transmission gears. The clutch state commands send high or low control pressure to the clutches.



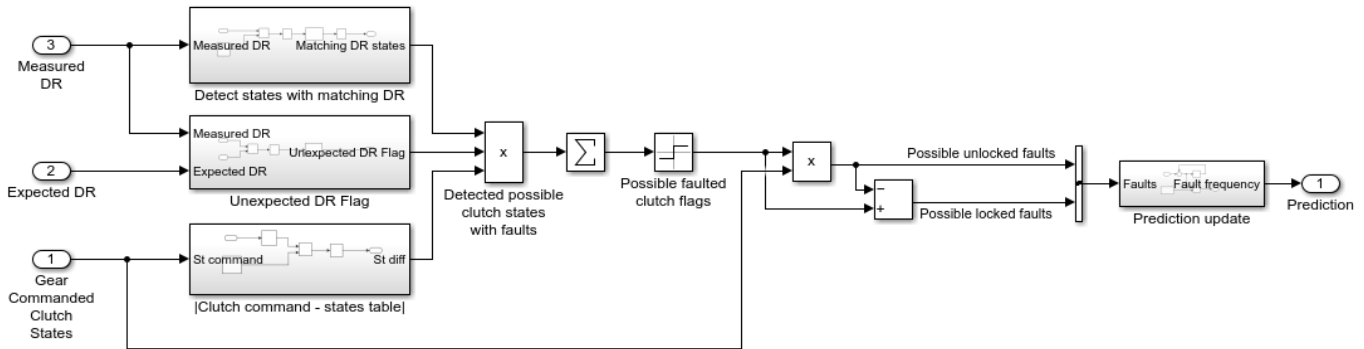
Fault Classifier

This Simulink subsystem implements a basic multi-class classifier to predict fault types and locations. Faults can be stuck-off (unlocked) or stuck-on (locked) and can occur in any of the 6 clutches, resulting in a classification problem with 12 classes. This classifier receives the measured drive ratio, expected drive ratio, and gear commanded clutch states as inputs.

This classifier uses training data derived from a MATLAB function that sweeps through the full-factorial set of all locked/unlocked clutch states. The function simulates the model to collect the measured drive ratios for each state. The function puts the model into Fast Restart mode to collect data for the 64 states without needing to recompile the model in between iterations. The function uses Simscape logging results to exclude scenarios with clutches that have activated control pressures but are slipping, which likely indicates an overconstrained specified clutch state that would result in a severe transmission failure. The training function outputs a matrix containing clutch states and measured drive ratios.

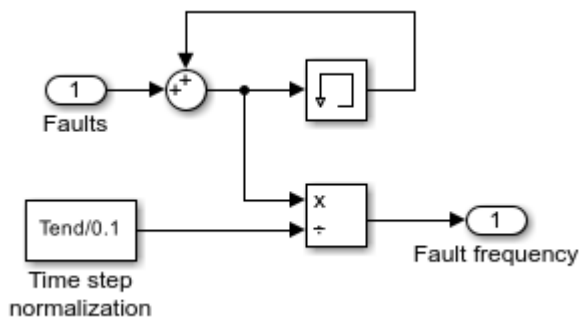
The classifier uses matrix algebra to compare the inputs to the training data and expected clutch states for the transmission gear. Possible clutch faults are tracked as the product of three flags: if the drive ratio differs from the expected drive ratio for the gear command, if the drive ratio matches one or more drive ratios from the training data, and if the clutch states from the training data differ from the commanded clutch state. This product is summed and normalized for all states in the training data. The result is compared to the gear clutch command to determine if the unexpected clutch state is an unlocked or locked clutch.

The Fault Classifier subsystem is configured to discretely execute and update the prediction every 0.1 seconds.



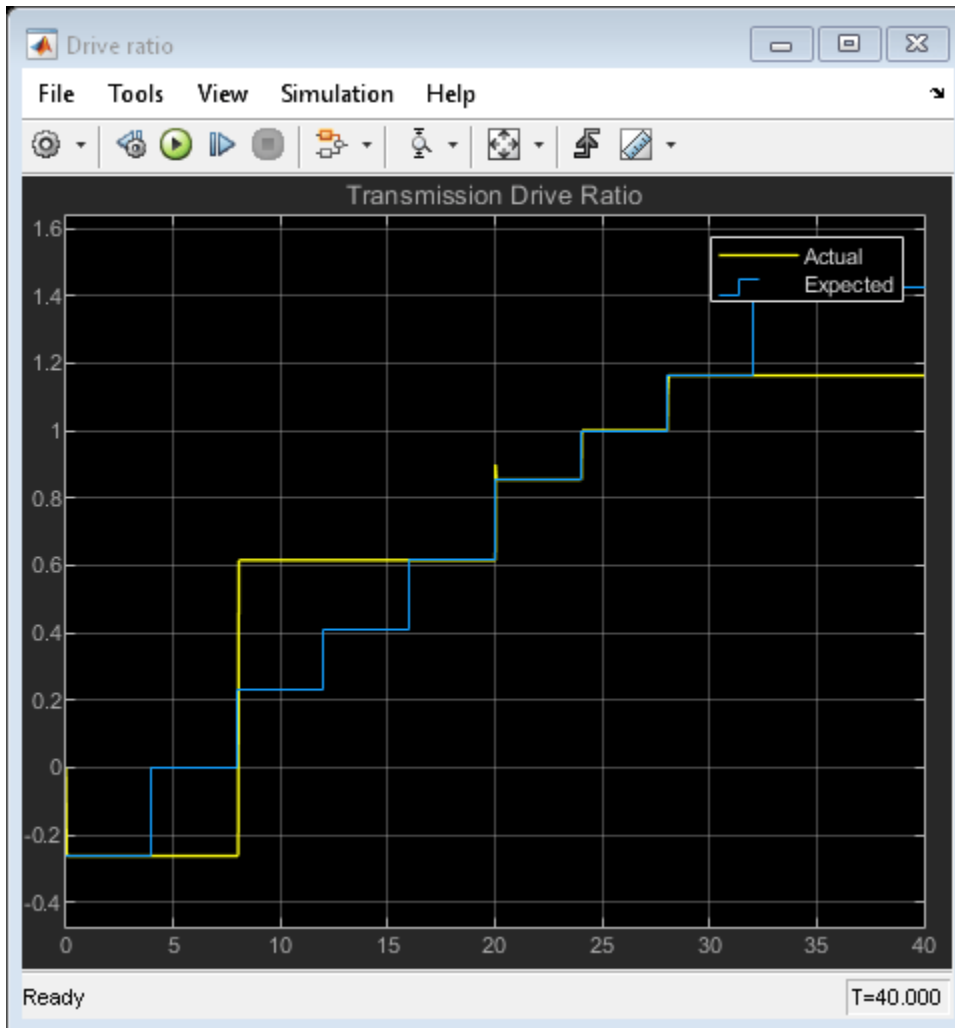
Prediction Update

The Prediction update subsystem adds the fault prediction of the current time step to the running prediction from previous time steps. The final running prediction is normalized by the total number of times that the Prediction update subsystem will run in the simulation, which equals the simulation Stop Time / 0.1 seconds.



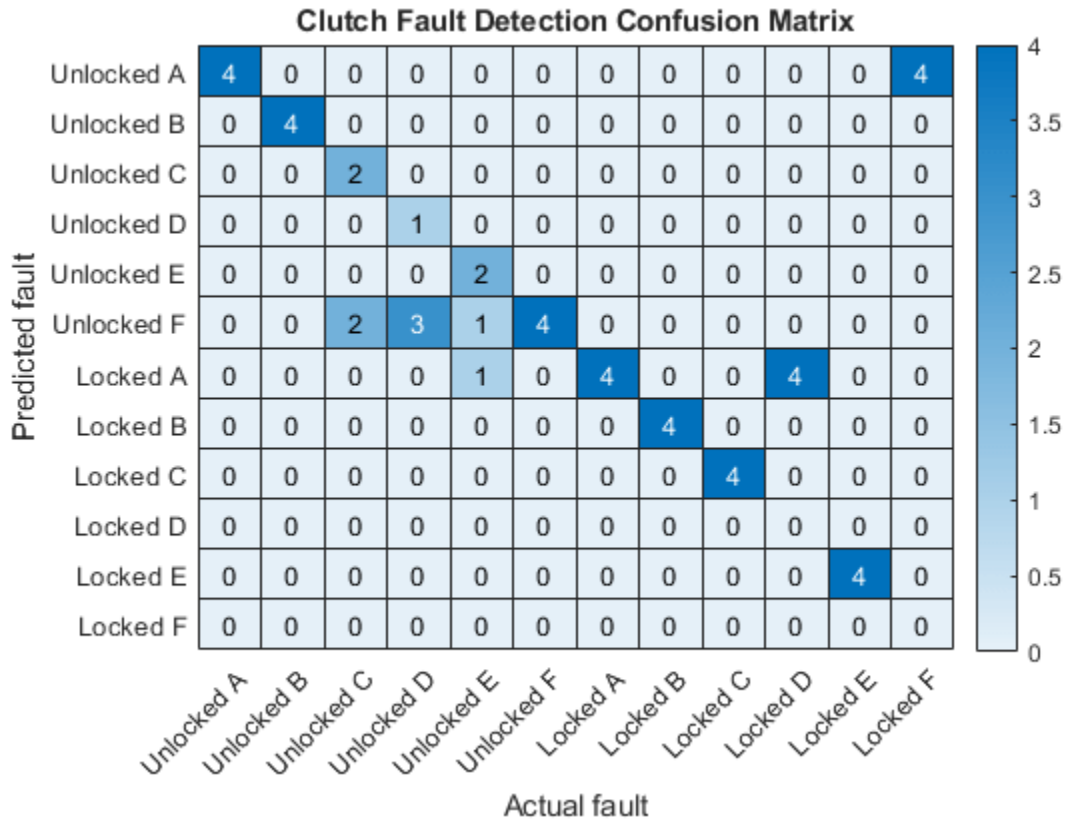
Simulation Results from Scope

This scope compares the actual and expected drive ratios when clutch A has a locked fault. This example defines the transmission ratio as the output velocity divided by the input velocity.



Results from Multiple Simulations

The plot below shows the confusion matrix of the transmission fault classifier for different test scenarios. To generate the plot, this example injects each fault into the transmission, adjusts scenario conditions, simulates the model, and records the fault class that receives the highest probability-like score from the classifier. These test scenarios vary the transmission base shaft velocity and follower shaft load for each fault class. The model takes advantage of Fast Restart mode to simulate the test cases more quickly. This example makes the load damping coefficient a run-time parameter so it can change the parameter value while the model is in Fast Restart. This basic classifier has varying accuracy for the different faults. Adjusting the sensors and fault prediction algorithm used in this model and simulating additional training scenarios may improve the results.

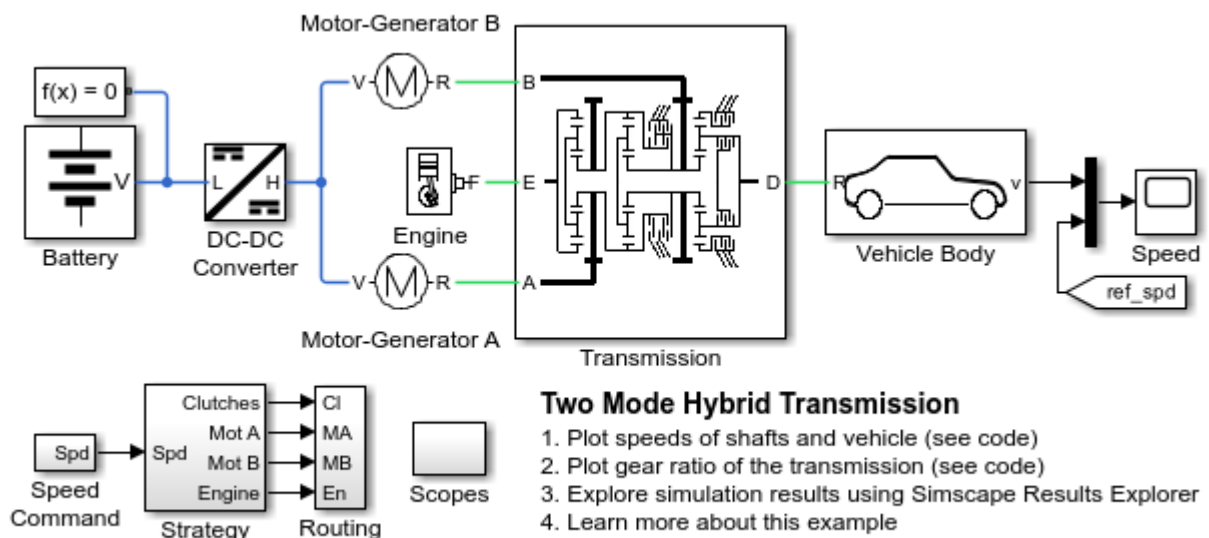


Two Mode Hybrid Transmission

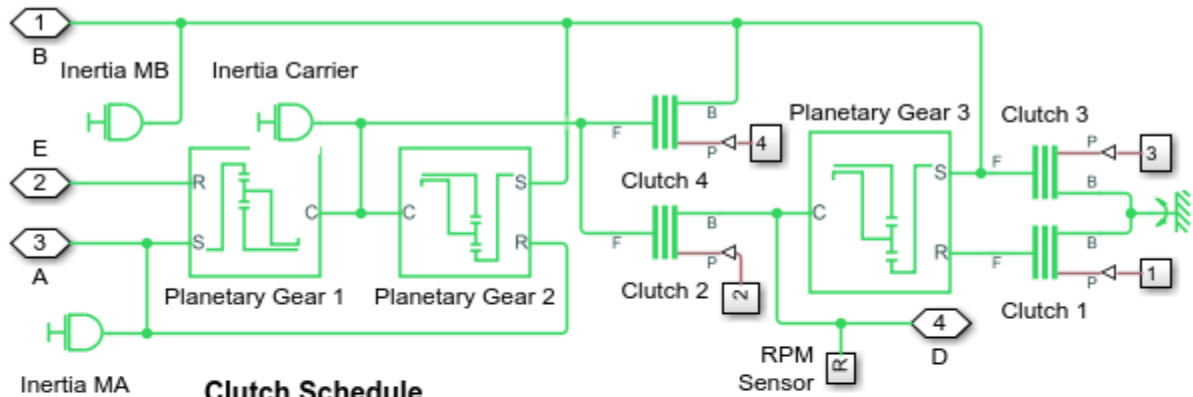
This example shows the basic architecture of a two mode hybrid transmission. It consists of three planetary gear sets and four clutches. This combination permits four fixed gear ratios plus two power-split modes. The power split modes are used to transition between fixed gear ratios and for heavy acceleration/deceleration. The fixed ratios help with efficiency when cruising. For the first power split (input-split regime), only Clutch 1 is engaged. For the second power split (compound-split regime), only Clutch 2 is engaged. Engaging two clutches simultaneously removes one degree of freedom and hence results in a fixed ratio.

In this example, the Strategy subsystem defines a set of rules that determine when to change between transmission modes during a steady acceleration profile. A full vehicle control strategy would also need to include control of electrical power as a function of speed, driver demand and battery state.

Model



Transmission Subsystem



Clutch Schedule

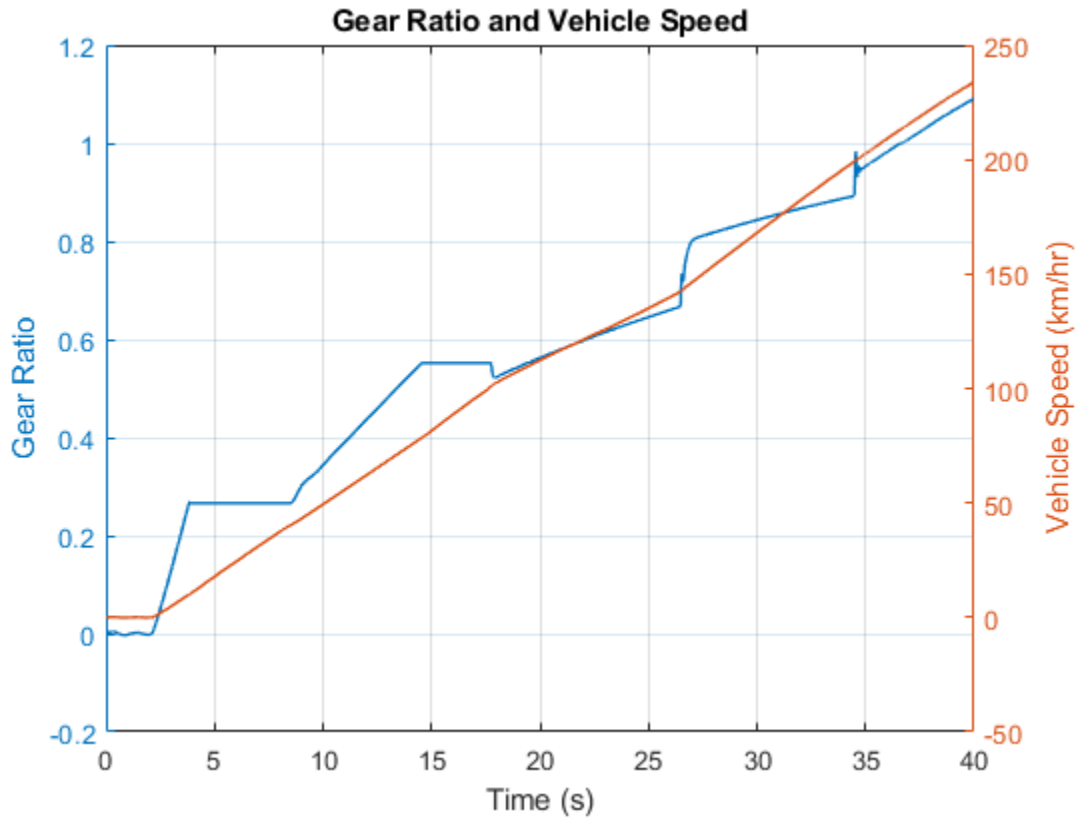
Gear/Mode	CI 1	CI 2	CI 3	CI 4
Input Split	1	0	0	0
Compound Split	0	1	0	0
1	1	0	0	1
2	1	1	0	0
3	0	1	0	1
4	0	1	1	0

Simulation Results from Simscape Logging

The plot below shows the speeds of the engine, motor-generator shafts, and vehicle speed as it progresses through each of the modes of a two mode hybrid powertrain. The modes include fixed gears and split modes where the flow of power is split at the input to the powertrain or both at the input and output (compound split).



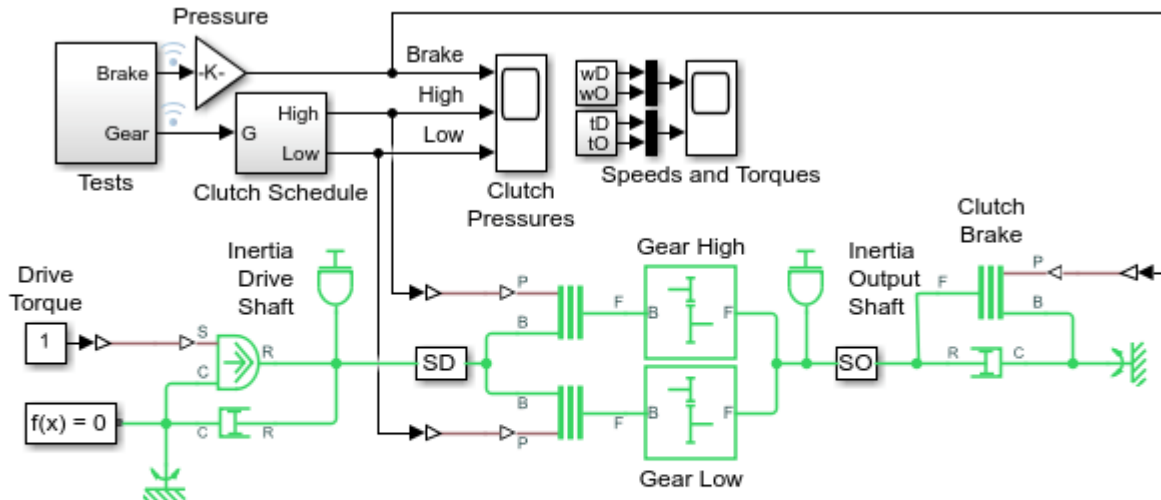
The plot below shows the gear ratio of the two mode hybrid powertrain transmission. The two electric motors enable the set of planetary gears to act as a continuously variable transmission.



Two-Speed Transmission

This example shows a simple two-speed transmission. Two clutches control which gear is selected. A brake is connected to the output shaft and can be controlled independently.

Model



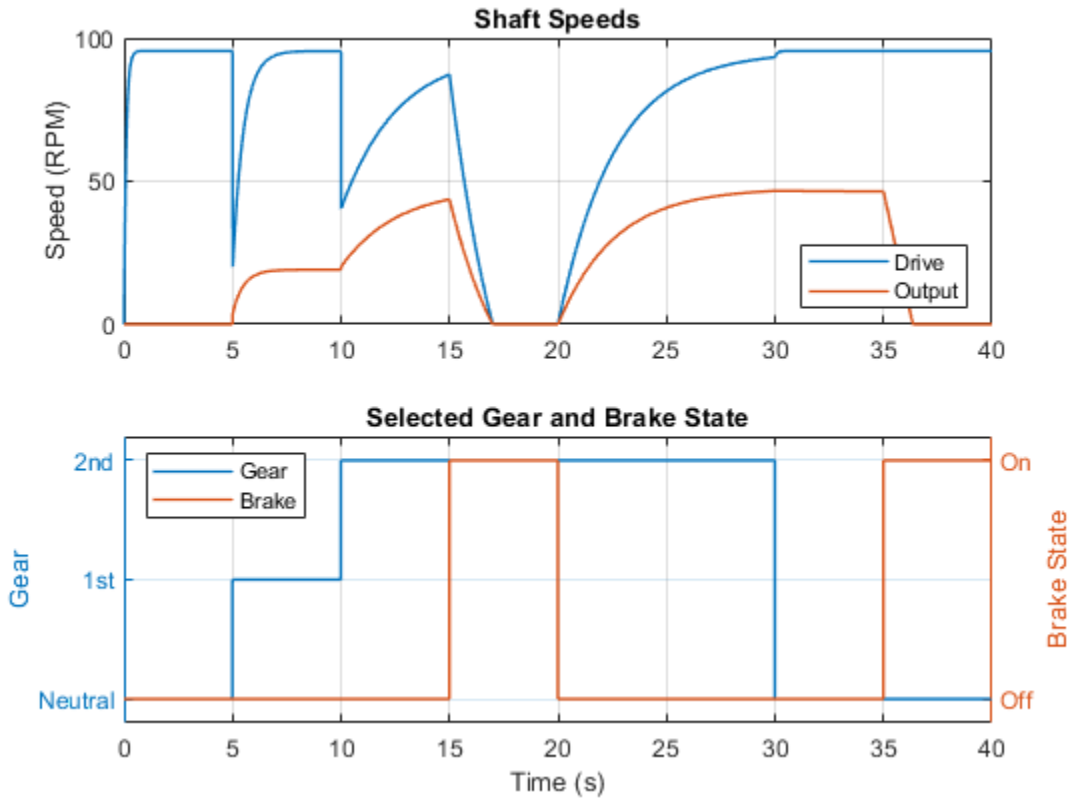
Two-Speed Transmission

1. Plot shaft speeds (see code)
2. Explore simulation results using Simscape Results Explorer
3. Learn more about this example

Copyright 2003-2022 The MathWorks, Inc.

Simulation Results from Simscape Logging

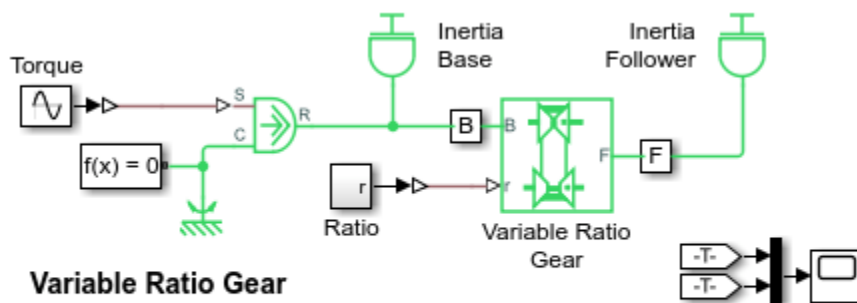
The plots below show the speeds of the drive shaft and output shaft for a two-speed transmission. The speed of the shafts changes as the clutches for the gears and the brake are engaged and disengaged.



