SimDriveline

For Use with Simulink®

◼ Modeling

◼ Simulation

◼ Implementation

User's Guide

Version 1

The MathWorks

How to Contact The MathWorks:

	`www.mathworks.com`	Web
	`comp.soft-sys.matlab`	Newsgroup
	`support@mathworks.com`	Technical Support
	`suggest@mathworks.com`	Product enhancement suggestions
	`bugs@mathworks.com`	Bug reports
	`doc@mathworks.com`	Documentation error reports
	`service@mathworks.com`	Order status, license renewals, passcodes
	`info@mathworks.com`	Sales, pricing, and general information
☎	508-647-7000	Phone
	508-647-7001	Fax
✉	The MathWorks, Inc. 3 Apple Hill Drive Natick, MA 01760-2098	Mail

For contact information about worldwide offices, see the MathWorks Web site.

SimDriveline User's Guide
© COPYRIGHT 2004 – 2005 by The MathWorks, Inc.

Trademarks

MATLAB, Simulink, Stateflow, Handle Graphics, Real-Time Workshop, and xPC TargetBox are registered trademarks of The MathWorks, Inc.

Other product or brand names are trademarks or registered trademarks of their respective holders.

Patents

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

Revision History

Contents

Introducing SimDriveline

1

Simple Models

2

i

Advanced Methods

3

Blocks — Categorical List

4

5 | **Blocks — Alphabetical List**

A | **Technical Conventions**

B | **Bibliography**

Index

1

Introducing SimDriveline

With SimDriveline, you can model drivetrain and powertrain systems in an easy, natural way within MATLAB® and Simulink®. This chapter introduces you to SimDriveline, with an overview and an example of modeling drivetrains.

What Is SimDriveline?

SimDriveline is a block diagram modeling environment for the engineering design and simulation of drivelines, or idealized powertrain systems. The function of a driveline is to propel a vehicle or craft by transferring its engine torque and power into vehicle momentum and kinetic energy. Drivelines consist of bodies spinning around fixed axes and subject to Newton's laws of motion. The bodies can revolve about one axis, multiple parallel axes, or multiple nonparallel axes. Simple and complex gears constrain the bodies to revolve together and transfer torque up and down the driveline axes. Locking and unlocking clutches switch the driveline from one gear set to another. Gears and clutches make up transmissions.

With SimDriveline, you represent a driveline machine with a connected block diagram, like other Simulink models, and you can group blocks into hierarchical subsystems. You can initiate and maintain rotational motion in a driveline with actuators while measuring, via sensors, the motions of driveline elements and the torques acting on them. You can return sensor signals to the driveline via actuators, forming feedback loops and the basis for controls.

The SimDriveline libraries offer blocks to represent rotating bodies; gear constraints among bodies; special dynamic elements such as spring-damper forces, rotational stops, and clutches; transmissions; and sensors and actuators. SimDriveline also lets you analyze linearized versions of your models and generate code from them.

SimDriveline and Physical Modeling

SimDriveline is part of Simulink Physical Modeling, encompassing the modeling and design of systems according to basic physical principles. Physical Modeling runs within the Simulink environment and interfaces seamlessly with the rest of Simulink and with MATLAB. Unlike other Simulink blocks, which represent mathematical operations or operate on signals, Physical Modeling blocks represent physical components or relationships directly.

Note This SimDriveline User's Guide assumes that you have some experience with building and running models in Simulink.

Related Products

The MathWorks provides several products that are especially relevant to the kinds of tasks you can perform with SimDriveline.

Requirements for SimDriveline

You must have the following products installed to use SimDriveline:

- MATLAB 7.1 or later
- Simulink 6.3 or later

Other Related Products

The related products listed on the SimDriveline product page at the MathWorks Web site include toolboxes and blocksets that extend the capabilities of MATLAB and Simulink. These products will enhance your use of SimDriveline in various applications.

Physical Modeling Product Family

In addition to SimDriveline, the Physical Modeling product family includes SimMechanics, for modeling and simulating general mechanical systems, and SimPowerSystems, for modeling and simulating electrical power systems. Use these products together to model physical systems in Simulink.

For Information About MathWorks Products

For more information about any MathWorks software products, see either

- The online documentation for that product if it is installed
- The MathWorks Web site at www.mathworks.com; see the "Products" section

Running a Demo Model

The demo model of this section, `drive_simpson_ideal`, simulates a complete drivetrain. This model will help you understand how to model driveline components with SimDriveline blocks, connect them into a realistic model, use Simulink blocks as well, and simulate and modify a drivetrain model.

The driveline mechanism modeled here 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.