Variable Ratio Gear

This example shows a variable ratio gear coupling two inertias (shafts). The gear ratio between the follower (F) and the base (B) is steadily increased from 1:1 to 2:1. The ratio of the follower to base angular speed decreases with gear ratio, and conversely the ratio of the follower to base torque increases with gear ratio. The variable gear is implemented using the Variable Ratio Transmission block which includes compliance. Appropriate stiffness and damping values for the compliance will depend upon the levels of torque transmitted by the variable gear. If the compliance damping is unrealistically low, the system can develop high frequency oscillations that require small simulation time steps.

Model



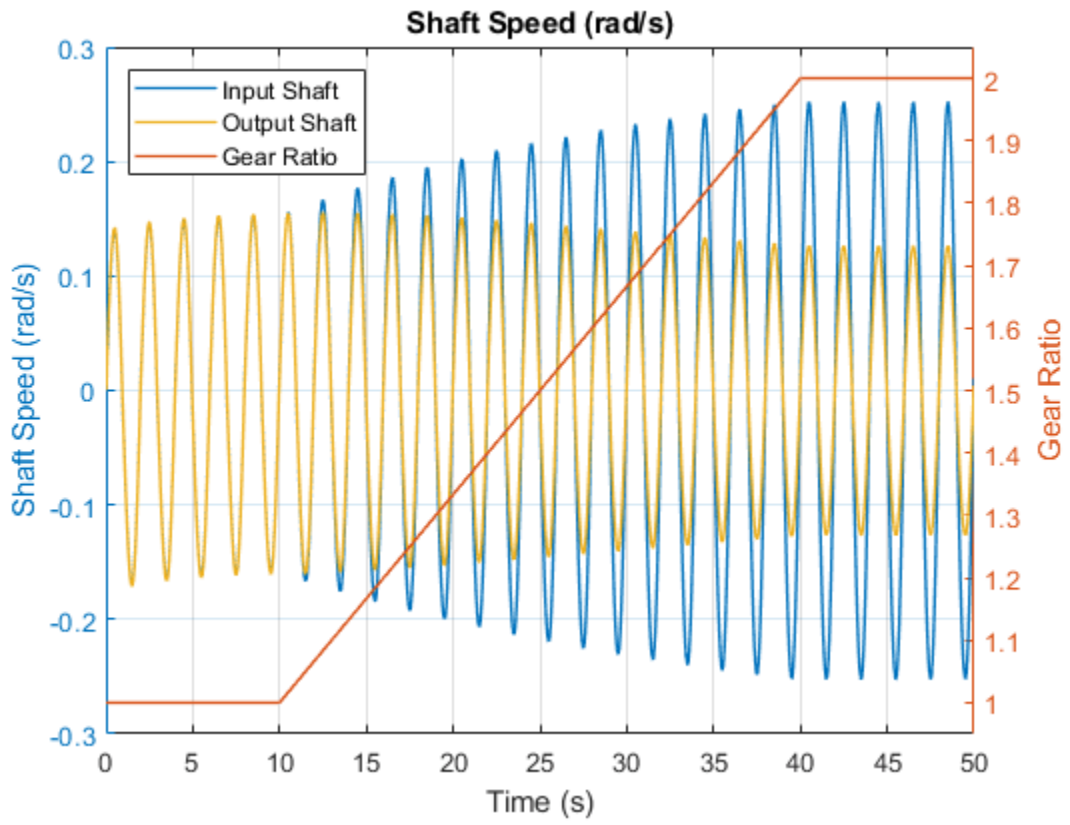
Variable Ratio Gear

1. Plot shaft speeds (see code)
2. Explore simulation results using Simscape Results Explorer
3. Learn more about this example

Copyright 2003-2022 The MathWorks, Inc.

Simulation Results from Simscape Logging

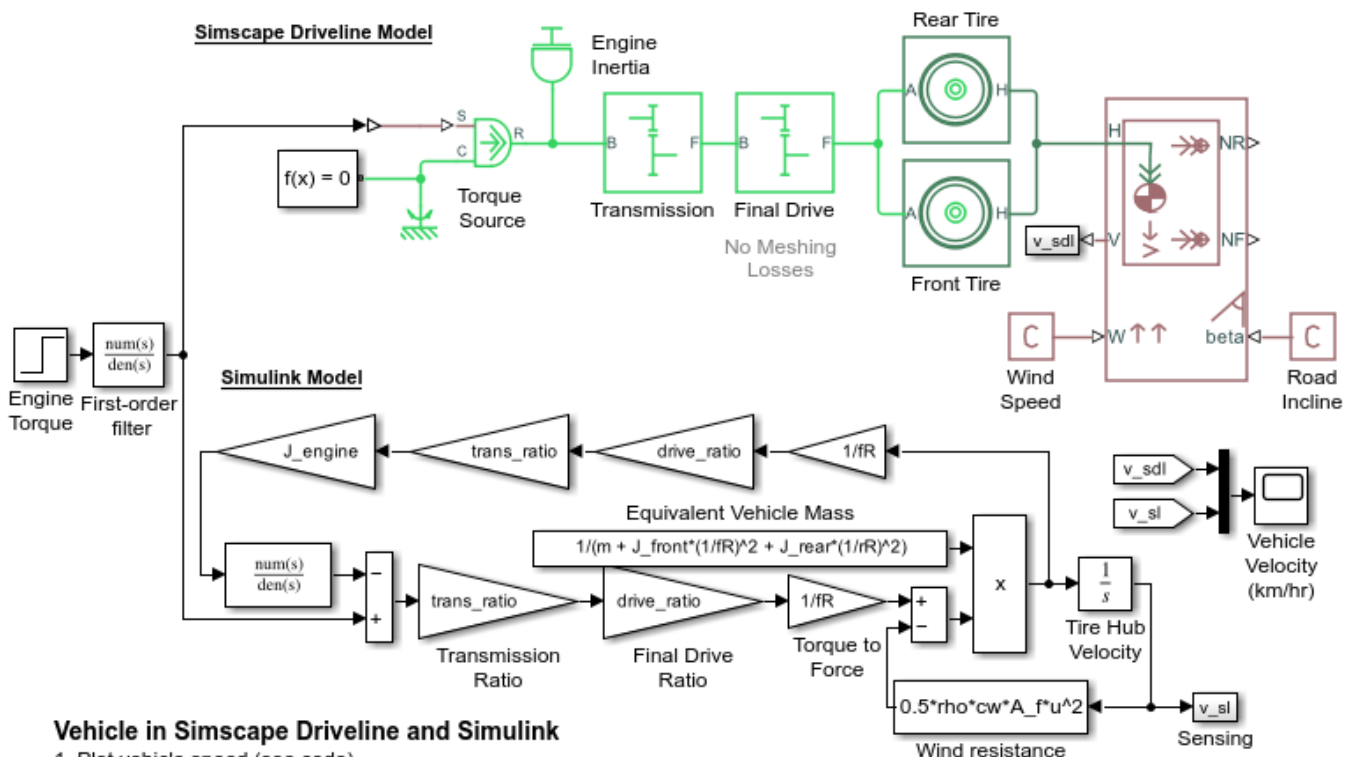
The plot below shows the input and output shaft speeds for a variable ratio gear that is driven by an oscillating torque source. As the ratio increases, the effective inertia decreases and the input shaft maximum speed increases.



Vehicle in Simscape Driveline and Simulink

This model shows two equivalent simplified vehicles modeled in Simscape™ Driveline™ and Simulink®. The simulation results are identical, and the Simscape Driveline model is easily extensible to include different effects and a higher level of modeling fidelity. Meshing losses in the gears and more detailed tire modeling can be added without introducing algebraic loops.

Model



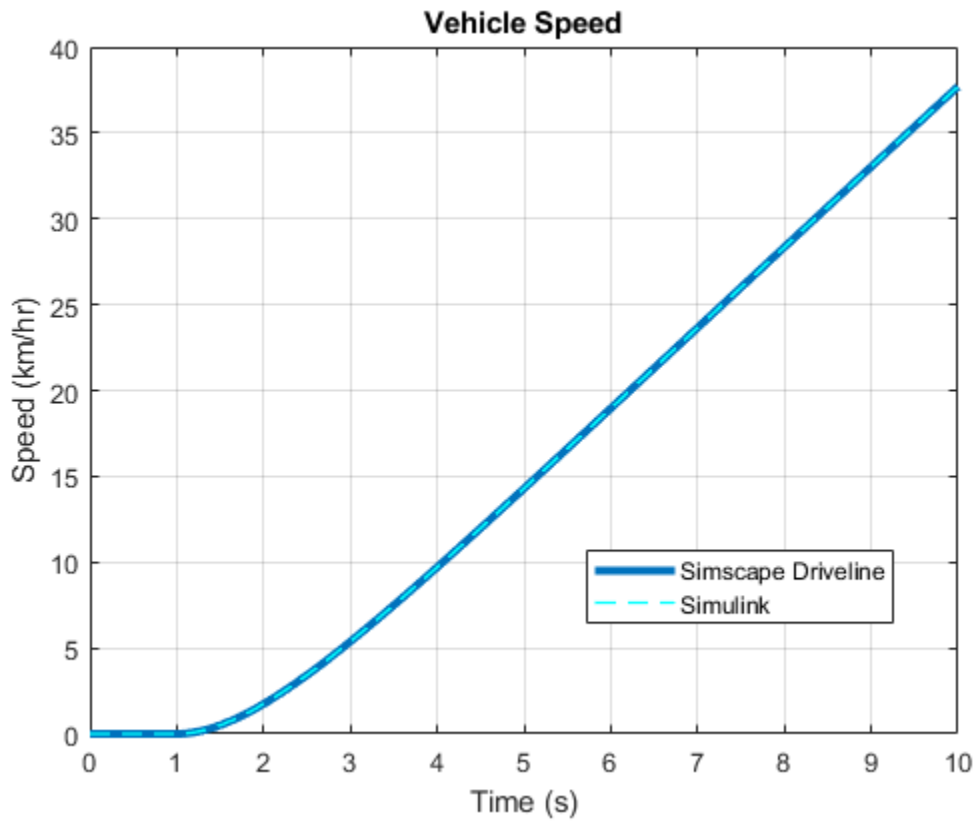
Vehicle in Simscape Driveline and Simulink

1. Plot vehicle speed (see code)
2. Configure friction in Final Drive gear of Simscape Driveline model:
No Losses, Constant Efficiency, Load-Dependent Efficiency
3. Explore simulation results using Simscape Results Explorer
4. Learn more about this example
5. Learn more about modeling physical networks

Copyright 2004-2022 The MathWorks, Inc.

Simulation Results from Simscape Logging

The plot below compares the results of the Simscape Driveline and Simulink models. When the models are equivalent, the results are identical. Meshing losses and other physical effects can be easily added to the Simscape Driveline model by enabling them in blocks.

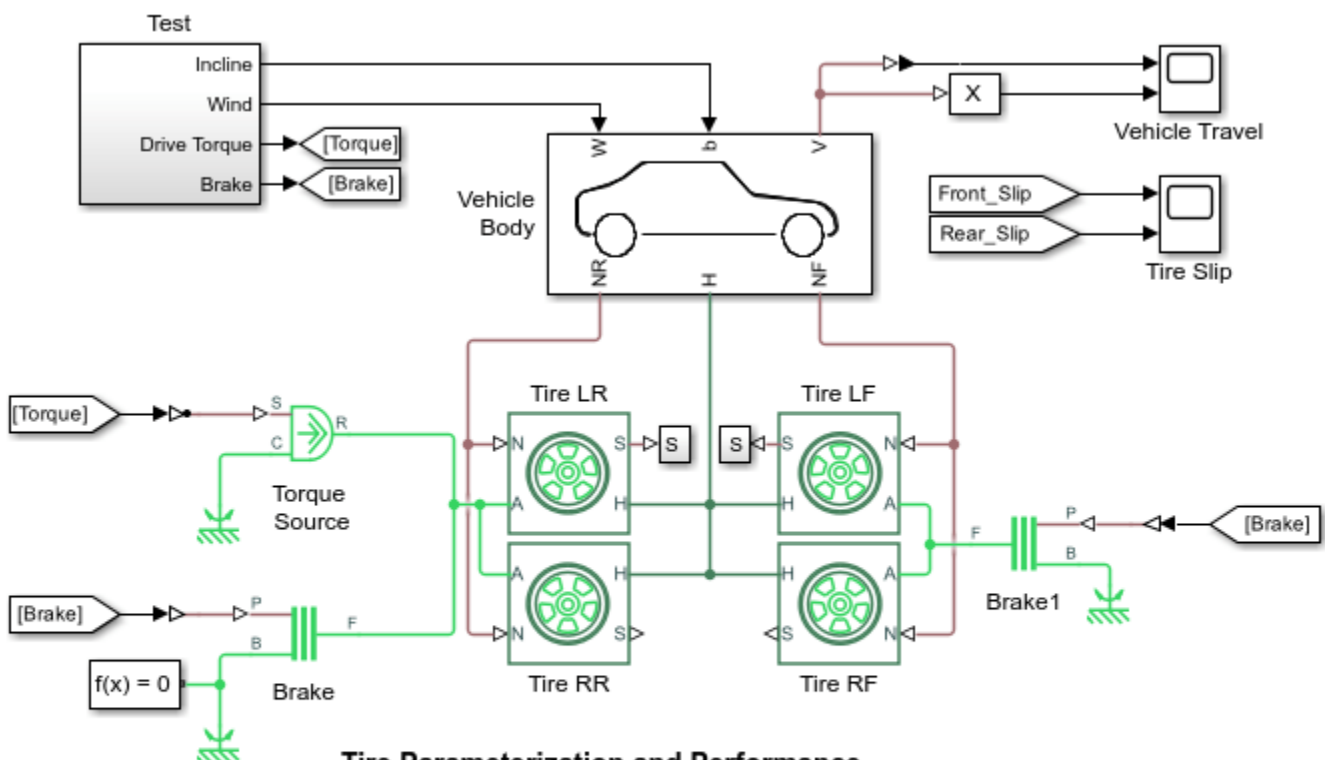


Tire Parameterization and Performance

This example shows how to use Simscape™ Driveline™ to fit tire Magic Formula coefficients to experimental data and to measure the simulated stopping distance of a vehicle supported by the tires. The example uses the `sdUtility.tirread` function to load initial tire parameters from a .TIR file. Then, the example uses the MATLAB® optimization function `fminsearch` with a test harness model to fit the tire coefficients to the experimental data. Other products available for performing this type of parameter fitting with Simscape Driveline models are the Optimization Toolbox™ and Simulink® Design Optimization™. These products provide predefined functions to manipulate and analyze blocks using GUIs or a command line approach. This example uses Fast Restart to quickly simulate tire response for different parameters during the optimization.

Model

The model contains a simple vehicle body supported by four tires. An Ideal Torque Source and two Disk Friction Clutch blocks specify the wheel torque conditions. The test scenario in this example accelerates and then brakes the vehicle.



Tire Parameterization and Performance

1. Fit Magic Formula parameters to experimental data using test harness (see code)
2. Estimate stopping distance (see code)
3. Explore simulation results using Simscape Results Explorer
4. Learn more about this example

Copyright 2022 The MathWorks, Inc.

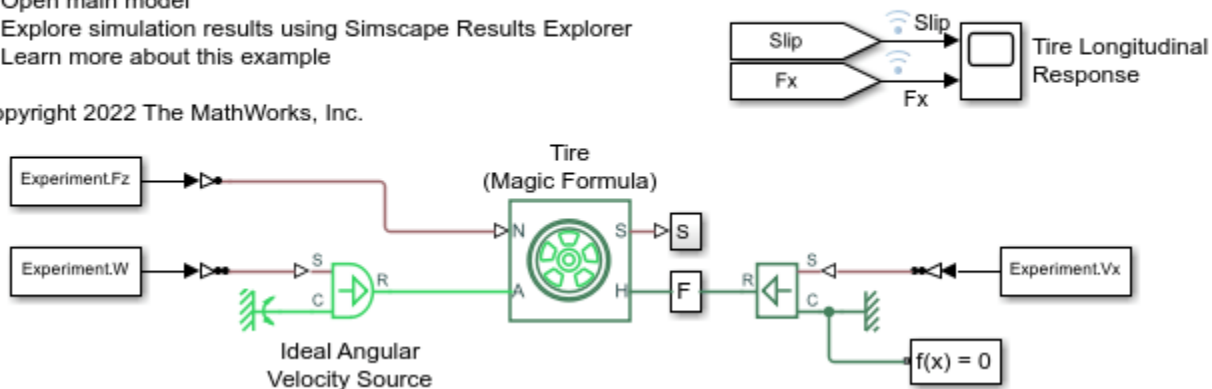
Optimization Procedure and Results

This example considers 14 Magic Formula coefficients that affect the tire longitudinal force-slip behavior. The parameter fit procedure uses a test harness model that drives a tire with the same vertical load, rotational velocity, and longitudinal velocity as the experimental data. An Ideal Force Sensor measures the longitudinal force of the tire. The optimization objective function considers the least squares error between the simulated and experimentally measured longitudinal forces. The model uses Fast Restart to keep the model compiled between optimization iterations.

Tire Parameterization and Performance Harness

1. Fit Magic Formula parameters to experimental data (see code)
2. Open main model
3. Explore simulation results using Simscape Results Explorer
4. Learn more about this example

Copyright 2022 The MathWorks, Inc.



This example fits the Magic Formula longitudinal coefficients to experimental data provided in A. Ortiz, J. Cabrera, A. Guerra, and A. Simon, An easy procedure to determine Magic Formula parameters: A comparative study between the starting value optimization technique and IMMA optimization algorithm *Vehicle System Dynamics*. Taylor & Francis, 2006. The optimization procedure initializes the 14 coefficients with guessed values. The first figure below shows the mismatch between the initial tire longitudinal force-slip characteristics and the experimental data. The second figure shows closer agreement after the parameter optimization. Since `fminsearch` is an unconstrained nonlinear optimizer that locates a local minimum of a function, varying the initial estimate will result in a different solution set.

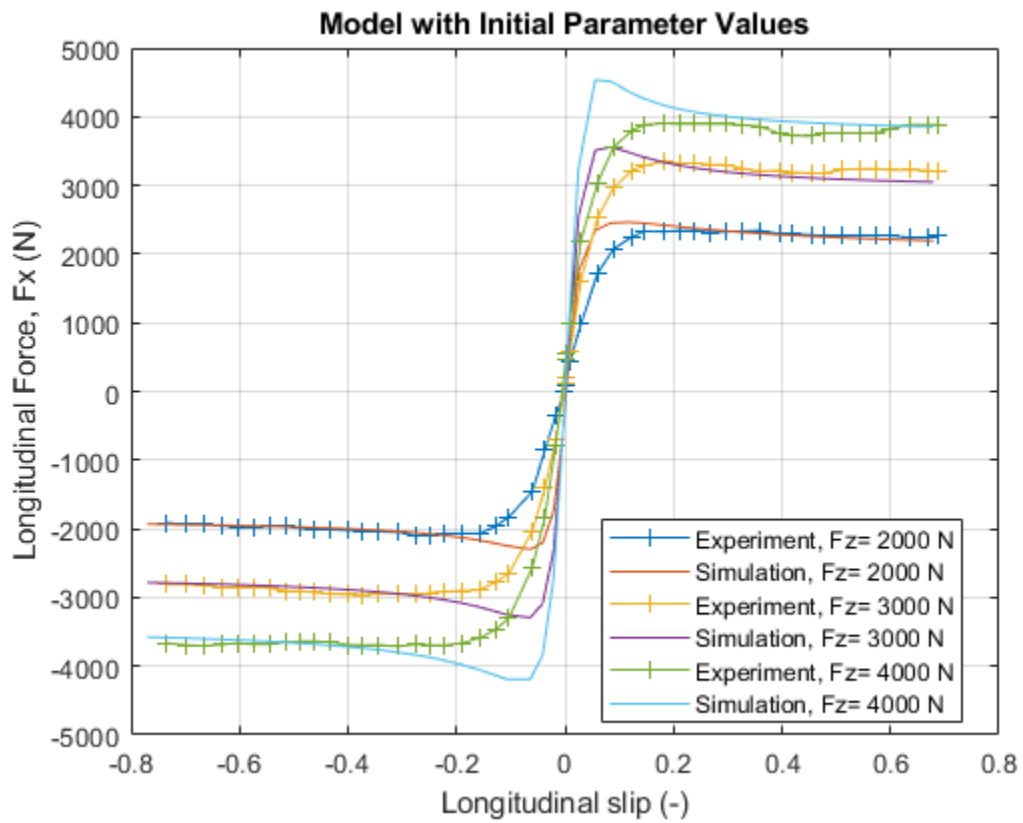
Initial coefficient values are:

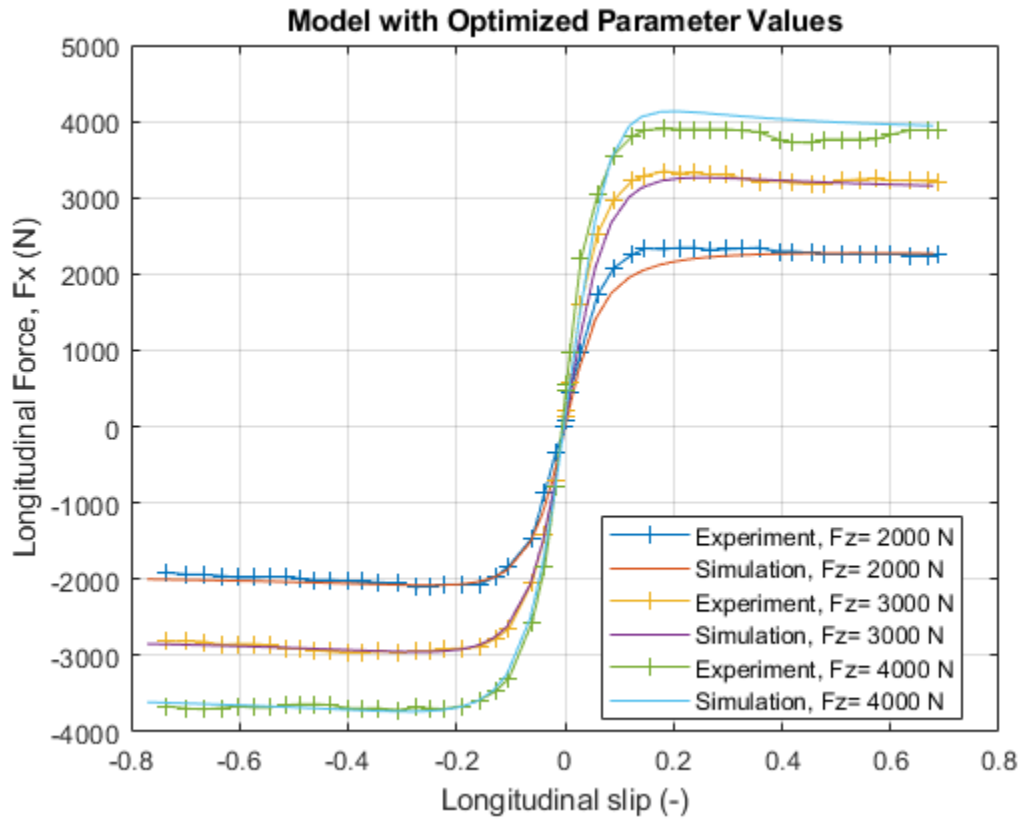
```
P_Cx1 = 1.397
P_Dx1 = 1.1
P_Dx2 = -0.182
P_Ex1 = -0.459
P_Ex2 = -1.5
P_Ex3 = 0.006
P_Ex4 = -0.91
P_Kx1 = 38
P_Kx2 = 2.031
P_Kx3 = -0.6
P_Hx1 = -0.0017
P_Hx2 = 0.003
P_Vx1 = 0.057
P_Vx2 = -0.03
```

Optimized coefficient values are:

```
P_Cx1 = 1.26268
P_Dx1 = 0.977901
```

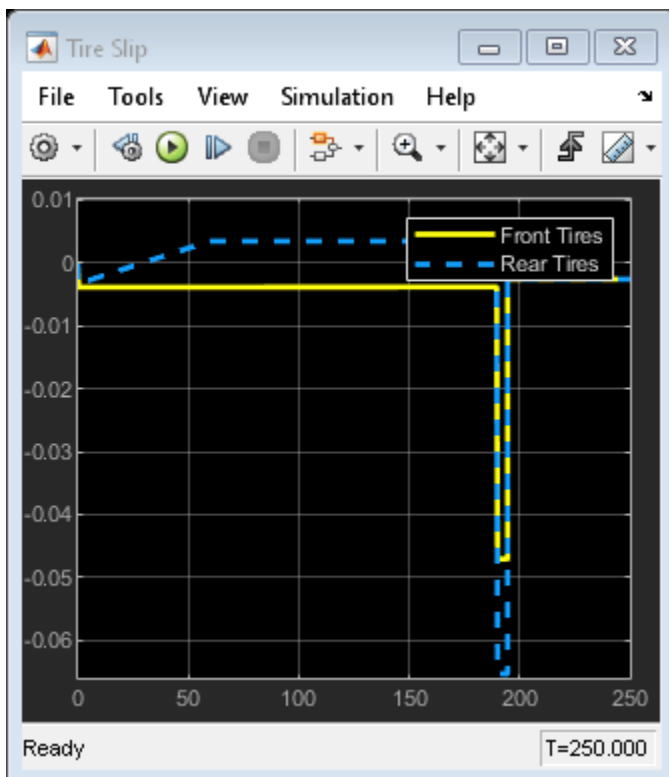
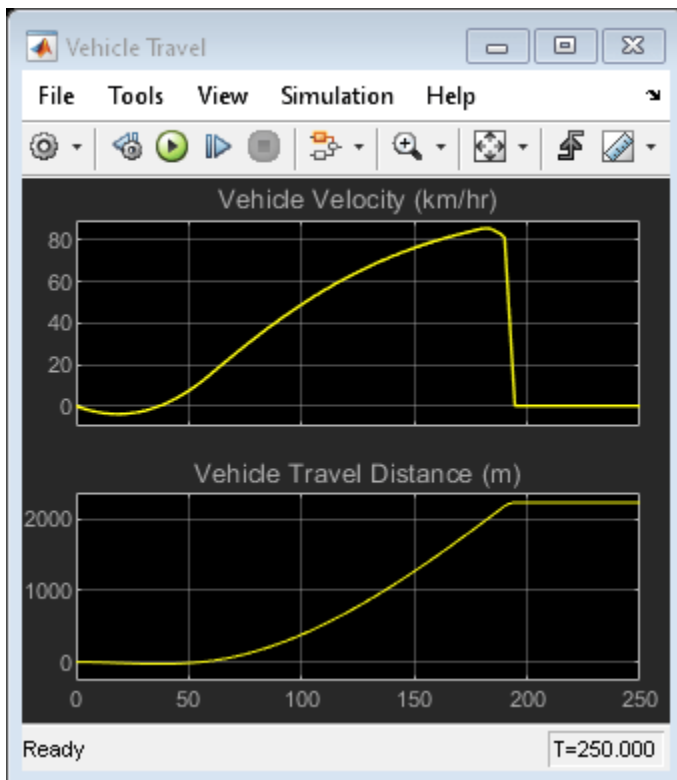
P_Dx2 = -0.209517
P_Ex1 = -0.487528
P_Ex2 = -1.63709
P_Ex3 = 0.00638911
P_Ex4 = -1.06055
P_Kx1 = 14.2857
P_Kx2 = 2.12097
P_Kx3 = -0.539739
P_Hx1 = -0.00208473
P_Hx2 = 0.0032325
P_Vx1 = 0.063874
P_Vx2 = -0.0273162





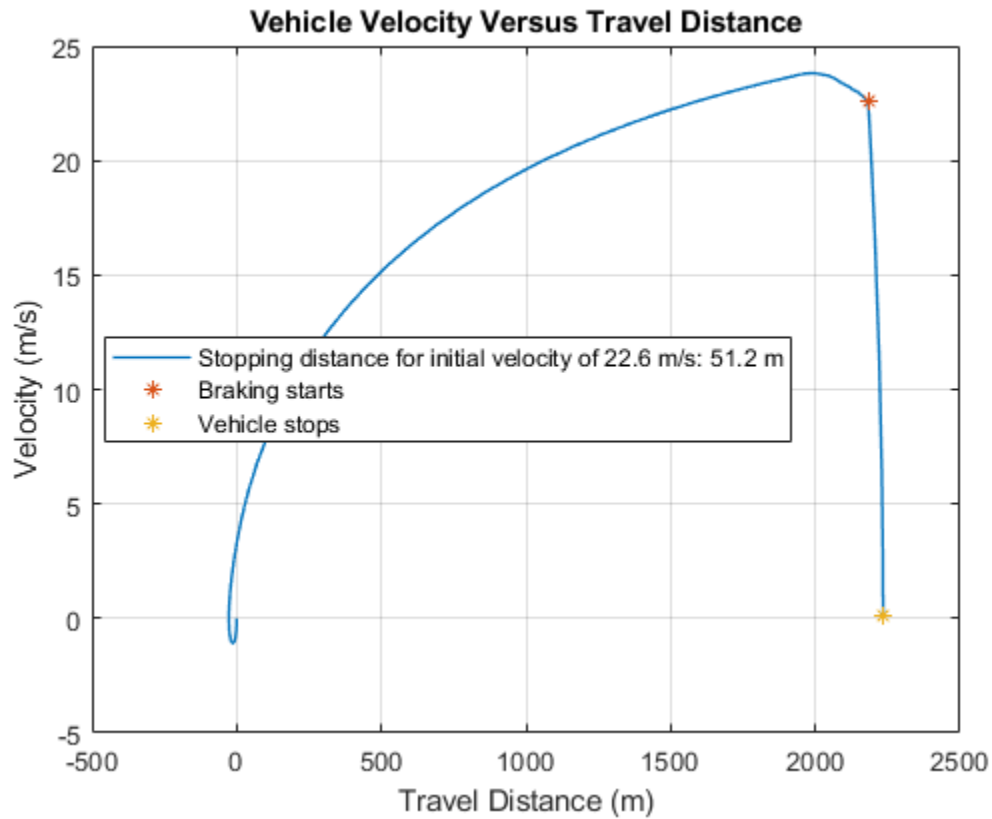
Simulation Results from Scopes

The scopes show vehicle velocity and travel distance, as well as front and rear tire slip. Tire slip spikes when the brakes are first applied.



Simulation Results from Simscape Logging

This plot shows the vehicle velocity versus travel distance. The markers indicate when the brake is first engaged and when the vehicle comes to a stop. With these vehicle and tire parameters, the vehicle has a stopping distance of approximately 44 meters when braking starts at a vehicle velocity of 21 meters per second.



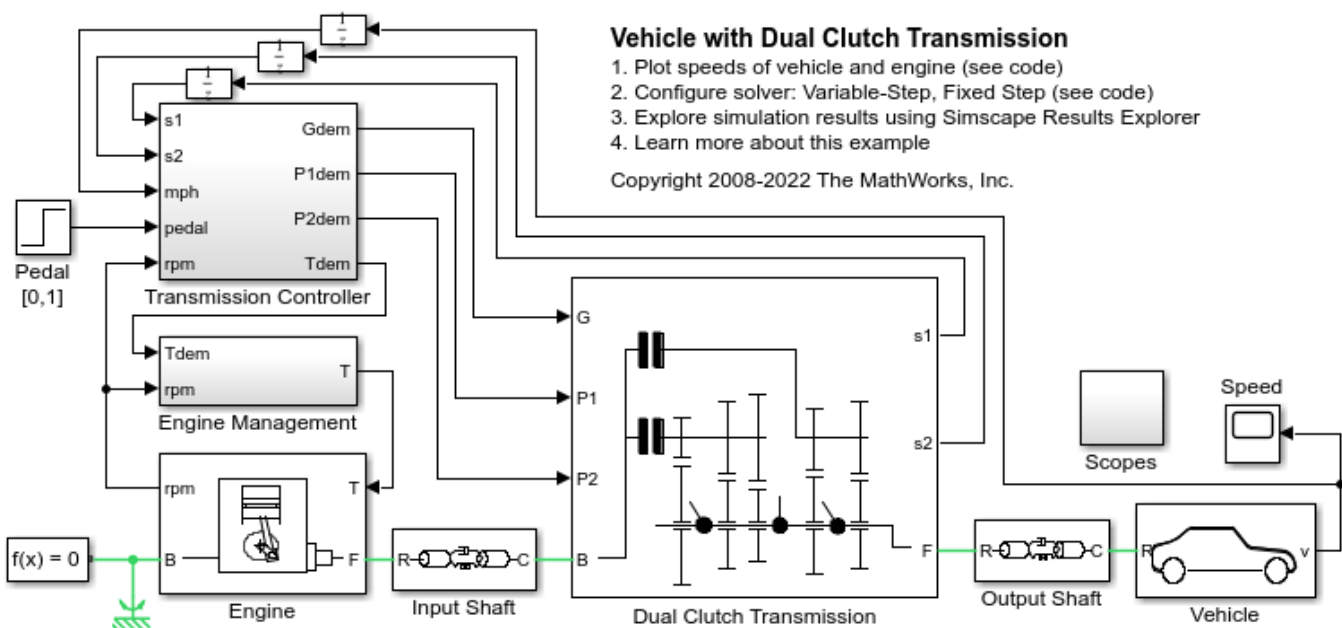
Vehicle with Dual Clutch Transmission

This example shows a vehicle with a five-speed automatic dual-clutch transmission. The transmission controller converts the pedal deflection into a demanded torque. This demanded torque is then passed to the engine management. The pedal deflection and the vehicle speed are also used by the transmission controller to determine when the gear shifts should occur. Gear shifts are implemented via the two clutches, one clutch pressure being ramped up as the other clutch pressure is ramped down. Gear pre-selection via dog clutches ensures that the correct gear is fully selected before the on-going clutch is enabled.

You can set the transmission controller to work for up to 10 gears. The base workspace variable 'GrMax' sets the maximum gear of the transmission controller. The transmission controller in this example uses a maximum gear of 5.

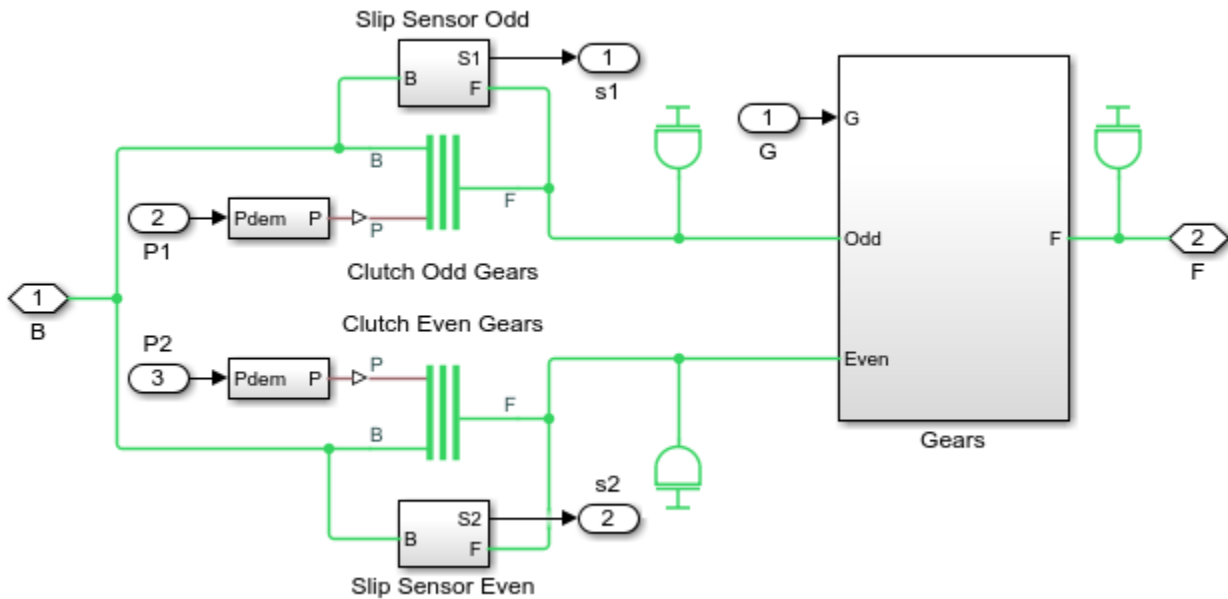
This model can be configured for fixed-step simulation, making it suitable for hardware-in-the-loop testing.

Model



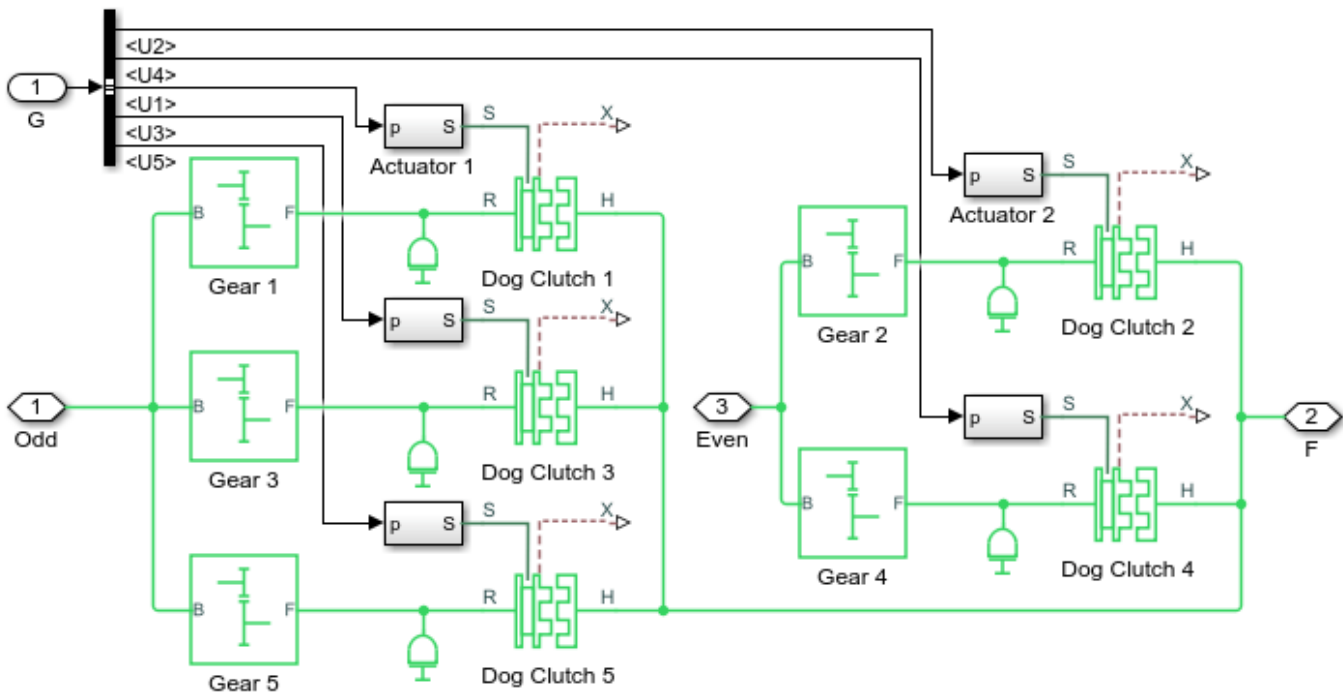
Dual Clutch Transmission Subsystem

Two Disk Friction Clutch components from Simscape™ Driveline™ are connected in parallel, modeling the two clutches in a dual-clutch transmission. Only one of these clutches is engaged at a time. Engaging and disengaging these clutches is how gear changes are made.



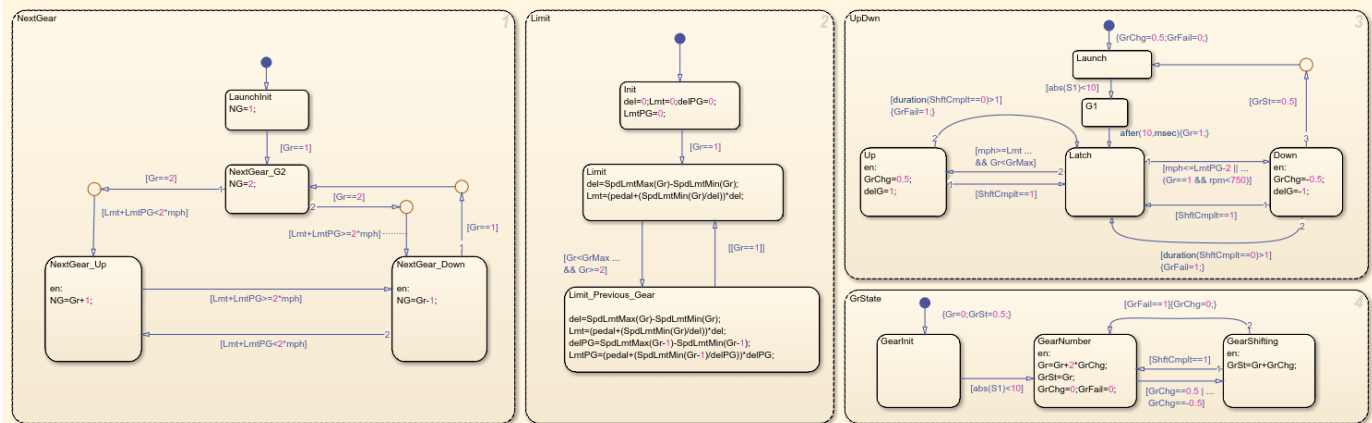
Gears Subsystem

Five Dog Clutch components from Simscape Driveline are connected between the follower shaft of each gear to the transmission output shaft. The controller preselects the next gear to which the transmission will shift by engaging the dog clutches. At any point, two of the dog clutches will be engaged, that for the current gear and that for the next gear to which the transmission will shift.



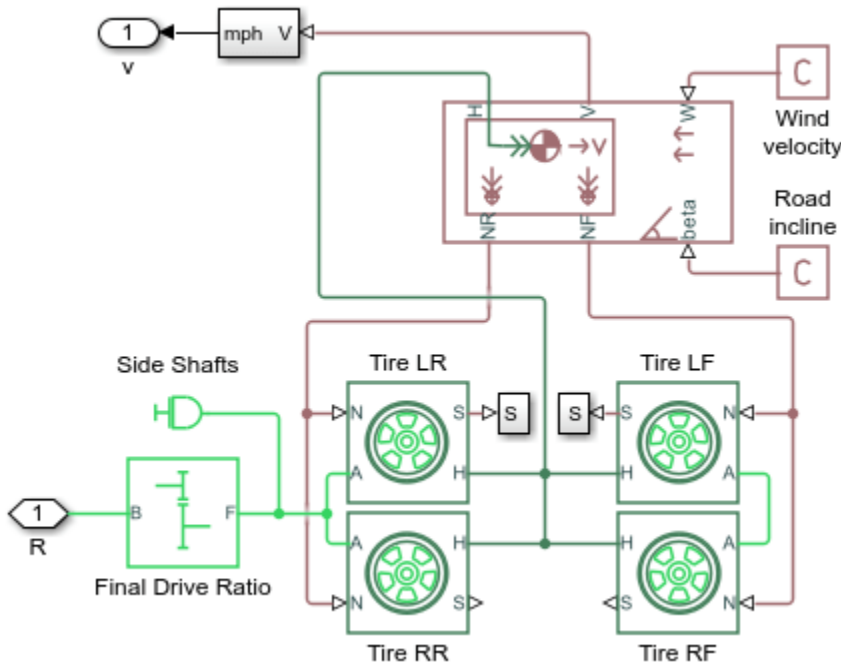
Transmission Controller

This subsystem implements a logic for auto transition of gear. The logic is implemented in Stateflow™, it has four parallel flows for gear up/down shift, velocity limit calculation for gear shifting, Next gear prediction based on limits and gear state to compute the current gear.



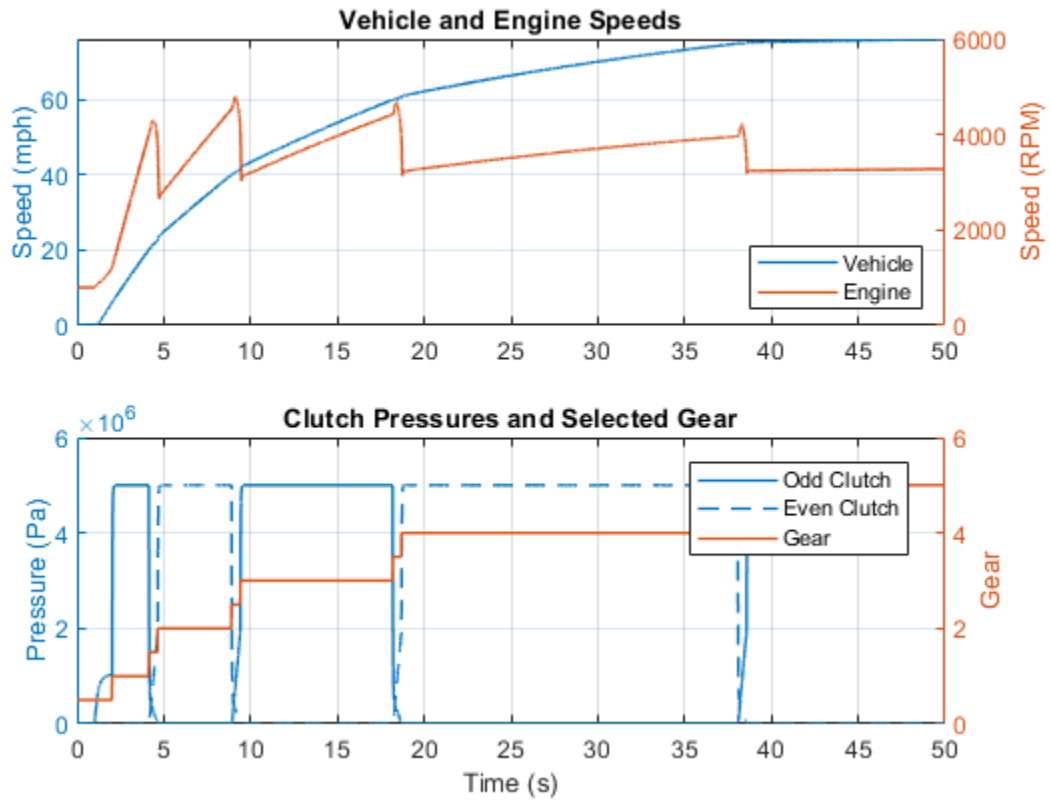
Vehicle Subsystem

Tire models from Simscape Driveline that are characterized by the Magic Formula are connected to a longitudinal model of the vehicle dynamics. The driveshaft is connected to the rear wheels. The normal force for each axle is used in the slip calculation for each wheel.



Simulation Results from Simscape Logging

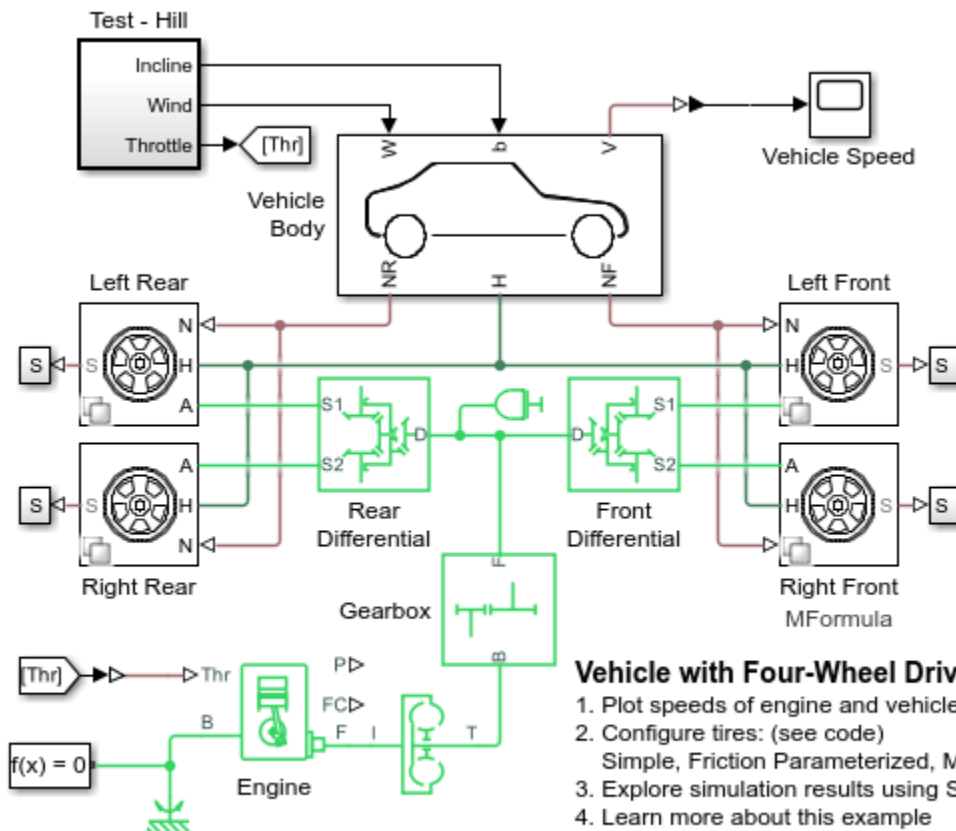
The plots below show the engine and vehicle speed as it accelerates through the first three gears in the dual-clutch transmission. The half-gear states indicate gear shifts are taking place by engaging and disengaging the dual clutches.