What the Model Illustrates

The `drive_simpson_ideal` model contains a driveline that accepts a driving torque and 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 Simpson four-speed transmission subsystem, based on two gears and four clutches. You can set the transmission to four different gear combinations, allowing four different effective torque and angular velocity ratios. A fifth clutch acts as a brake on the driven shaft.

The Simpson 4-Speed Transmission subsystem illustrates a critical feature of transmission design, the *clutch schedule*. To be fully engaged, the transmission, with four clutches and two gears, requires two clutches to be locked and the other two unlocked at any time. The choice of which two clutches to lock determines the effective gear ratio across the transmission. The clutch schedule is 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 completely disengaged, no torque or angular motion at all is transferred across the transmission.

Clutch Schedule for the Simpson 4-Speed Transmission

Gear Setting	Clutch A State	Clutch B State	Clutch C State	Clutch D State
1	*L*	F	F	*L*
2	*L*	F	*L*	F
3	*L*	*L*	F	F
4	F	*L*	*L*	F

L = locked, F = free

Opening the Model

You can open the Simpson transmission demo in several ways:

- Enter drive_simpson_ideal at the MATLAB command line.

- Open the MATLAB help browser from the MATLAB desktop. In the navigator pane to the left, click the **Demos** tab. Open the **Simulink** node, then the **SimDriveline** subnode. Under **Transmission & Car Models**, locate the Simpson 4-Speed Transmission demo entry and double-click it.

Examine the model and its structure. Open each subsystem in turn.

- The main model window contains the Simpson transmission subsystem, the input or driver shaft assembly, and the output or driven shaft assembly. Each assembly consists of a wheel with applied kinetic friction. The driver shaft transmits an externally specified torque down the driveline.

  The main model also includes a brake clutch. When this clutch is locked, the driven shaft stops turning. This clutch must remain unlocked if the Simpson transmission is engaged.
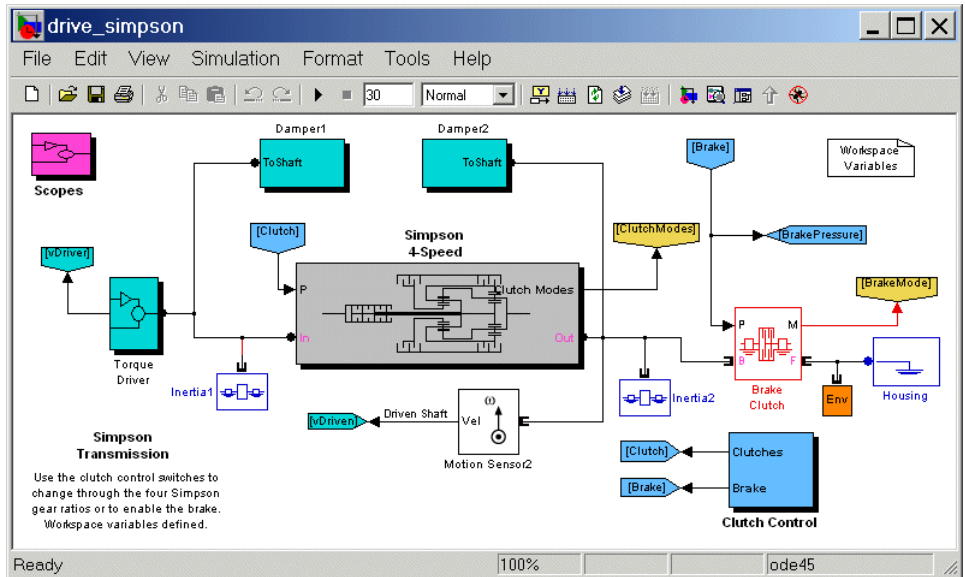
Main Model Window

- The Simpson 4-Speed Transmission subsystem is a set of four clutches, two planetary gears, and four inertias (rotating bodies). Within the subsystem, open the clutch schedule block to see the four possible gear settings for the Simpson 4-speed transmission. Two clutches must be locked at any one time for the transmission to be engaged. Exactly two clutches must be engaged simultaneously to avoid conflicting constraints on the gear motions.

Simpson 4-Speed Transmission Subsystem
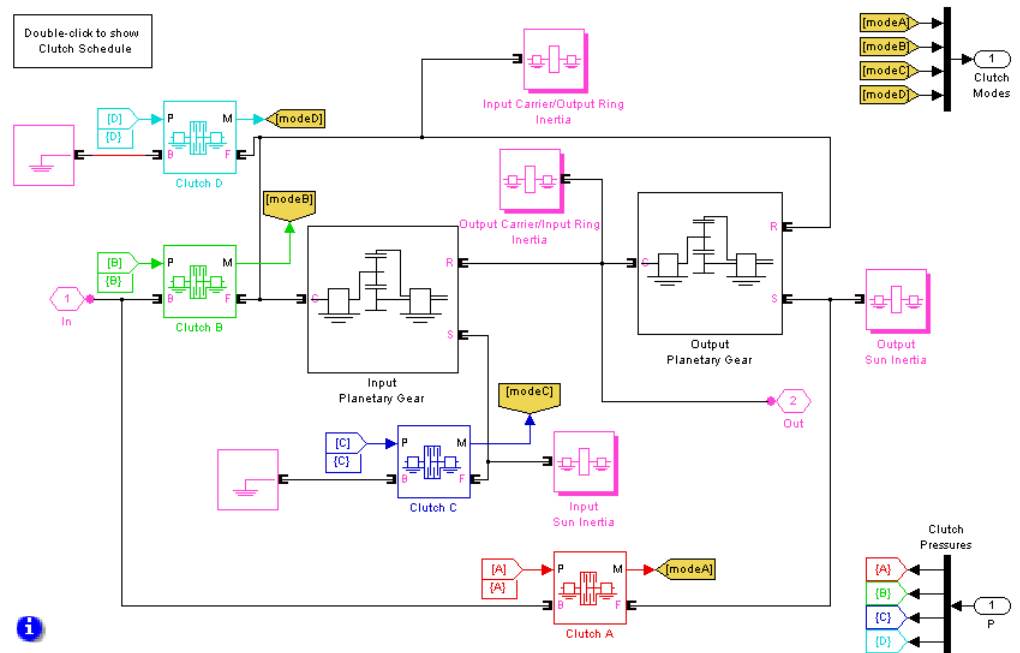
- The Clutch Control subsystem provides the pressures that lock the
  necessary clutches. The clutch controller is programmed to move the
  transmission through a fixed sequence of gears, then unlock all the
  transmission clutches, allowing the driven shaft to "coast" for a time, and
  then engage and lock the brake clutch to stop the driven shaft.
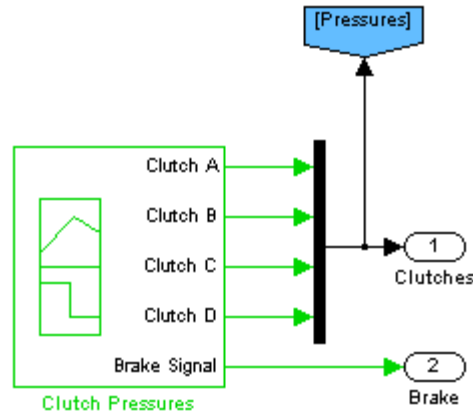
Clutch Control Subsystem

- The Scopes subsystem provides Scope blocks to display the clutch pressure,
  driver and driven shaft velocity, and clutch mode signals.
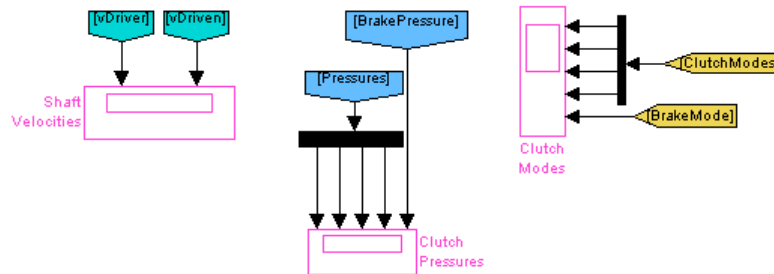
Scopes Subsystem