Vehicle with Four-Wheel Drive

This model shows a four-wheel drive vehicle starting from rest and ascending a 15 degree incline. Initially the vehicle rolls backwards until the engine develops sufficient torque to counter the slope. The tire compliance dynamics can be seen as the vehicle starts to accelerate. The model variant chosen for all of the tires can be set to the Simple, Friction Parameterized, or Magic Formula tire model using the hyperlinks in the model.

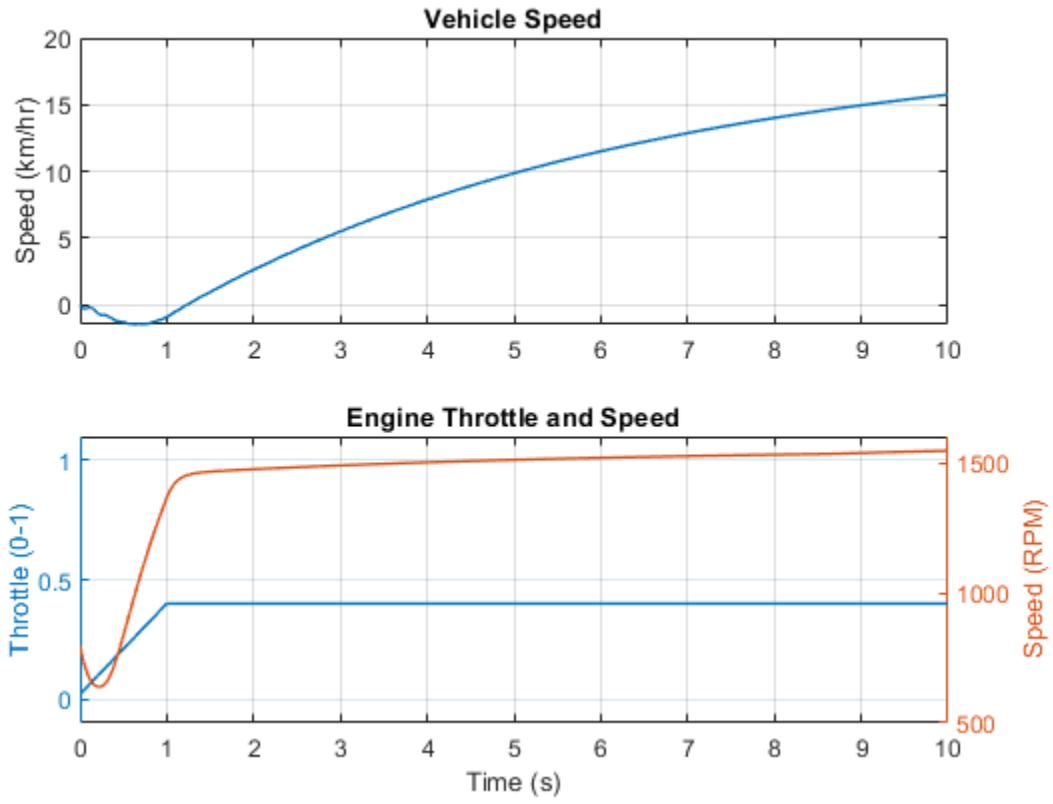
Model



Copyright 2004-2022 The MathWorks, Inc.

Simulation Results from Simscape Logging

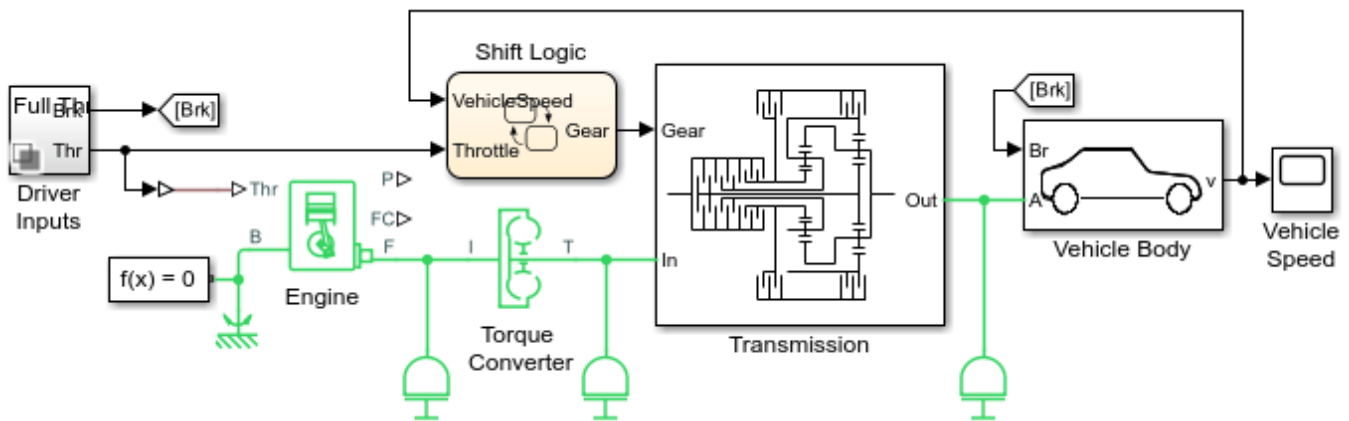
The plots below show the engine and vehicle speed. As the vehicle is starting at rest on an incline, it rolls backwards until the torque of the engine is enough to push the vehicle up the hill.



Vehicle with Four-Speed Transmission

This example shows a complete vehicle with Simscape™ Driveline™ components, including the engine, drivetrain, four-speed transmission, tires, and longitudinal vehicle dynamics. The transmission controller is implemented as a state machine in Stateflow®, selecting the gear based on throttle and vehicle speed.

Model

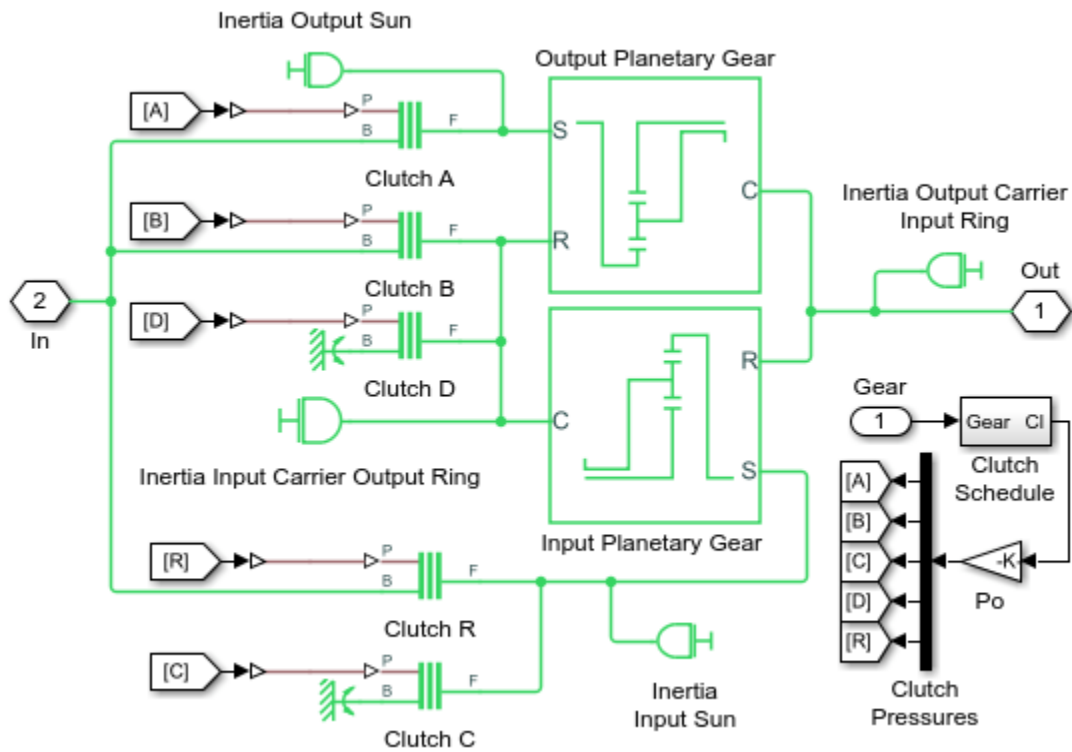


Vehicle with Four-Speed Transmission

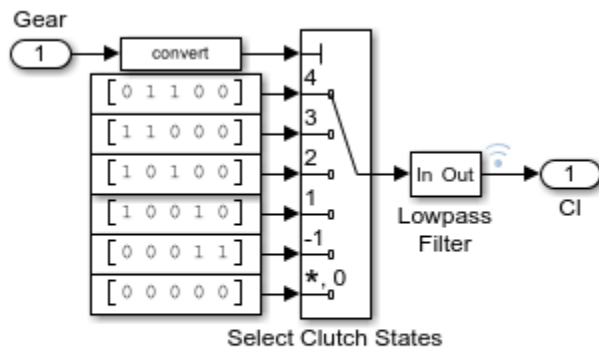
1. Plot speeds of shafts and vehicle (see code)
2. Explore simulation results using Simscape Results Explorer
3. Learn more about this example

Copyright 2003-2022 The MathWorks, Inc.

Transmission Subsystem



Clutch Schedule Subsystem



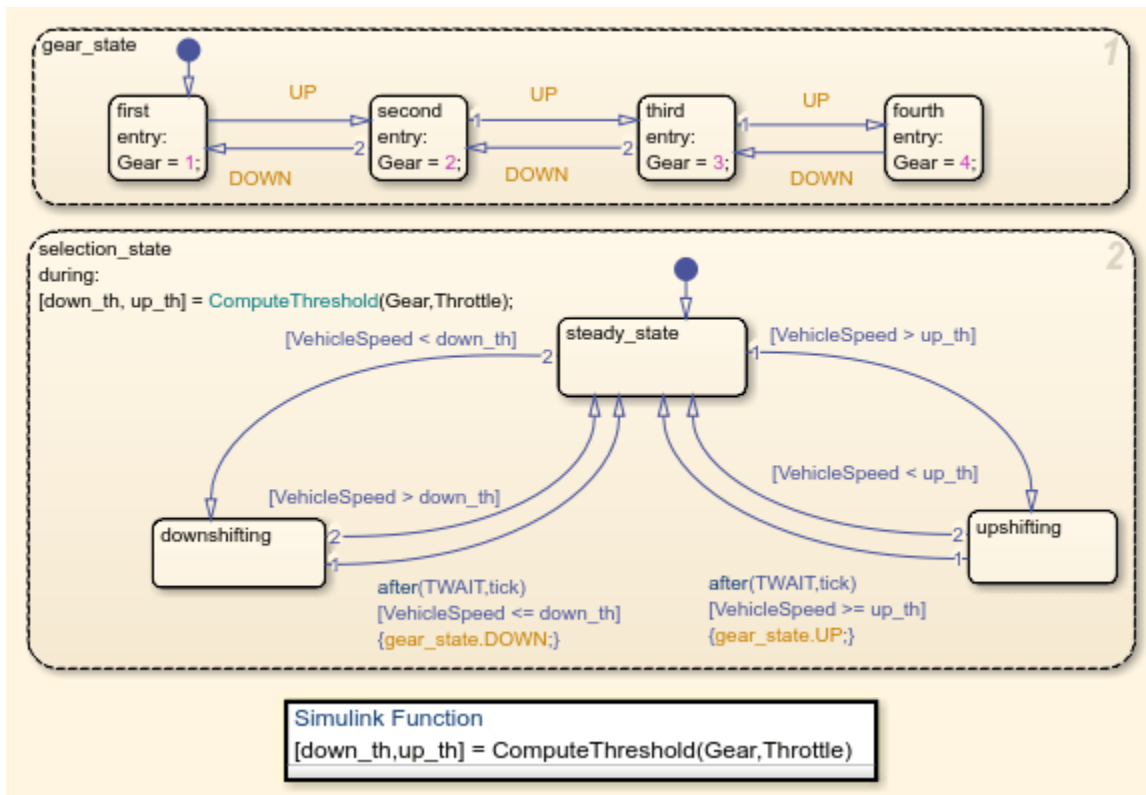
Clutch Schedule

Gear	A	B	C	D	R	Ratio
R	0	0	0	1	1	-g1
1	1	0	0	1	0	g2
2	1	0	1	0	0	$(g1+g2)/(1+g1)$
3	1	1	0	0	0	1
4	0	1	1	0	0	$g1/(1+g1)$

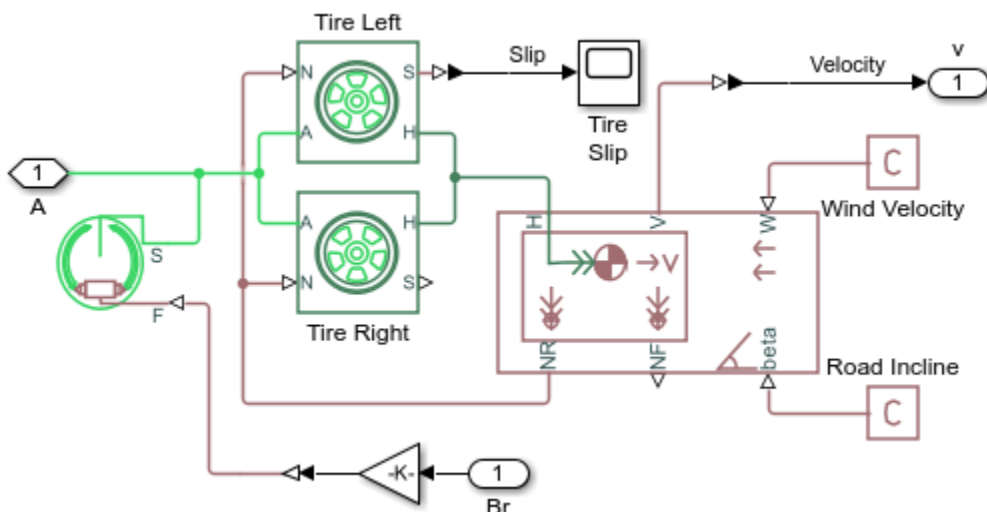
0 - Disengaged, 1 - Engaged

g1: Input planetary ring/sun gear ratio
g2: Output planetary ring/sun gear ratio

Shift Logic Subsystem

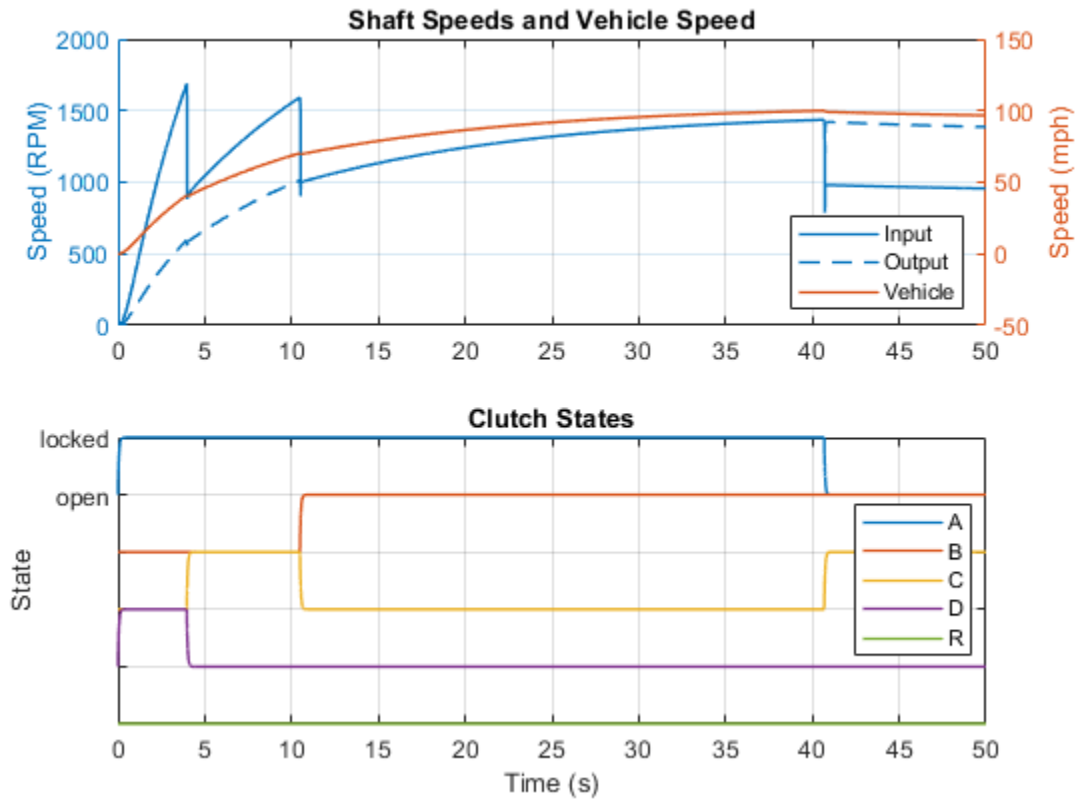


Vehicle Body Subsystem



Simulation Results from Simscape Logging

The plot below shows the input and output shaft speeds of the transmission as well as the vehicle speed. The clutch states are also plotted (locked or open), indicating the selected gear of the transmission.



Vehicle with Manual Transmission

This example shows a vehicle that has a four-speed manual transmission. The key elements of the transmission are four synchronizers. By engaging or disengaging these synchronizers and associated dog clutches, the transmission provides four ratios 3.581, 2.022, 1.384, and 1, respectively. The synchronizers are modeled using the Cone Clutch and Dog Clutch blocks.

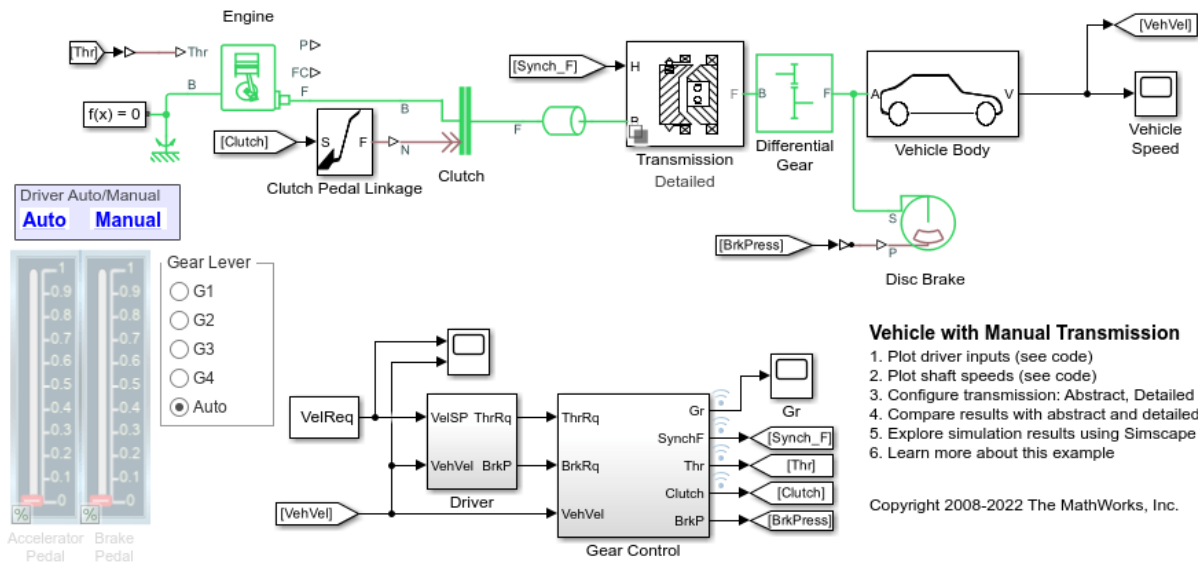
The model also provides an abstracted representation of the gearbox that is significantly less complex and thus simulates much faster. It uses simply a variable ratio gear to represent the various gear ratios in the detailed transmission, plus a clutch to represent neutral. By eliminating nearly all of the discontinuities, it is designed for hardware-in-the-loop simulation. To select this variant, use the hyperlinks in the model.

The model supports automatic and manual driver modes. In manual driver mode, you use Simulink Dashboard blocks to control the gearbox shifting and synchronization. Manual: To run the model in manual driver mode, click on the "Manual" hyperlink. Hyperlink activates the sliders, sets the simulation pacing to '1' and sets the simulation time to 'inf'. For the start put the Gear lever dashboard G1, Accelerator pedal and Brake pedal to '0' and start the simulation. During simulation, each gear shift can be a single gear shift or multiple gear shifts. Use the sliders to adjust the accelerator pedal and brake pedal. In manual mode gear shifting should be done based on vehicle speed as abrupt gear shifting can cause engine speed errors.

Auto: Auto is the default mode for simulation. To run the model in auto mode, click on the "Auto" hyperlink. Hyperlink de-activates the sliders for the pedals, it disables the simulation pacing and sets the simulation time to 100 sec which is the velocity profile time length in the signal editor. Velocity from different mat files can be loaded and simulation time needs to be changed accordingly. During simulation, a PI controller representing the driver adjusts the throttle and brake in response to a requested vehicle velocity provided by a Signal Editor block. An automatic gear shifter implemented in Stateflow™ shifts the gear based on gear speed limits, which are set in the Model pre-load function.

Control is set for 4 gear transmission in auto mode as default.

Model

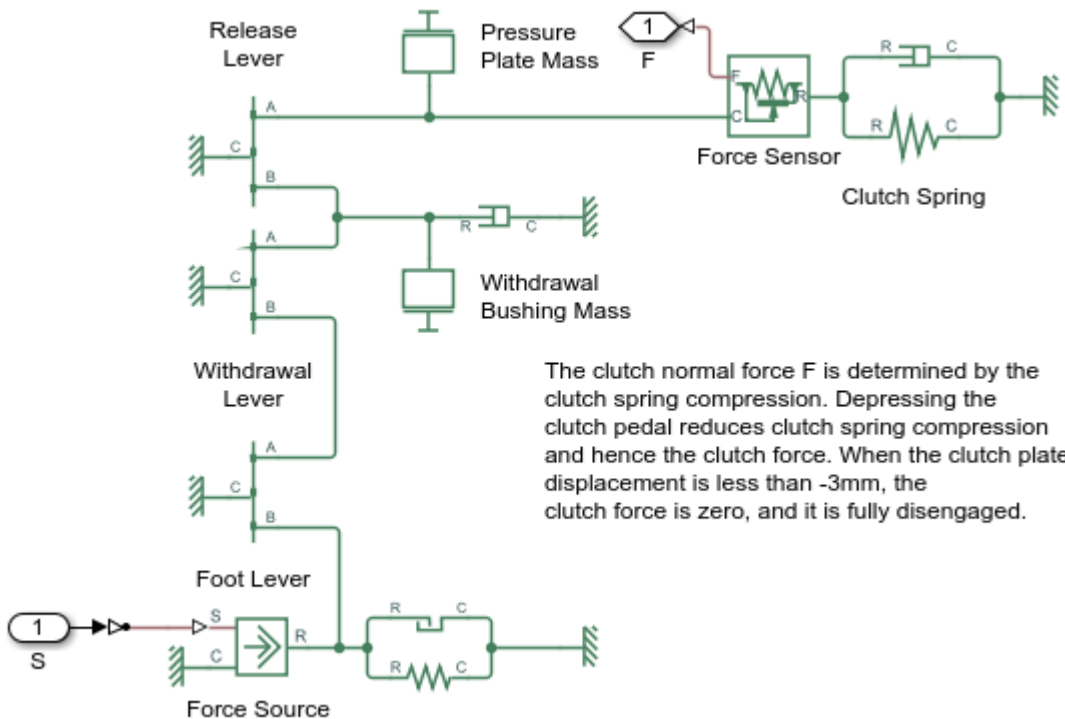


Vehicle with Manual Transmission

1. Plot driver inputs (see code)
2. Plot shaft speeds (see code)
3. Configure transmission: Abstract, Detailed
4. Compare results with abstract and detailed variants (see code)
5. Explore simulation results using Simscape Results Explorer
6. Learn more about this example

Copyright 2008-2022 The MathWorks, Inc.

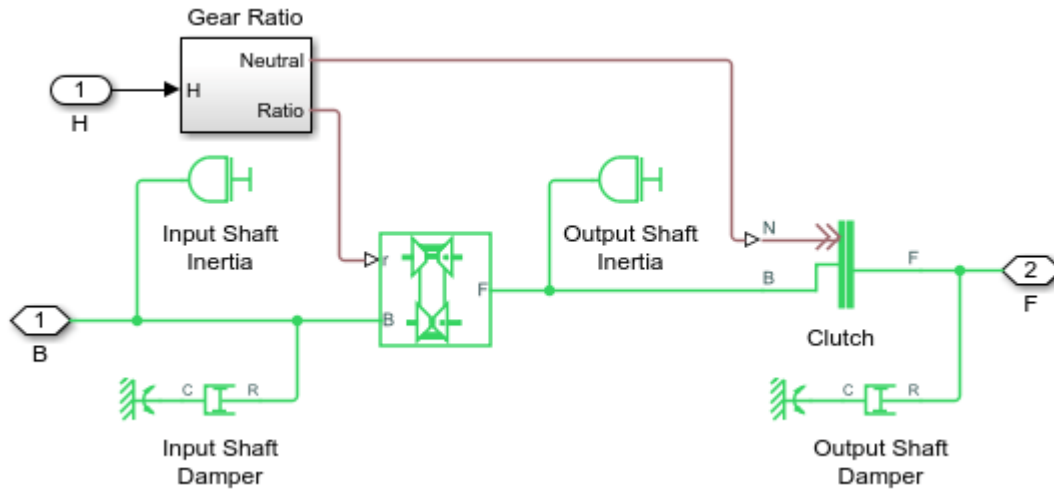
Clutch Pedal Linkage Subsystem



The clutch normal force F is determined by the clutch spring compression. Depressing the clutch pedal reduces clutch spring compression and hence the clutch force. When the clutch plate displacement is less than -3mm , the clutch force is zero, and it is fully disengaged.

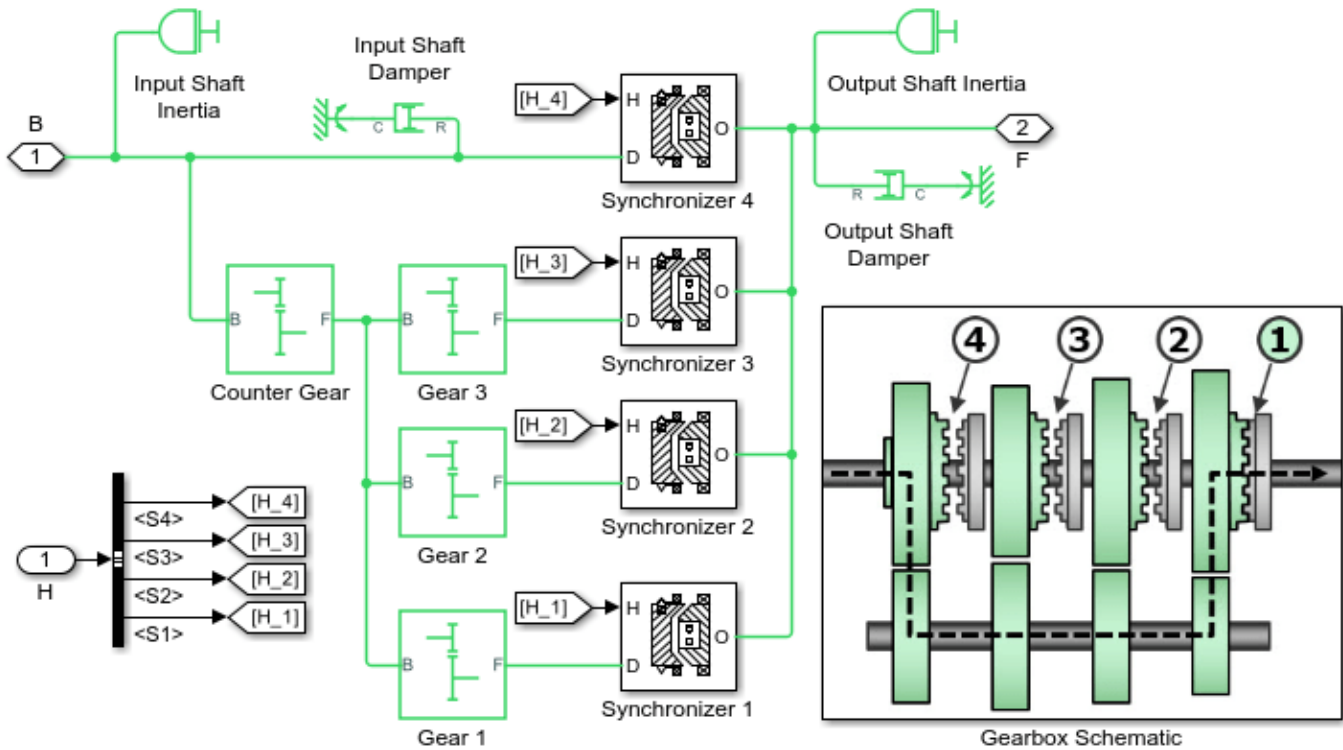
Abstract Transmission Variant

This is the simplified version of the transmission. Using a variable ratio gear that accepts the gear ratio as an input signal, we remove nearly all of the discontinuities. When properly tuned, it can match the results of the detailed variant.



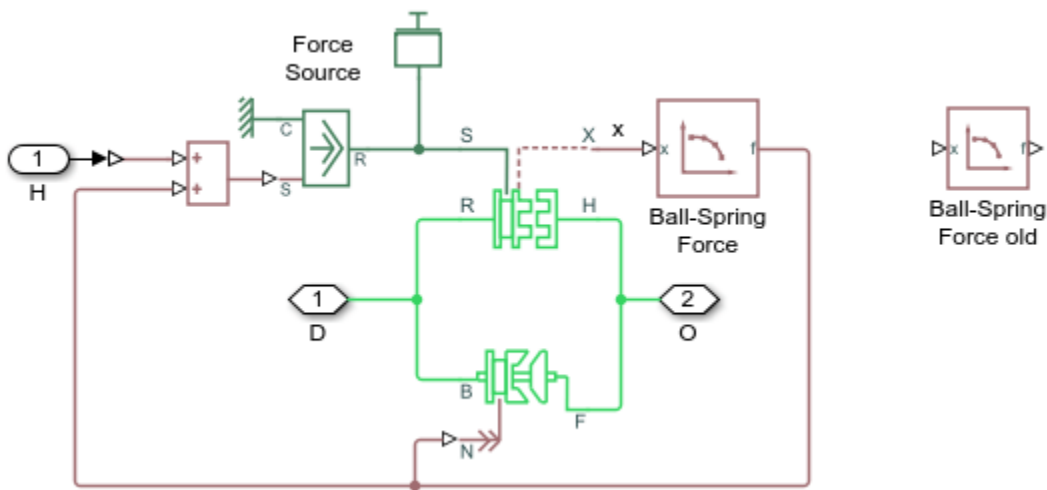
Detailed Transmission Variant

This is the detailed version of the transmission. It contains all the gears plus the dog clutches and cone clutches. This variant provides very accurate results and is perfect for tuning control algorithms and estimating fuel economy.



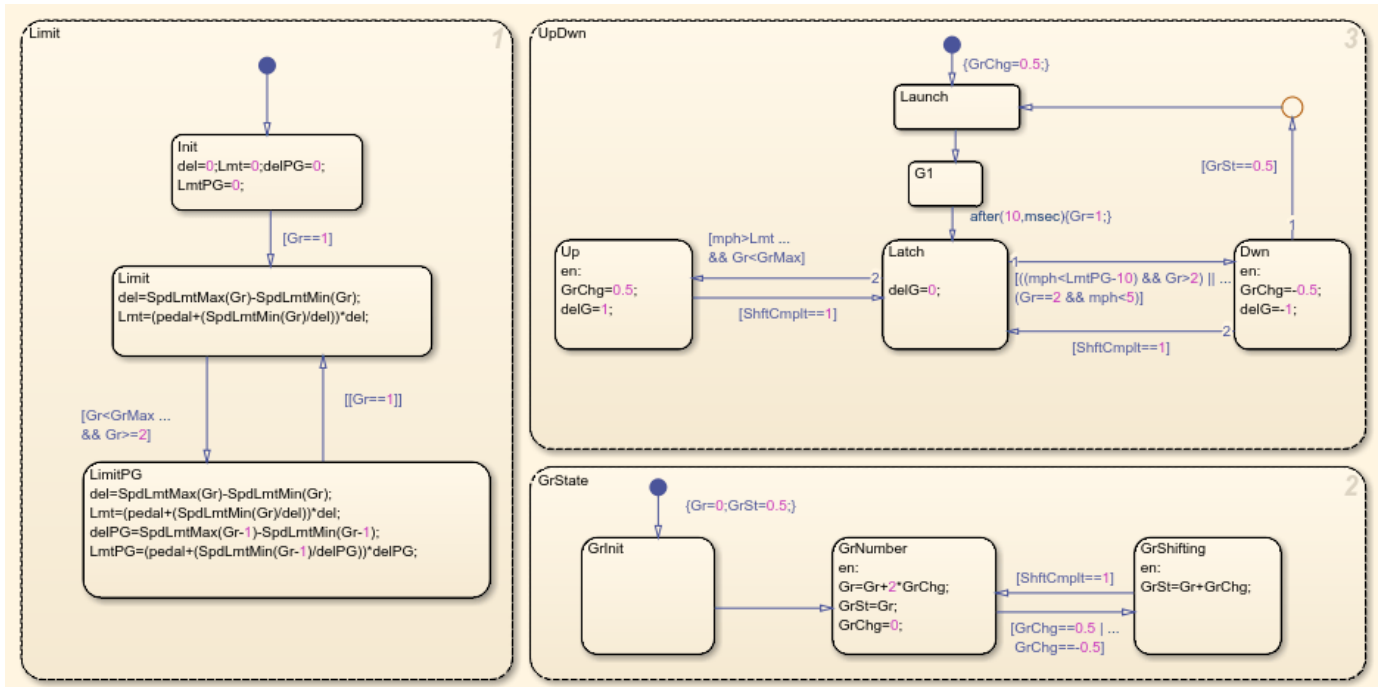
Synchronizer 1 Subsystem

This subsystem implements a customized model of a synchronizer. Here, the structure of the synchronizer can be seen and modified. Simscape™ Driveline™ also provides pre-built single and double-synchronizer blocks.



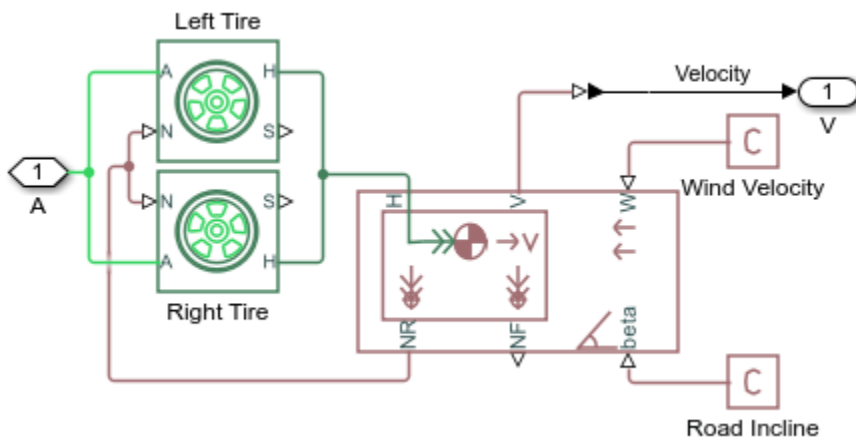
Gear Control Subsystem

This subsystem implements a logic for auto transition of gear. The logic is implemented in Stateflow™, it has three parallel flows for gear up/down shift, velocity limit calculation for gear shifting and gear state flow to give the current gear.



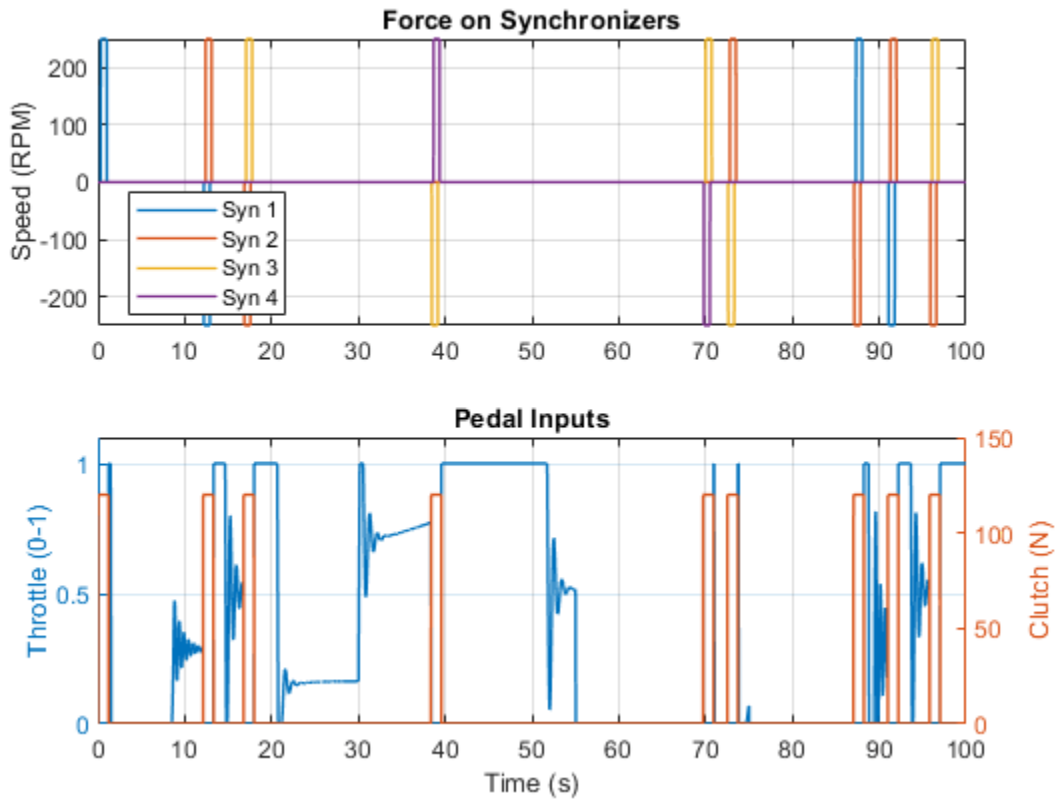
Vehicle Body Subsystem

This subsystem implements the longitudinal vehicle dynamics and the tire model. It is sufficient for running drive cycles to estimate fuel economy and to tune transmission control algorithms.

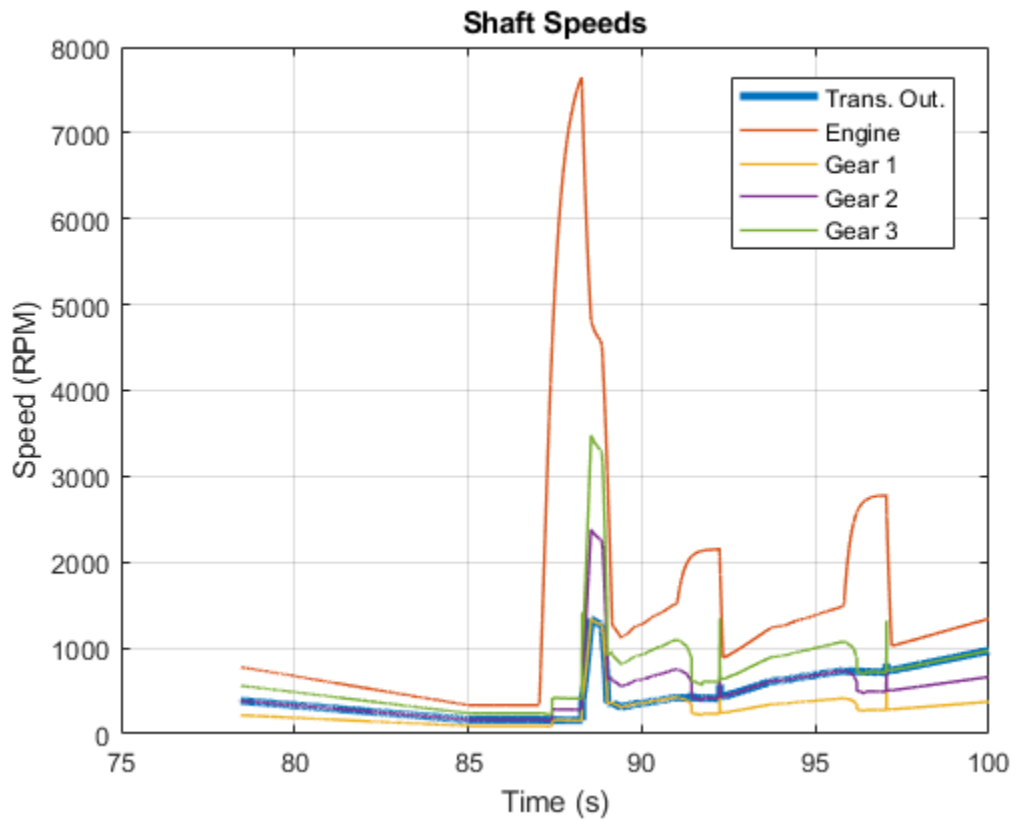


Simulation Results from Simscape Logging

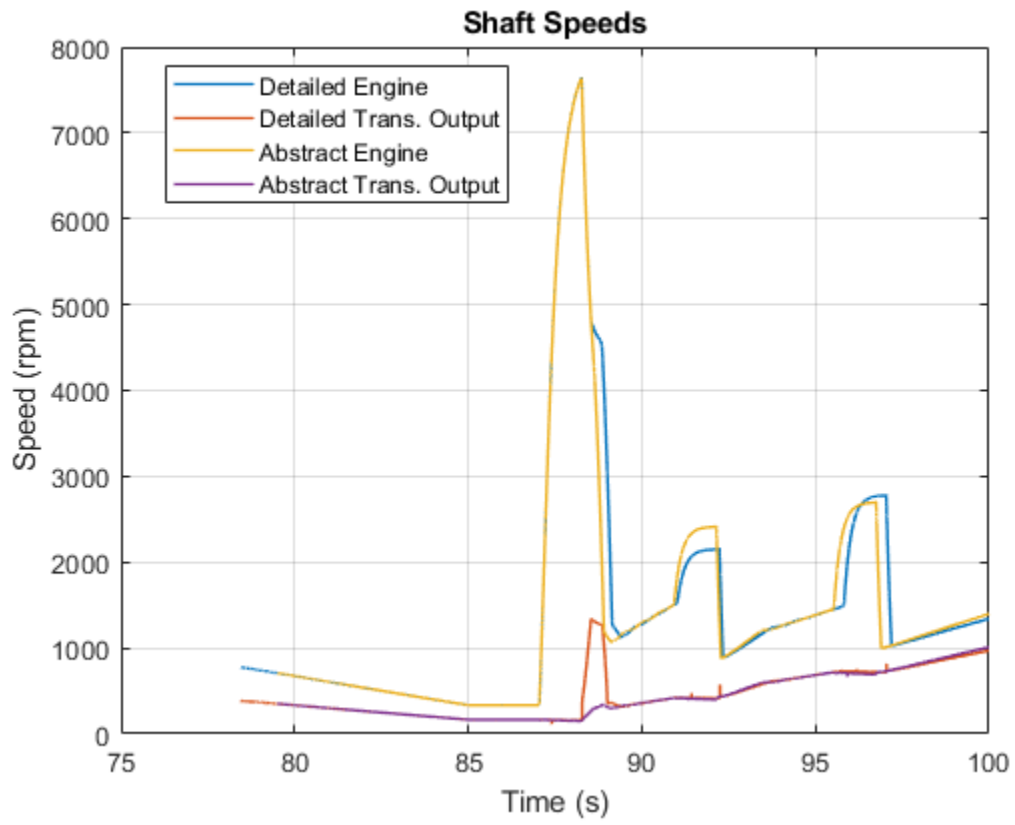
The plots below show the driver inputs to the engine and transmission. The top plot shows the forces applied to the synchronizers to select the active gear, and the bottom plot shows the inputs to the throttle and clutch.



The plot below show the speeds of various shafts in the drivetrain. The output shaft speed is equal to the speed of the gear whose synchronizer is engaged. The engine stays in appropriate speed range even though the vehicle is accelerating because of the chosen gear.



The plot below show the behavior of the abstract models and the detailed model. The results are very similar, indicating that the abstract models can be used without a significant loss of accuracy.



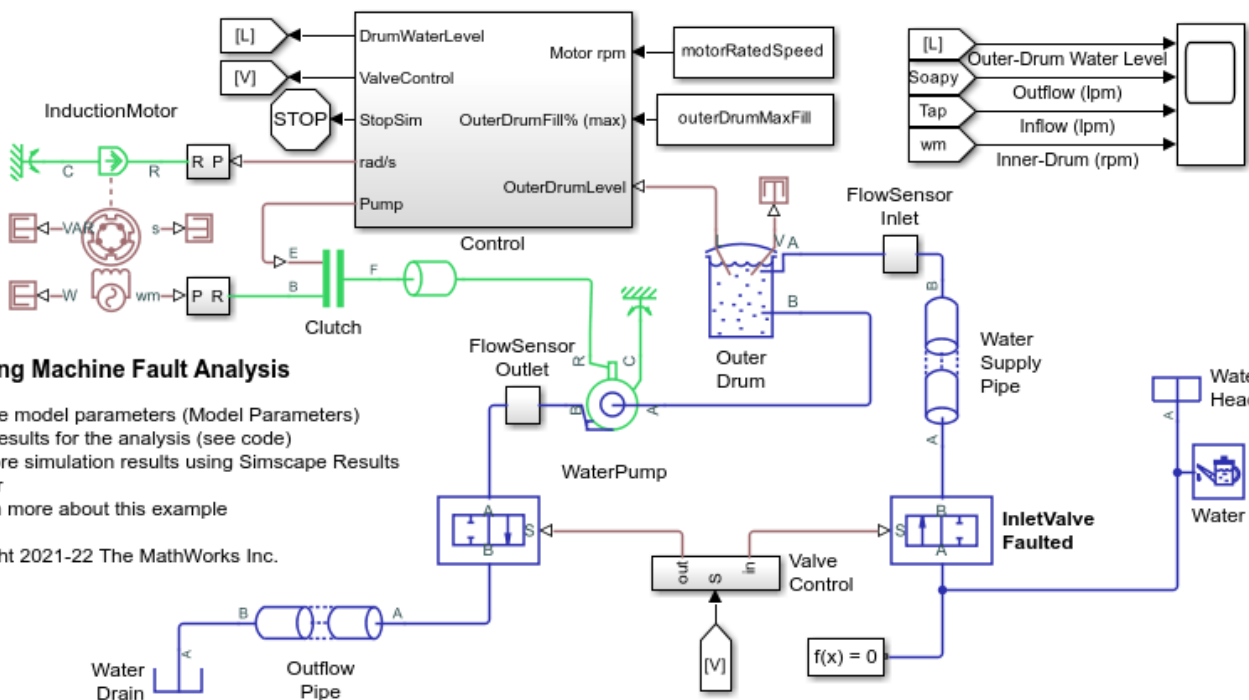
Washing Machine Fault Analysis

This example shows how to model a washing machine and introduce a fault in its operation. The machine water supply faults at a specific time and this example models the system response under this scenario. The electric motor switches off and the machine operation is aborted by discharging the partially filled outer drum.