Running the Model

To display the Simpson driveline model's behavior,

1 Open the Scopes subsystem and then each of the Scope blocks. 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, 4, 8, and 12 seconds of simulation time. Compare these clutch pressure signals to the clutch schedule in the Simpson transmission subsystem to determine which gear settings the model is implementing. (In fact, the model steps through gears 1, 2, 3, and then 1 again, before coasting and then braking.)
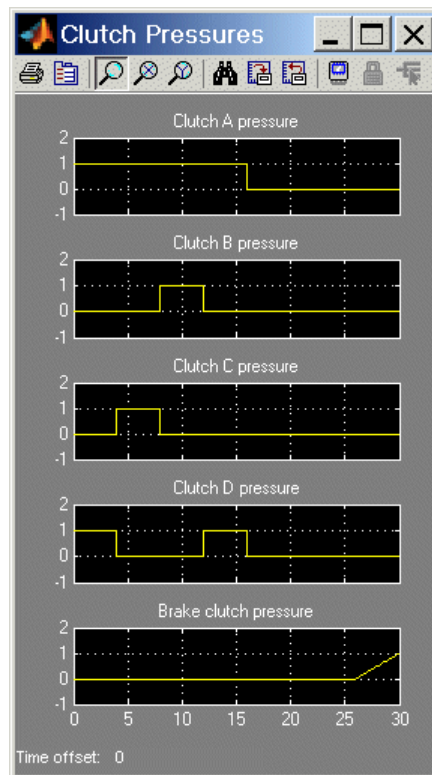
4 Observe the clutch modes at the same time. When a clutch mode is zero, that clutch is locked. The sequence of clutch locking and unlocking matches the sequence from the clutch schedule.
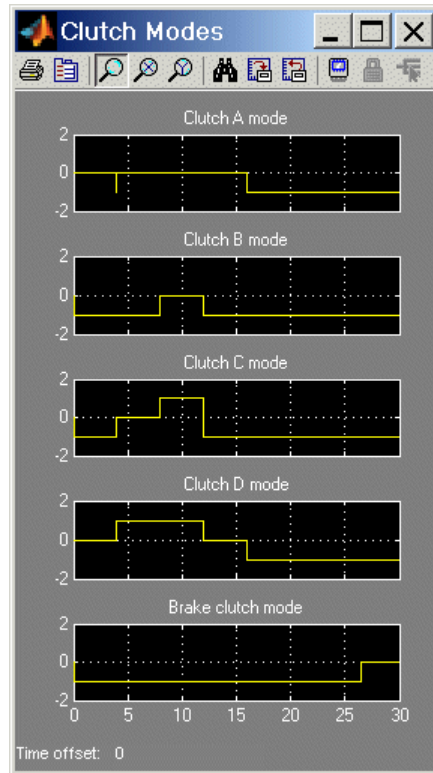
5 Compare the angular velocities of the driven and driver shafts. The effect of the transmission is the result of the two planetary gears coupled in different ways in the different gear settings. The effective *drive ratio* of output to input shafts is the *reciprocal* of the ratio of output to input angular velocities.
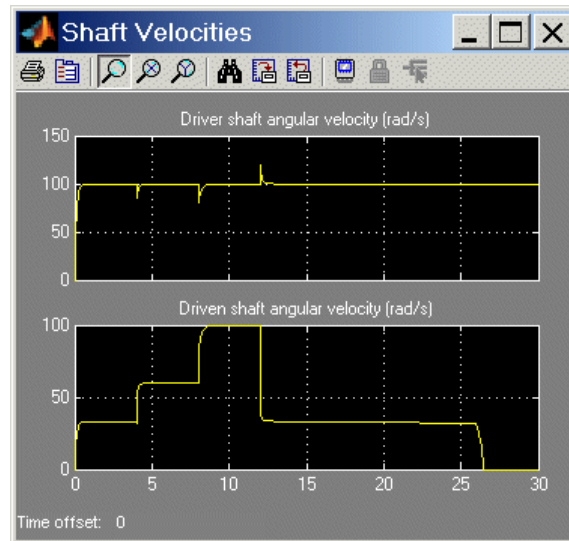
6 Observe what happens at 16 seconds. The transmission clutch pressures drop to zero, and the transmission disengages. The transmission ceases to transfer angular motion and torque from the driver to the driven shaft, and the driven shaft continues to spin simply from inertia. A small kinetic friction damping gradually slows the driven shaft over the next 10 seconds.
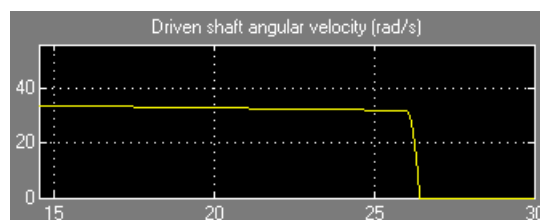
7 At 26 seconds of simulation time, the brake clutch pressure begins to rise from zero, and the brake clutch engages. The driven shaft decelerates more drastically now. At about 26.4 seconds, the brake clutch locks, and the driven shaft stops rotating completely.
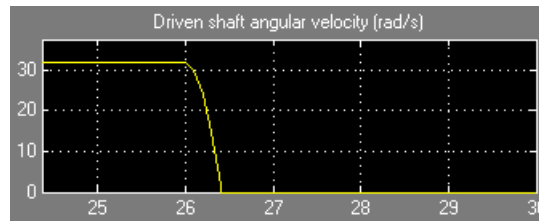
Modifying the Model

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

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

- Change the gear sequence.