Model Overview

```
open_system('sscv_washing_machine_fault_analysis.slx')
```

```
set_param(find_system('sscv_washing_machine_fault_analysis','FindAll','on','type','annotation',
```



Washing Machine Fault Analysis

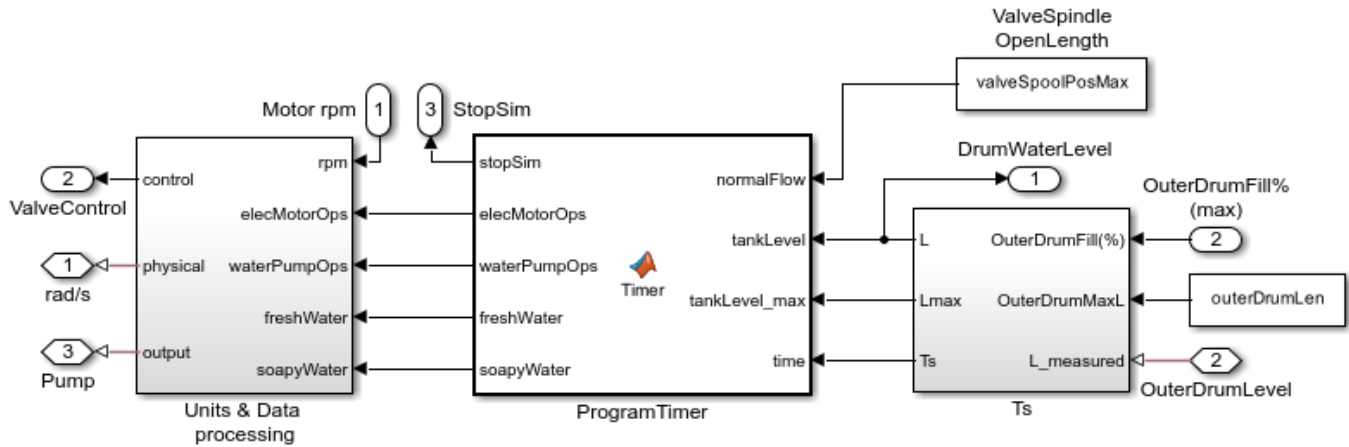
1. Define model parameters (Model Parameters)
2. Plot results for the analysis (see code)
3. Explore simulation results using Simscape Results Explorer
4. Learn more about this example

Copyright 2021-22 The MathWorks Inc.

Control Overview

To modify or edit the machine wash cycle, use the ProgramTimer subsystem. The ProgramTime subsystem comprises the response to a fault.

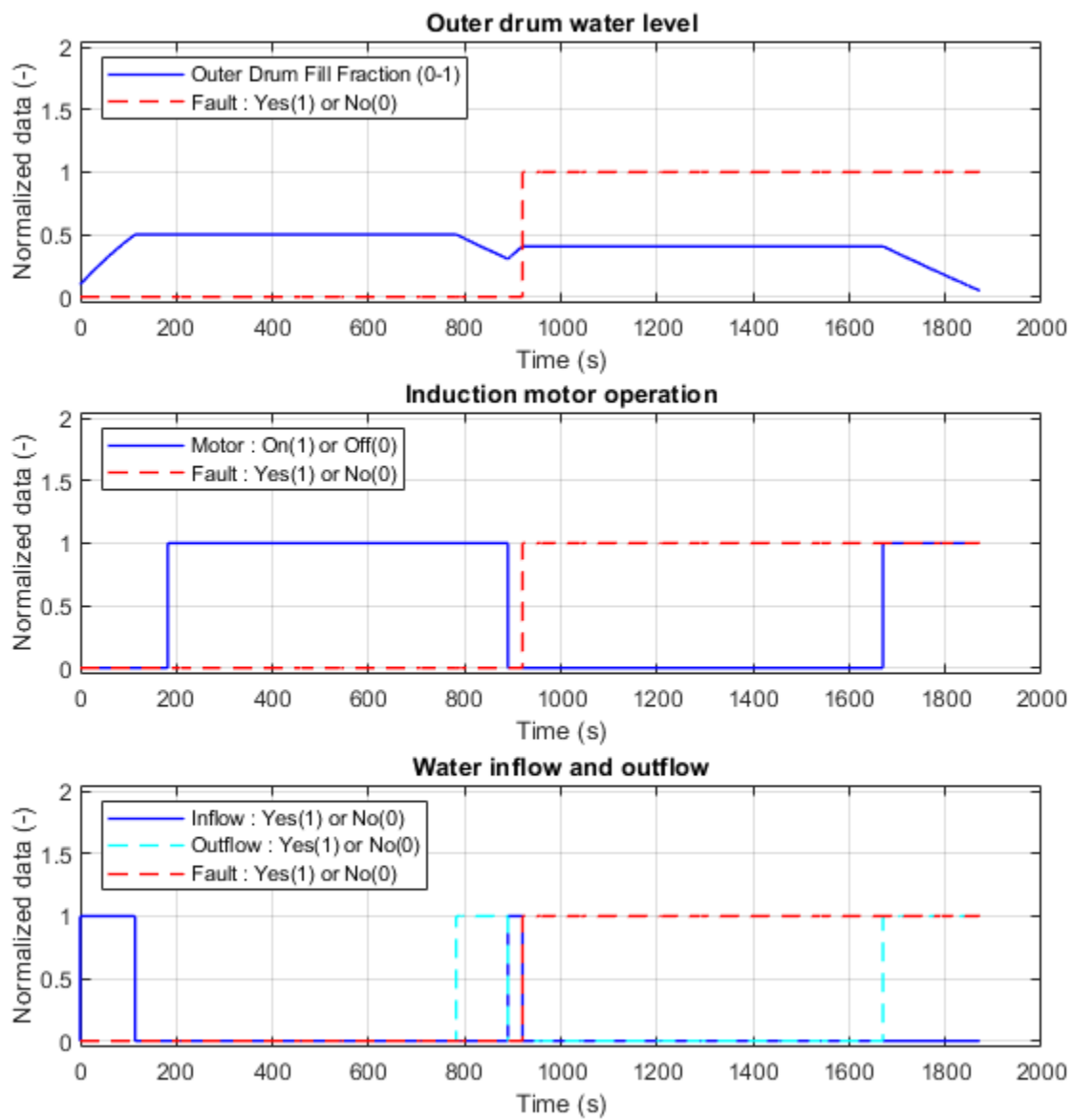
```
open_system('sscv_washing_machine_fault_analysis/Control','force')
```



Simulation Results from Simscape™ Logging

The plot below shows the water inlet and outlet valve operation (On/Off) and the pump operation with outer drum fill level. When the inlet valve faults, the pump switches off. After some time, the outlet valve opens to discharge the partially filled washing machine drum.

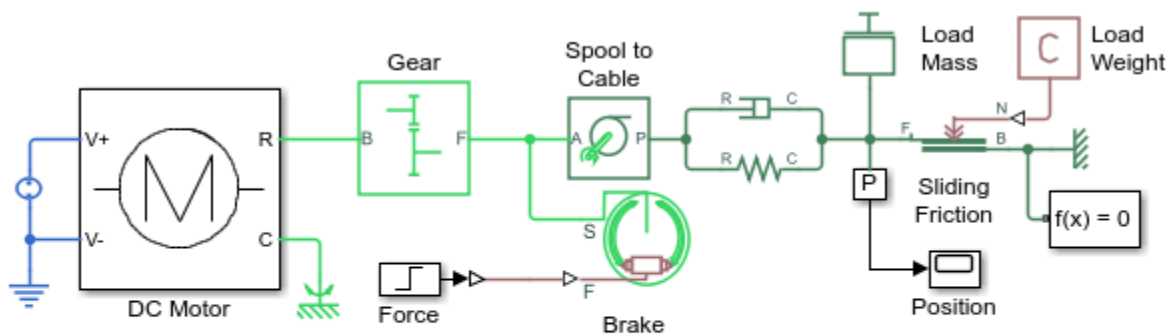
```
ssc_v_washing_machine_fault_analysis_plotResults;
```



Winch with Brake

This example shows a winch driven by a DC motor and controlled with a double-shoe brake. The motor is connected to a 24 volt supply, and the shaft speed is stepped down with a 10:1 gear reduction. The pulled load is modeled as a mass with sliding friction. At a simulation time of six seconds, force is applied to an internal double-shoe brake, stopping the winch.

Model

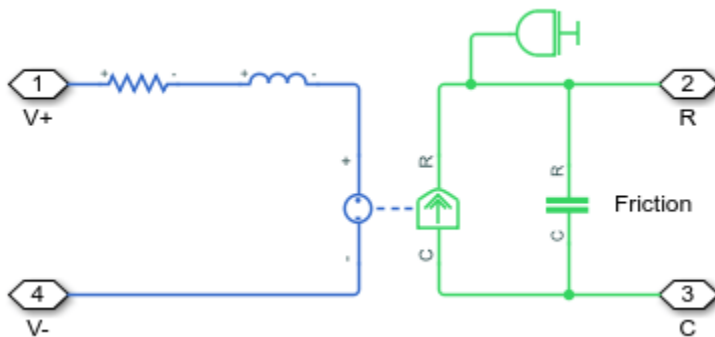


Winch with Brake

1. Plot position of load and motor power (see code)
2. Explore simulation results using Simscape Results Explorer
3. Learn more about this example

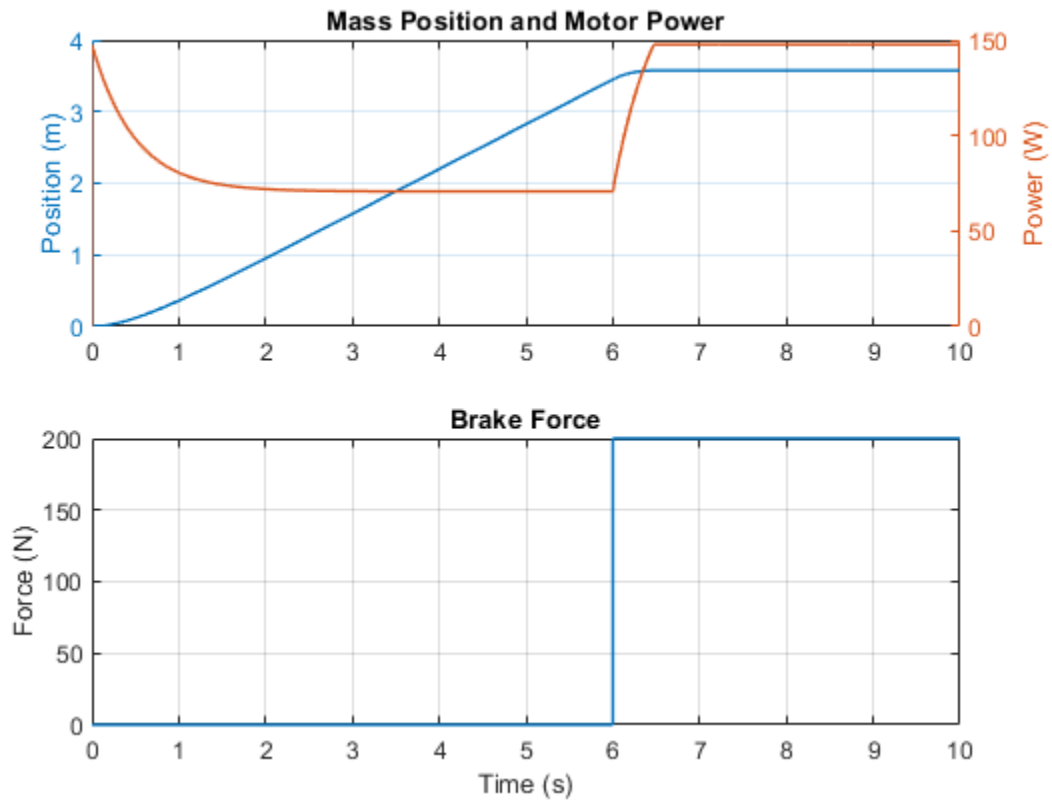
Copyright 2012-2022 The MathWorks, Inc.

DC Motor Subsystem



Simulation Results from Simscape Logging

The plot below shows the position of a mass dragged across a floor by a winch. When the brake is applied, the mass stops moving. This stalls the motor, increasing the amount of power it draws from its voltage source.

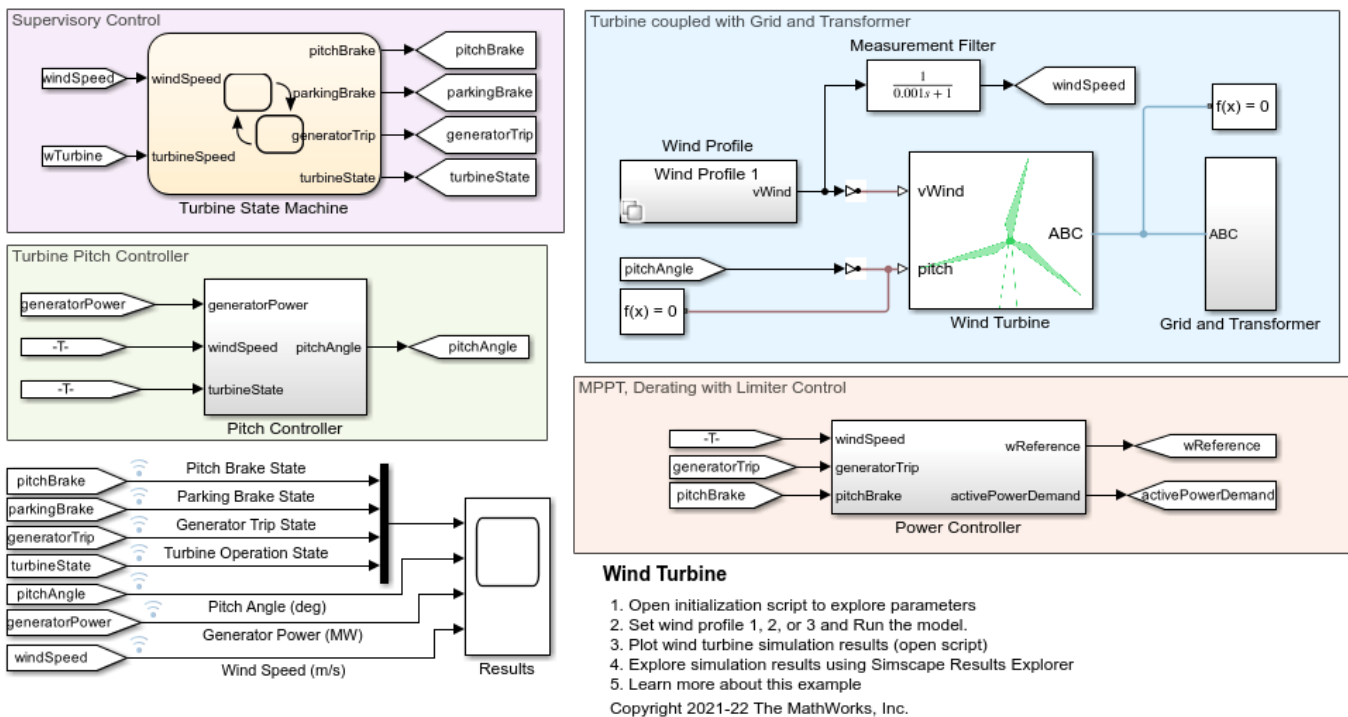


Wind Turbine

This example shows how to model, parameterize, and test a wind turbine with a supervisory, pitch angle, MPPT (maximum power point tracking), and derating control. When you run the plot function, it generates a plot of the state transitions, normalized physical quantities such as the wind speed, wind turbine rotation speed, generator power, and pitch angle.

Model

The following figure shows the model of a wind turbine. The mechanical and electrical domains each require their own Solver Configuration block.



Supervisory Control Subsystem

This subsystem demonstrates how to model the wind turbine state machine. The turbine state machine defines four wind turbine states.

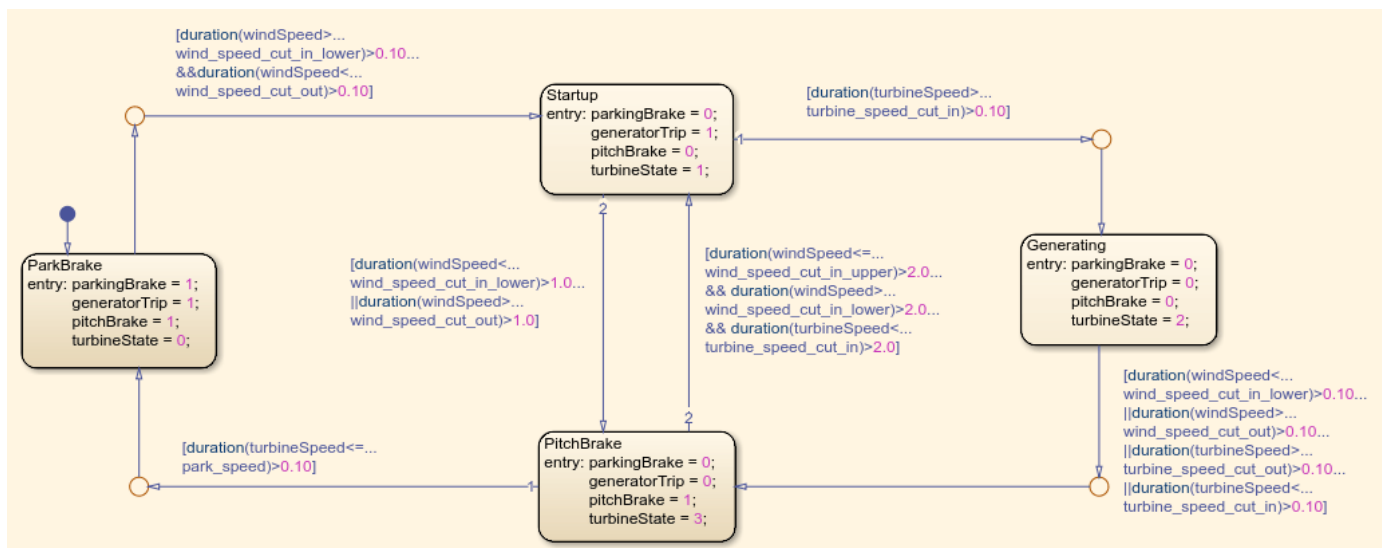
Park brake mode: This is the entry mode of the wind turbine operation. The wind turbine enters the park brake mode from the pitch brake mode when the turbine rotor speed is under the permissible limits for safe operation. During this mode, the generator is in the tripped state, the hydraulic park brake is engaged, and the wind turbine rotor blades are pitched to the braking angle for the aerodynamic braking. The hydraulic brake is the secondary method for braking the wind turbine.

Startup mode: The wind turbine enters the startup mode from the park brake mode when the wind speed is under the permissible limits for safe operation. The wind turbine enters this mode from the pitch brake mode when the wind speed and the turbine speed are under the permissible limits. During this mode, the generator is in the tripped state, the hydraulic park brake is released, and the

wind turbine rotor blades are pitched to the minimum angle for achieving the maximum turbine rotor acceleration.

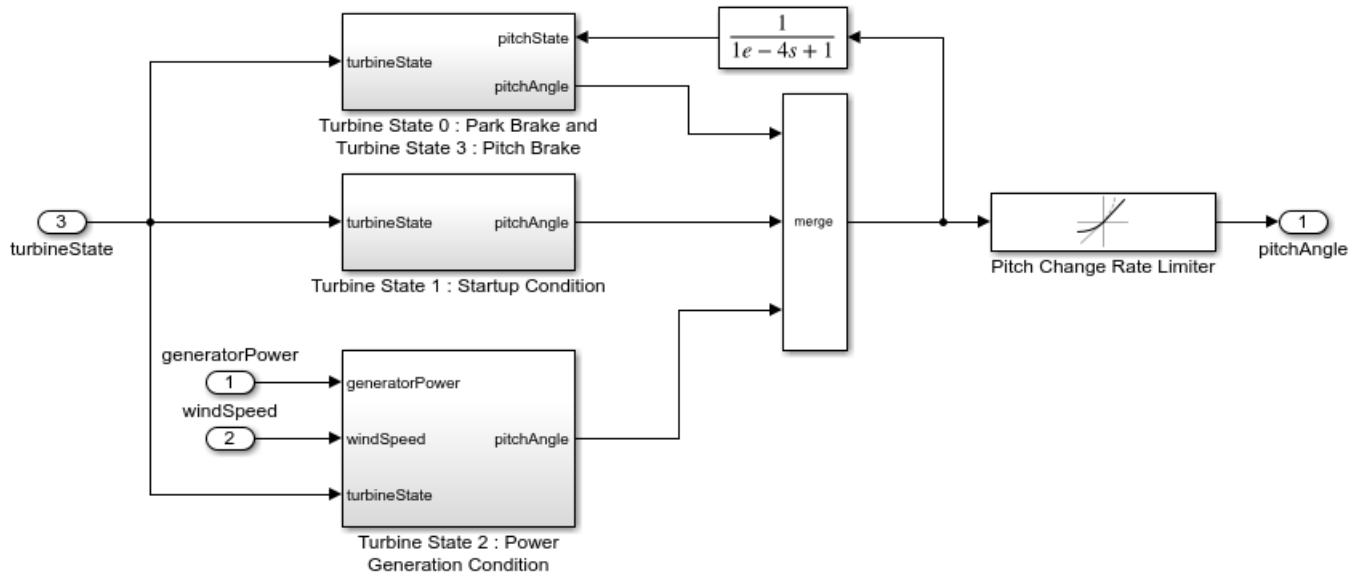
Generating mode: The wind turbine enters the generating mode from the startup mode when the wind turbine rotor speed goes above the turbine cut in speed. During this mode, the generator is connected to the transformer, the hydraulic park brake is released, and the wind turbine rotor blades are pitched to achieve the optimal electric power generation as per the operating conditions.

Pitch brake mode: The wind turbine enters pitch brake mode from generating mode when the wind speed and turbine rotation speed are not under the permissible limits. The wind turbine enters the pitch brake mode from the startup mode when the wind speed is not under the permissible limits. During this mode, the generator is connected to the transformer to consume the kinetic energy available in the rotor blade, the hydraulic park brake is released, and the wind turbine rotor blades are pitched to the braking angle for the aerodynamic braking. The aerodynamic braking is the primary method for braking the wind turbine.

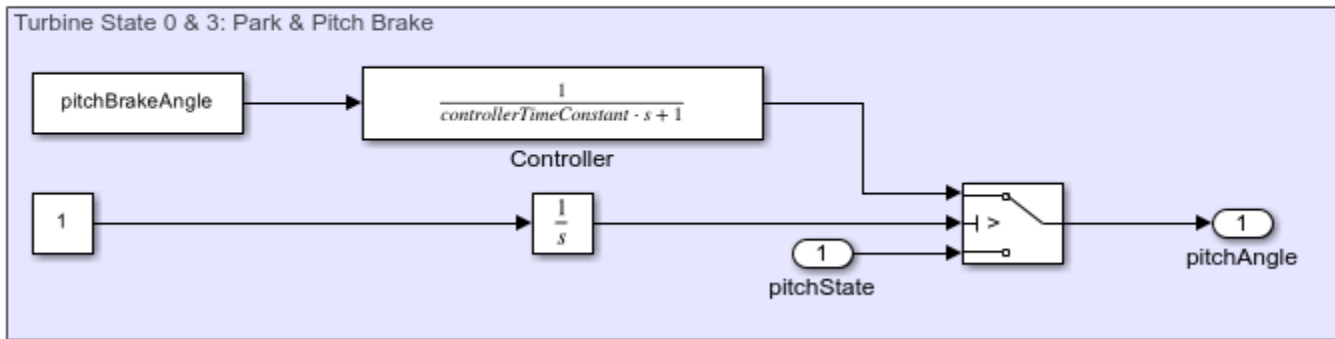


Pitch controller Subsystem

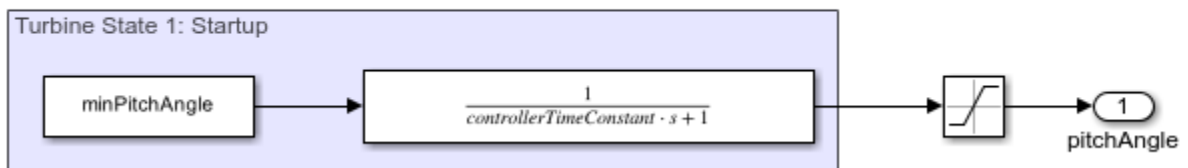
This subsystem demonstrates how to model the pitch angle controller of the wind turbine.



Park brake and pitch brake mode: The pitch angle is set to 95 degree for the aerodynamic braking during these modes.

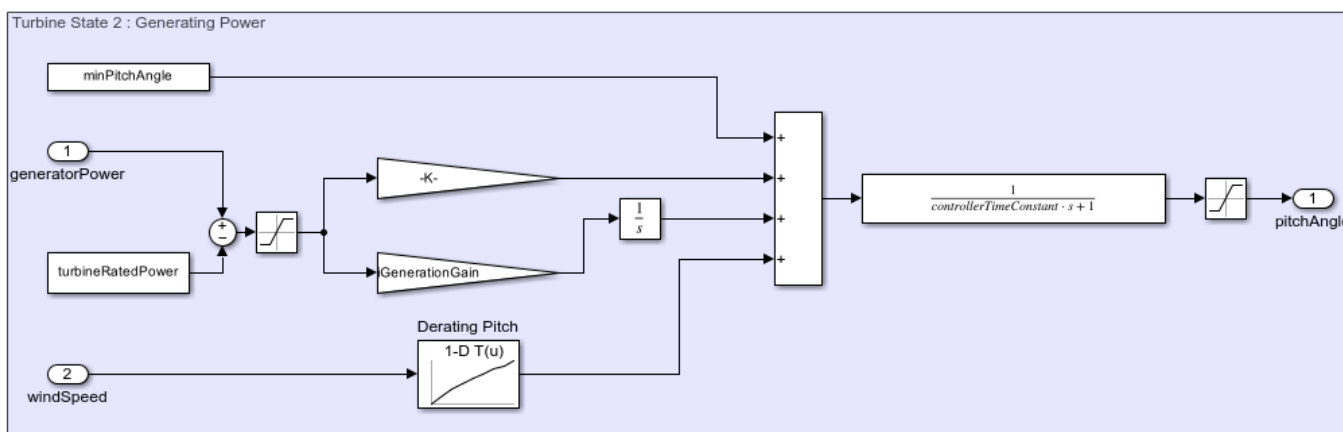


Startup mode: The pitch angle is set to 1 degree for achieving the maximum acceleration during this mode.



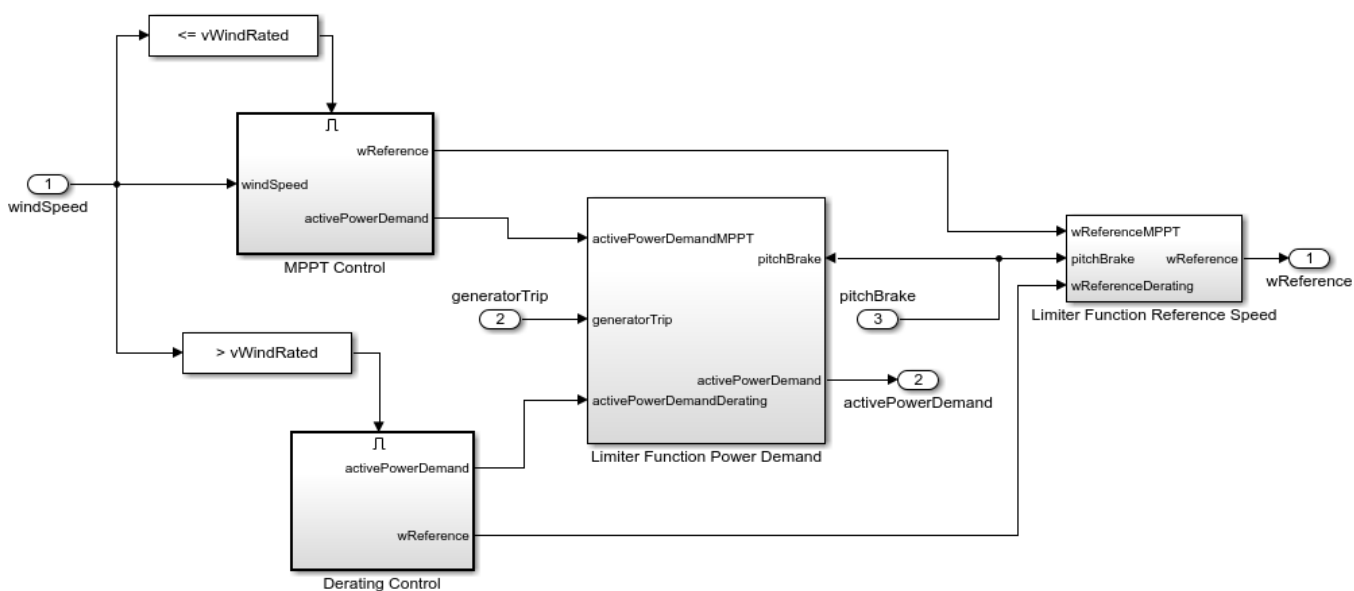
Generating mode: The pitch angle is set to achieve the optimal electric power generation as per the operating conditions.

1. The pitch angle remains at the lowest setting at 1 degree up to rated wind speed. This control acts in synchronization with the wind turbine's MPPT power control.
2. The pitch angle changes when the wind speed is above the rated wind speed in synchronization with the wind turbine's derating power control.

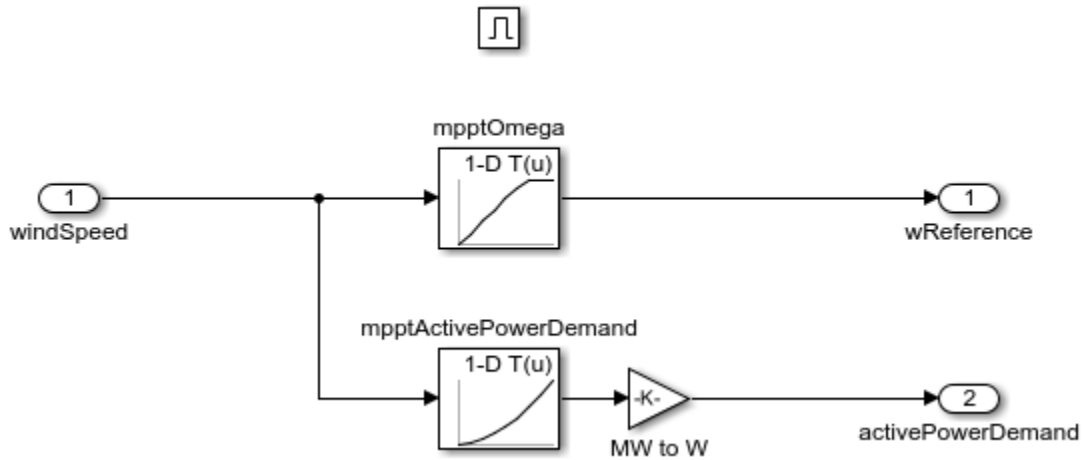


Power Controller Subsystem

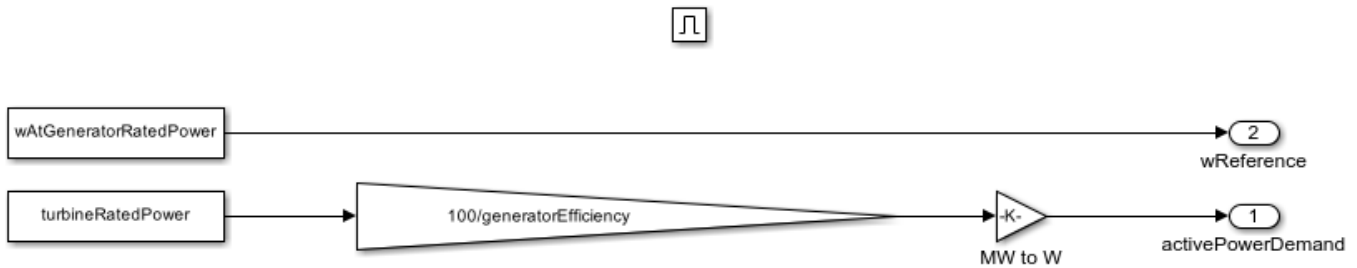
This subsystem demonstrates how to model the power demand and the generator input speed reference for the optimal torque loading on the wind turbine through the generator.



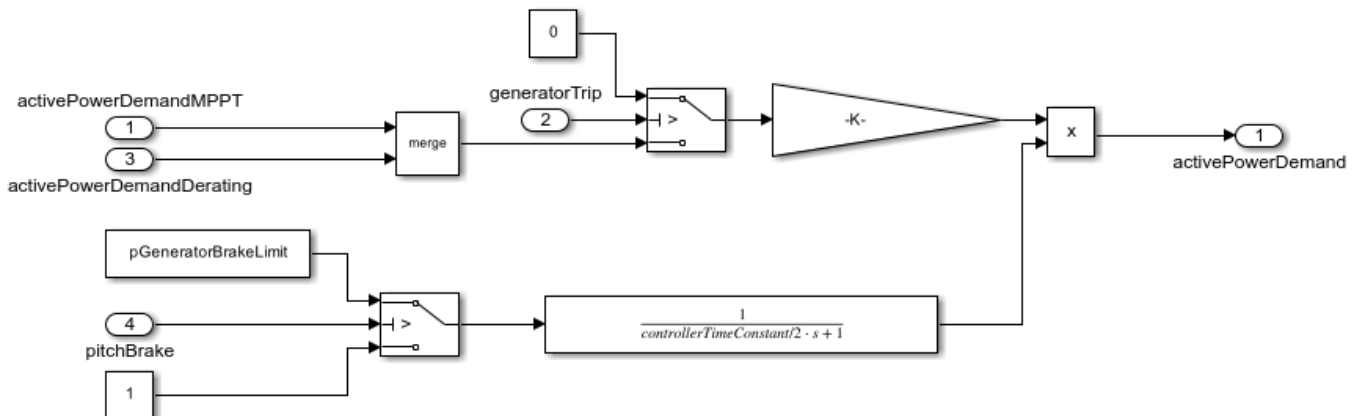
MPPT control: This mode is active up to the rated wind speed. The power demand and the generator reference speed is as per the wind turbine characteristic power performance curve.



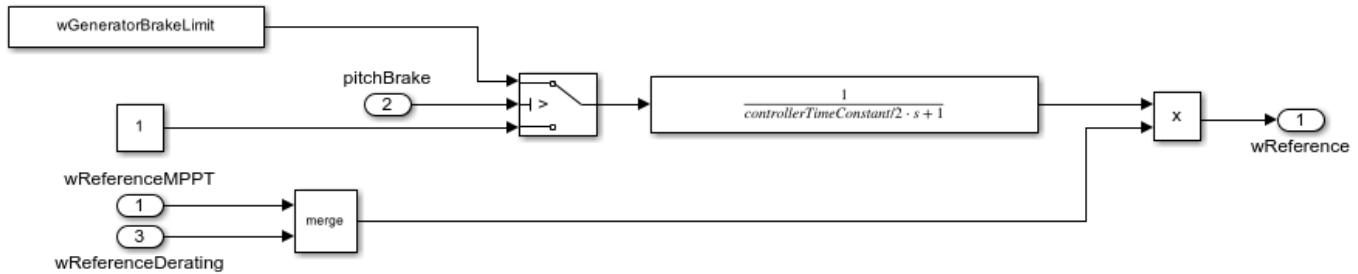
Derating control: This mode is active above the rated wind speed. The power demand and the generator reference speed is kept constant based on the rating of the wind turbine.



Limiter Function Power Demand: This function limits the power demand as per the wind turbine's mode of operation.

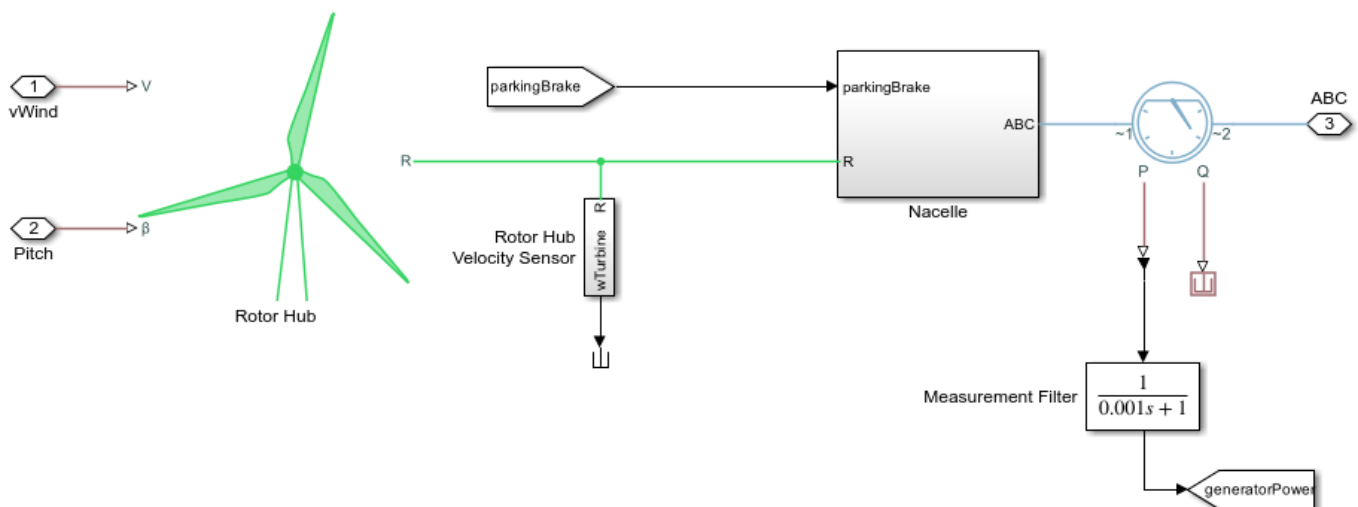


Limiter Function Reference Speed: This function limits the speed reference as per the wind turbine's mode of operation.



Wind Turbine Subsystem

This subsystem demonstrates how to model the wind turbine.



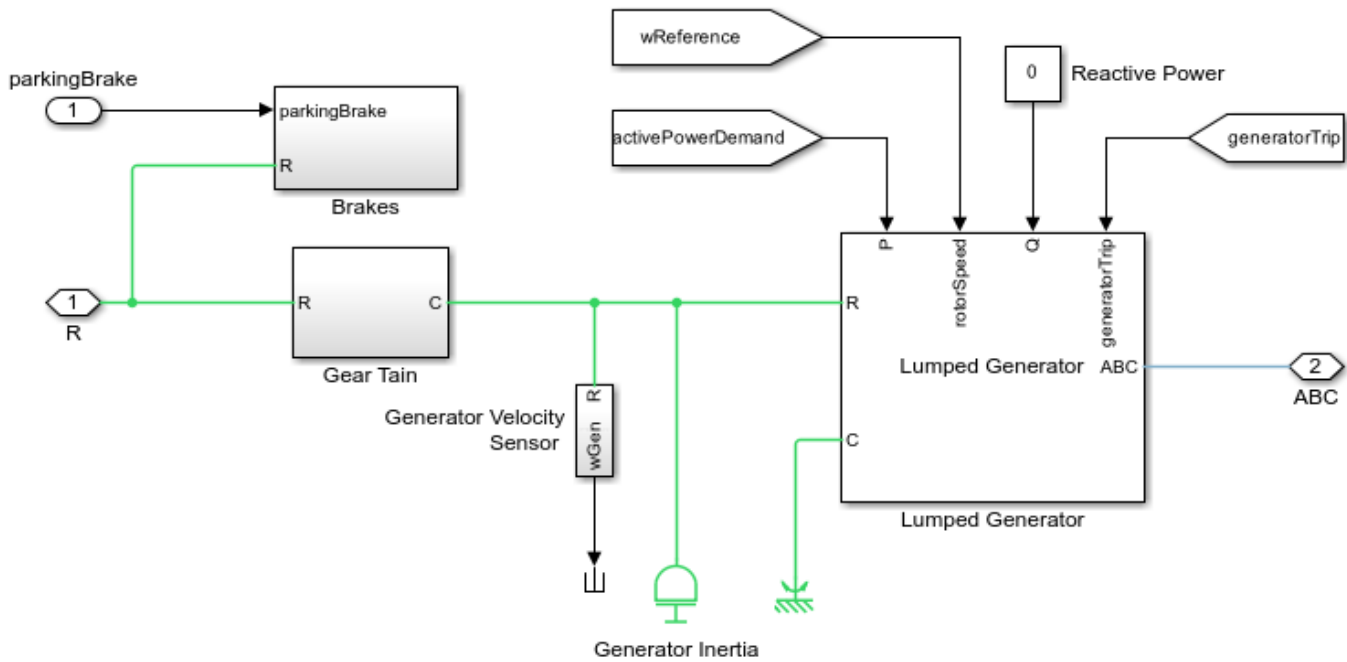
Rotor Hub: The rotor of a horizontal axis wind turbine is modeled using the Simscape™ Driveline™ Wind Turbine block. In this block, *mechanical power* extracted from the wind, P , is calculated as

$$P = 1/2 * rho * Ar * vWind^3 * cp(pitch, lambda)$$

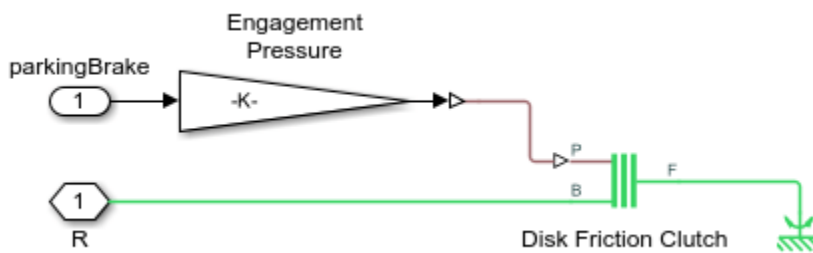
where,

- 1 rho is the air density
- 2 Ar is the area swept by the rotor blades
- 3 $vWind$ is the wind speed
- 4 cp is the power coefficient as a function of $lambda$ and $pitch$.
- 5 $lambda$ is the ratio of the rotor blade tip speed and the wind speed, it is also called Tip Speed Ratio (TSR)
- 6 $pitch$ is the rotor blade pitch angle

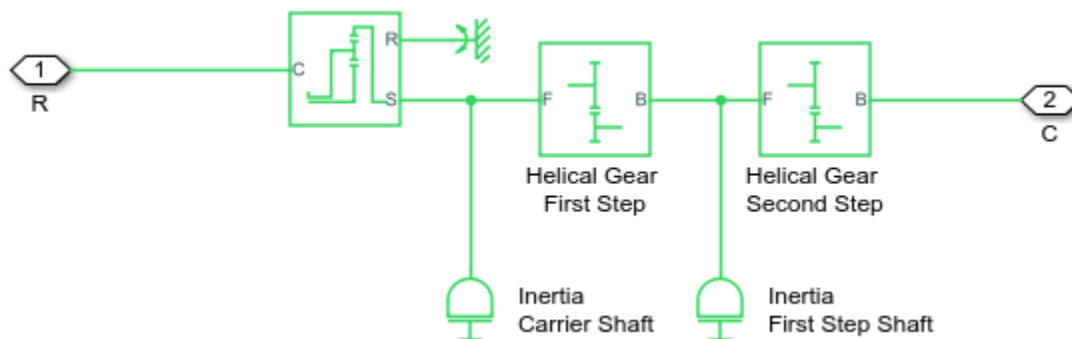
Nacelle: This subsystem demonstrates how to model the nacelle components of a wind turbine.



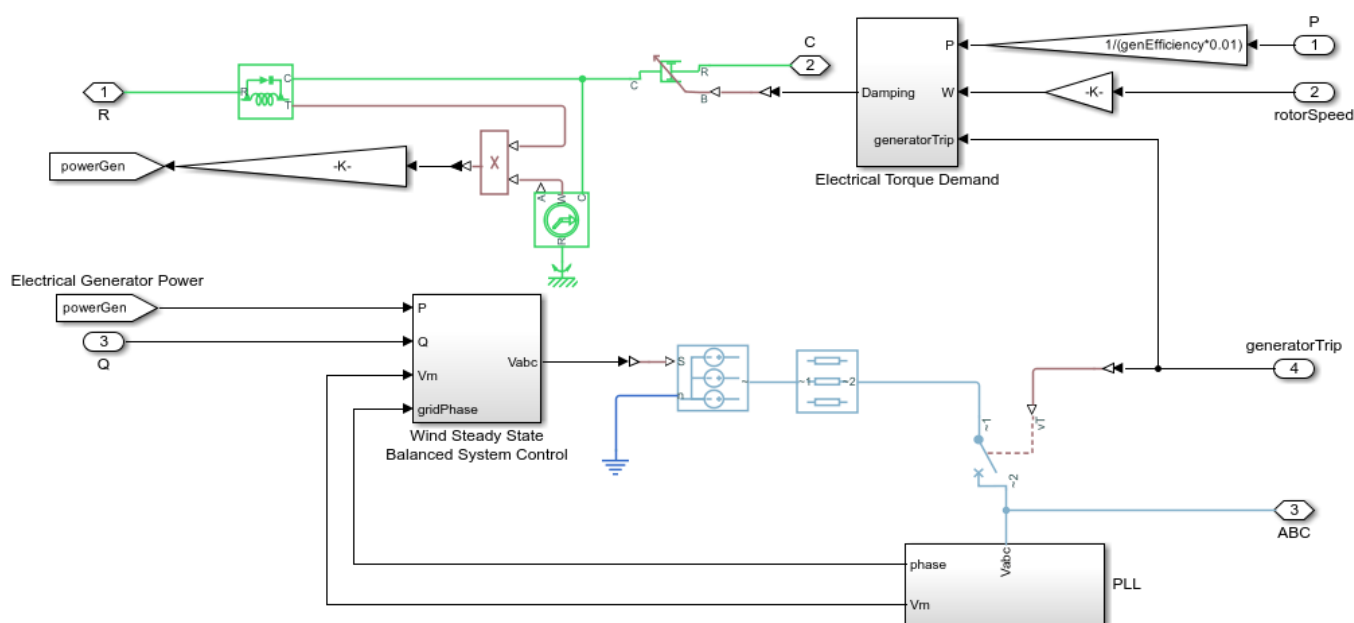
Brakes: This subsystem demonstrates how to model the brakes in the nacelle. The hydraulics brake is a secondary braking method in the wind turbine. The brakes are engaged when the wind turbine speed goes below the parking brake speed while the wind turbine is under the park brake mode or during the wind turbine maintenance operation. If they are applied above the parking brake speed, the brake can burn out or the nacelle can catch fire due to the excessive frictional heat generation.



Gear train: This subsystem demonstrates how to model the gear train in the nacelle. An epicyclic gear train represents the gear train system with the power losses. The gear train contributes to high power loss in the wind turbine operation.



Lumped generator: This subsystem demonstrates how to model the generator in the nacelle. This is a simplified model of a generator to increase the simulation speed.



The wind turbine has a larger time constant and slower response than a traditional doubly-fed induction generator (DFIG) system. To simulate wind turbine control, you must run the simulation longer.

The closed-loop DFIG system is faster than wind turbine control systems such as pitch control. Therefore, a low fidelity lumped DFIG generator system is practical for improving simulation speed and providing flexibility. The lumped generator system integrates with the wind turbine system to simulate different aspects of the control algorithm.

The lumped generator model tracks the grid voltage and phase angle at the point of common coupling (PCC) using the phase locked loops (PLL). To improve the simulation speed, fast acting PLL is modelled using the grid voltage zero crossing detection. Lumped generator uses the grid voltage and phase obtained from the PLL and implements the power flow equation with the generator lumped leakage inductance and resistance. The lumped generator model takes the real power (P), reactive power (Q), and reference generator shaft speed as input.

In the wind turbine system, the lumped generator model gets the power reference and approximate speed reference input from the wind turbine power control system. Based on the reference input, the generator applies the load torque to the wind turbine shaft and supply the electrical power to the grid.



$$3V_g I_g^* = P + jQ$$

$$I_{gm}^* = \frac{2(P + jQ)}{3V_{gm}}$$

$$V_g = V_{gm} \angle 0^\circ$$

$$V_{ig} = V_{gm} + jX_L I_{gm} + R I_{gm} = V_{igm} \angle \phi$$

$$V_{iga}(t) = V_{igm} \sin(\omega t + \phi)$$

$$V_{igb}(t) = V_{igm} \sin(\omega t + \phi - 120^\circ)$$

$$V_{igc}(t) = V_{igm} \sin(\omega t + \phi - 240^\circ)$$

where,

- 1 V_g is grid rms phase voltage at PCC
- 2 I_g is grid rms phase current at PCC
- 3 V_{gm} is grid peak phase voltage at PCC
- 4 I_{gm} is grid peak phase current at PCC
- 5 V_{igm} is electrical generator peak induced phase voltage
- 6 V_{ig} is electrical generator induced phase voltage
- 7 X_L is lumped leakage reactance of induction generator (wrt to stator)

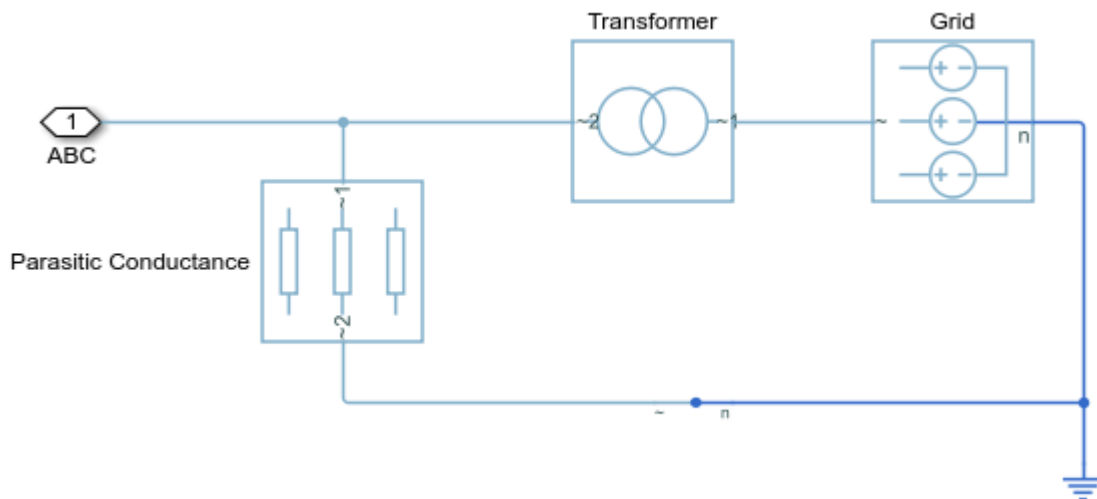
- 8 R is lumped resistance of induction generator (wrt to stator)
- 9 ωt is instantaneous grid voltage angle

Here, V_{gm} and ωt are obtained from the PLL.

The lumped generator system model block should be used only in fixed frequency, full sinusoidal, and balanced three phase grid connected systems. This model is not suitable for the transient electrical stimulation such as the fault ride through and the grid frequency variation simulation.

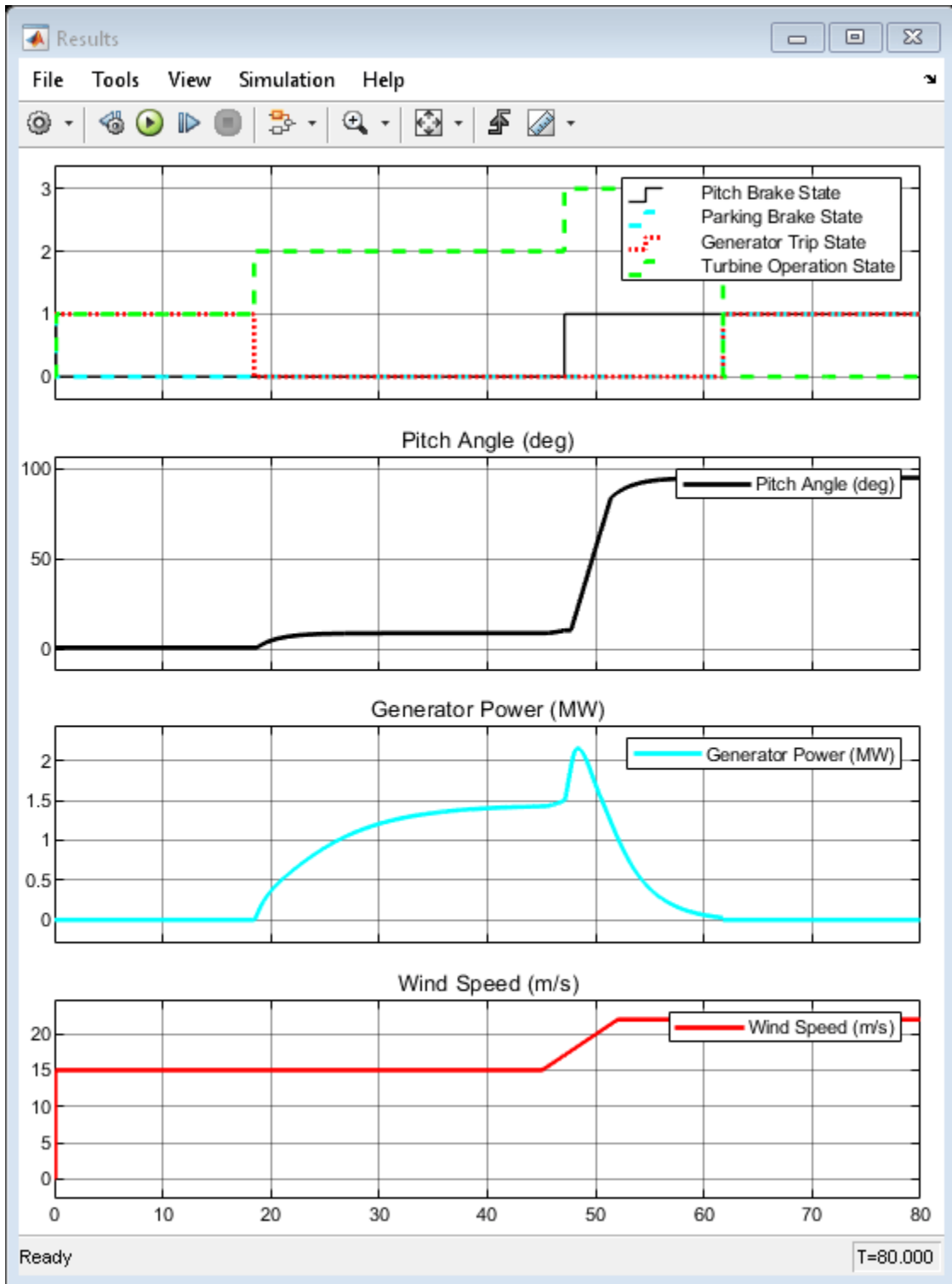
Grid and Transformer Subsystem

This subsystem demonstrates how to model the grid and transformer. The parasitic conductance is added to increase the simulation speed.



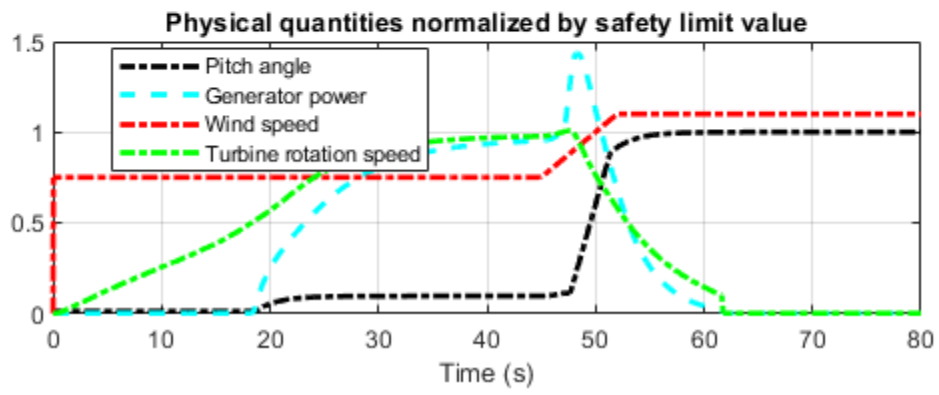
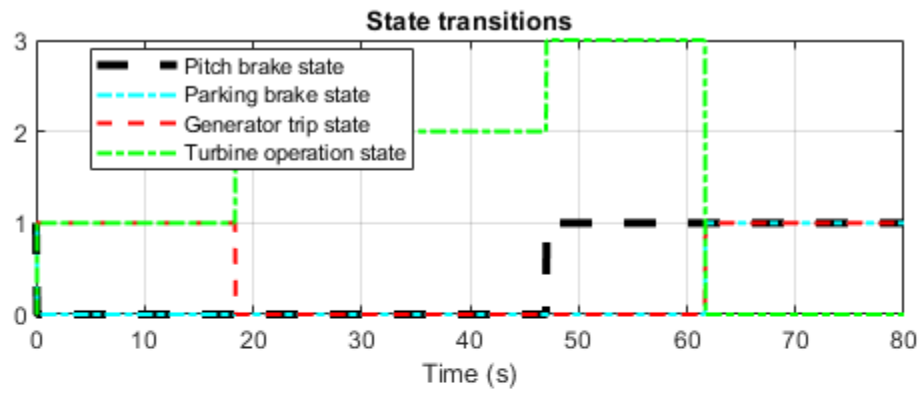
Simulation Results from Scope

This figure shows the supervisory control states, turbine blade pitch angle, generator power, and wind speed.



Simulation Results from Simscape Logging

This model generates plot of the state transitions, the normalized physical quantities such as the wind speed, wind turbine rotation speed, generator power, and pitch angle.



Wind Turbine Driveline with Vibrations

This example shows a wind turbine rotor, controller, and flexible drive shaft with transverse vibrations. Click the button below to open the model.

Open Model

```
open_system('WindTurbineDrivelineWithVibrations')
```

Example Overview

This example contains a wind turbine rotor, controller, and flexible shaft with transverse vibrations. The direct-drive wind turbine has a power capacity of 10 MW. Controllers based on the NREL 5-MW reference wind turbine adjust the generator torque and collective blade pitch. The rotor and shaft are supported by 4 fluid film bearings. This example lets you optionally model transverse vibrations of the driveshaft due to a generator unbalanced magnetic force. The rotor dynamics can be modeled using a lumped mass finite element method or a reduced order model to simulate the shaft vibrations more efficiently. The reduced order models include a traditional static eigenmodes method and a speed-dependent eigenmodes method. This example includes scripts for analyzing the performance of the different reduced order models.

Open Model

```
open_system('WindTurbineDrivelineWithVibrations')
```

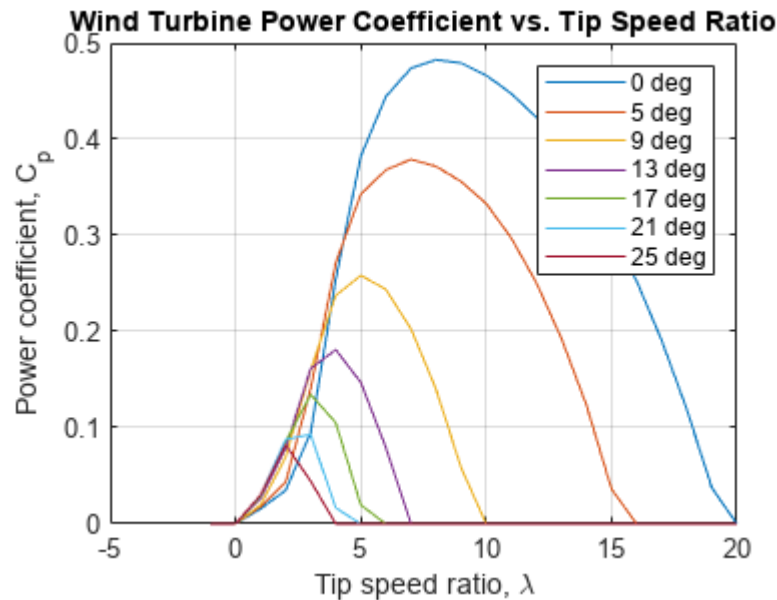
Aerodynamics

Rotor torque as a function of wind speed and blade pitch is modeled using a Wind Turbine block from Simscape Driveline™. The block models torque quasi-statically based on the wind turbine power coefficient curves and rotor speed.

Plot the wind turbine power curve

```
% Retrieve the model data
hws= get_param('WindTurbineDrivelineWithVibrations', 'modelworkspace');
rotor= getVariable(hws, 'rotor');

% Generate the plot
figure;
hold on
for i=1:length(rotor.pitch)
    plot(rotor.TSR, rotor.Cp(i,:), 'DisplayName', [num2str(rotor.pitch(i)) ' deg'])
end
box on
grid on
legend show
title('Wind Turbine Power Coefficient vs. Tip Speed Ratio')
xlabel('Tip speed ratio, \lambda')
ylabel('Power coefficient, C_p');
```



Controllers & Braking

Controllers for the 10 MW wind turbine are configured for the direct-drive rotation speeds and scaled up from the NREL 5-MW reference wind turbine described in *Jonkman, J.; Butterfield, S.; Musial, W.; Scott, G., Definition of a 5-MW reference wind turbine for offshore system development (No. NREL/TP-500-38060). National Renewable Energy Lab. Golden, CO, USA, 2009.*

The generator torque controller starts producing torque when the rotor speed exceeds the cut-in speed. When the generator power is less than the rated power, torque is a function of the generator speed. The torque values are chosen to maximize power. This example models this torque using a Lookup Table. When the power equals the rated power, the generator torque adjusts based on rotor speed to maintain the rated power. This example switches between the below-rated and above-rated torque generation modes using a Switch block. The Switch block switching criteria represents the rated power threshold via a blade pitch sensor: the collective blade pitch controller causes the blade pitch to exceed 1 deg when the rated power has been reached.

This model implements the generator torque using an Ideal Torque Source block. The model accounts for drivetrain losses as a simple rotational damper. Rotational Power Sensor and Ideal Rotational Motion Sensor blocks sense the generator power and speed, respectively. Generated mechanical power exceeds 10 MW before electrical losses are taken into account.

Open generator controller subsystem

```
open_system('WindTurbineDrivelineWithVibrations/Controlled Generator')
```

The collective blade pitch controller begins adjusting the blade pitch once the rated turbine speed is reached. The controller is a PI controller that tries to maintain the rated turbine speed. The proportional and integral coefficients are scheduled functions of the blade pitch. The blade pitch responds instantaneously to the controller command without accounting for inertial dynamics.

Open blade pitch controller subsystem

```
open_system('WindTurbineDrivelineWithVibrations/Pitch Controller')
```

When the wind speed exceeds the turbine's maximum wind speed, a brake is activated.

Open brake subsystem

```
open_system('WindTurbineDrivelineWithVibrations/Brake')
```

Simulation Results from Scopes

The scope shows the generated power, rotational velocity, and blade pitch response to the wind speed. Once the turbine rotational velocity exceeds a start-up threshold, the power generation begins. Once the rated power is reached, the blade pitch begins increasing to maintain the rated power.

View simulation results

```
open_system('WindTurbineDrivelineWithVibrations');
set_param('WindTurbineDrivelineWithVibrations/Scope','open','on');
out= sim('WindTurbineDrivelineWithVibrations'); %#ok<NASGU>

clear out
```

Simulation Results from Simscape Logging

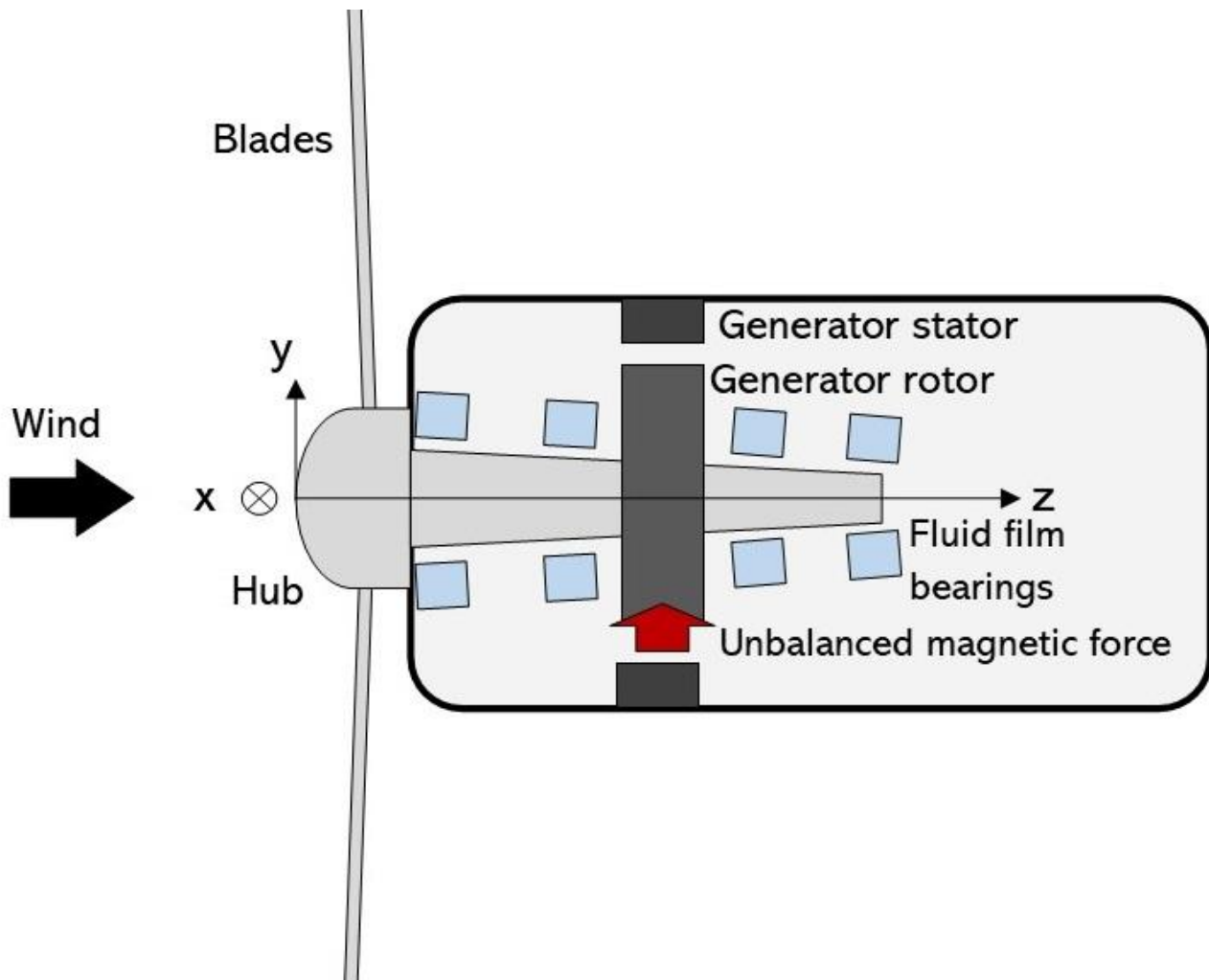
The script below uses results from Simscape logging to plot the wind speed, generated power, turbine rotational velocity, blade pitch, and generator torque in a figure. When the wind speed increases above the rated wind speed, the blade pitch increases to limit the aerodynamic power. When the turbine rotational velocity decreases in response to the increased blade pitch, the generator torque increases to maintain the rated power. When the wind speed decreases, the pitch controller decreases the blade pitch, and the turbine rotational velocity decreases at a faster rate. The generator torque decreases when the generated power is below the rated power.

Plot the controlled wind turbine response

```
WindTurbineDrivelineWithVibrationsPlotControlledPower;
% Uncomment the line below to see the code for generating the figure.
% edit WindTurbineDrivelineWithVibrationsPlotControlledPower;
```

Rotor Dynamics

This example lets you optionally model transverse vibrations of the driveshaft due to a generator unbalanced magnetic force. The Flexible Shaft block models varied shaft diameters, rigid inertias, and fluid film bearings along shaft. The example models the blades, hub, and generator rotor as rigid inertias.



Wind turbine driveline schematic

The Shaft subsystem separates the rigid rotational dynamics that drive the wind turbine and the flexible shaft vibration dynamics. The Ideal Rotational Motion Sensor, PS Gain, and Ideal Angular Velocity Source blocks excite the transverse dynamics at a harmonic of the shaft speed.

Open shaft subsystem and Shaft dialog box

```
set_param('WindTurbineDrivelineWithVibrations/Shaft', 'vibration_model', 'Lumped mass');
open_system('WindTurbineDrivelineWithVibrations/Shaft/Lumped mass');
open_system('WindTurbineDrivelineWithVibrations/Shaft/Lumped mass/Shaft');
```

Click the button below to simulate the model and run an animation of the shaft transverse vibrations. The button generates a standalone figure for running the animation. The image below is a screenshot of the animation.

Animate transverse vibrations

```
set_param('WindTurbineDrivelineWithVibrations/Shaft', 'vibration_model', 'Speed-dependent eigenm
WindTurbineDrivelineWithVibrationsAnimate;
% Uncomment the line below to see the code for generating the figure.
% edit WindTurbineDrivelineWithVibrationsAnimate;
```

Rotor Dynamics Analysis Workflow

The Flexible Shaft block can model the driveshaft transverse vibrations using a speed-dependent eigenmodes method that adjusts the modal properties as the shaft speed changes.

The script below plots the shaft eigen mode shapes and frequencies for varied shaft speeds. Modes with large displacements at the axial locations excited by external loadings and modes with natural frequencies equal to the shaft speed or external loading frequencies will have larger vibration responses, and you generally want to avoid designs with these characteristics. The button generates a standalone figure for plotting the mode shapes. The image below is a screenshot of the figure.

View eigenmode shapes

```
set_param('WindTurbineDrivelineWithVibrations/Shaft', 'vibration_model', 'Speed-dependent eigenmodes');
clear flex_shaft_mode_props % clear any existing mode properties
WindTurbineDrivelineWithVibrationsPlotEigenmodes;
% Uncomment the line below to see the code for generating the app.
% edit WindTurbineDrivelineWithVibrationsPlotEigenmodes
```

Click the button below to view the Campbell frequency analysis plot for the flexible shaft.

View Campbell frequency analysis plot

```
set_param('WindTurbineDrivelineWithVibrations/Shaft', 'vibration_model', 'Speed-dependent eigenmodes');
WindTurbineDrivelineWithVibrationsPlotCampbellDiagram;
% Uncomment the line below to see the code for generating the figure.
% edit WindTurbineDrivelineWithVibrationsPlotCampbellDiagram
```

Reduced Order Model Workflow

Rotor dynamics for the flexible shaft with speed-dependent bearings and rigid inertias can be modeled using a lumped mass finite element method or a more computationally efficient reduced order model. The reduced order models for simulating the transverse vibrations include a traditional static eigenmodes method and a speed-dependent eigenmodes method.

The lumped mass model for the flexible shaft transverse vibrations divides the shaft into linear finite elements with inertias centered at nodes. Each node has four degrees of freedom: two translational motions and two rotational motions. The two reduced order models are reductions of the lumped mass model.

The static eigenmode method computes the eigenmodes of the shaft-bearing-disk system at a nominal shaft speed. The method simulates the modal degrees of freedom rather than the node degrees of freedom. Since the number of modal degrees of freedom \ll the number of node degrees of freedom, the eigenmode model reduction significantly speeds up the simulation. The static eigenmode method does not adjust modal properties as the shaft speed changes, so large gyroscopic effects and changes in bearing properties with shaft speed limit simulation accuracy.

The speed-dependent eigenmode method computes the eigenmodes of the shaft-bearing-disk system at varied reference shaft speeds. During the simulation, the model updates the modal properties as speed-dependent gyroscopic and bearing properties change to improve simulation accuracy.

Click the button below to compare the performance of the different reduced order models with varied settings.

Compare Reduced Order Models

```
% This script simulates the model multiple times using the three different
```

```
% vibration modeling methods and plots the computational performance.  
WindTurbineDrivelineWithVibrationsPerformanceComparison;  
% Uncomment the line below to see the code for generating the figure.  
% edit WindTurbineDrivelineWithVibrationsPerformanceComparison
```

You can adjust lines 36-41 in WindTurbineDrivelineWithVibrationsPerformanceComparison.m to investigate the effects of increased lumped elements and modes.