Measuring the Drive Ratio of the Simpson Transmission States

The gear ratio (output to input) is the ratio of the output gear wheel radius to the input gear radius. Equivalently, the gear ratio is the ratio of the number of teeth on the output gear wheel to the number on the input wheel, or the ratio of the output torque to the input torque. The ratio of the angular velocities of output to input is the reciprocal of this gear ratio.

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

Add and connect the Simulink blocks necessary to measure the drive ratio
of the transmission.

1 From the Simulink Math Operations library, copy a Divide block and, from
the Simulink Sinks library, copy a Scope block.

2 From the Torque Driver subsystem outport, branch a signal line from
Motion Sensor1's angular velocity output and connect it to the X inport
on the Divide block. From the velocity output of Motion Sensor2, on the
driven (output) shaft, again branch a signal line. Connect it to the ÷ inport
on the Divide block.

The effective output-to-input drive ratio is the ratio of input to output
velocities.

3 Connect the outport of the Divide block to the Scope. Rename Scope to
Drive Ratio.
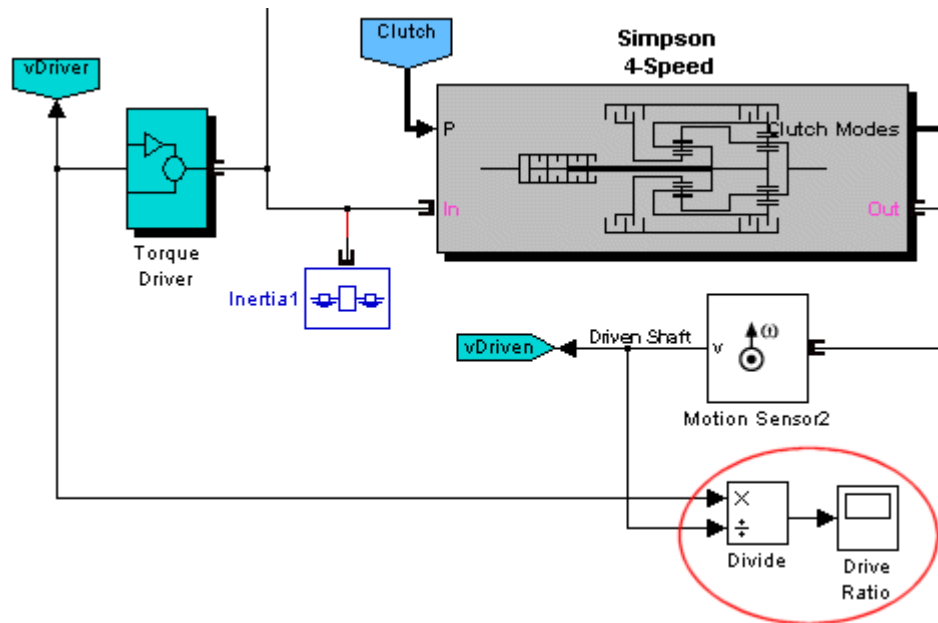
Simpson 4-Speed Model with Drive Ratio Measurement

4 Open the Drive Ratio scope and restart the demo. Observe how the drive ratio steps through a sequence of four-second states, in parallel with the clutch pressures and clutch modes, until it reaches 16 seconds. The drive ratio measurement after 16 seconds is not meaningful because the transmission is uncoupled.

Just after 26 seconds, the driven shaft velocity drops to zero, and the Divide block will produce divide-by-zero warnings at the MATLAB command line.
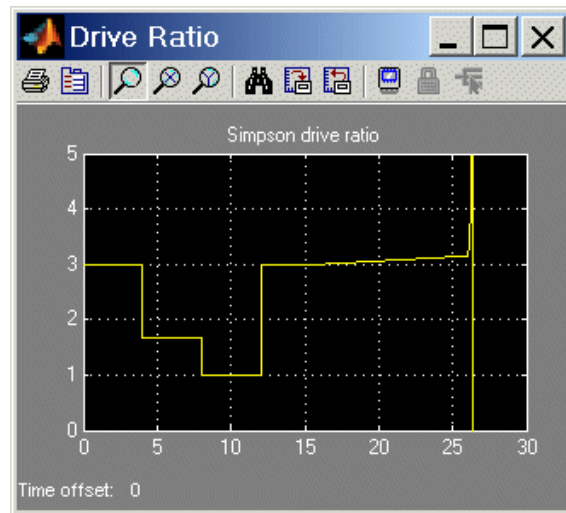
5 Look inside the Simpson 4-Speed Transmission subsystem for the Clutch Schedule block and open it. Consult the drive ratios for each gear, 1, 2, and 3, in terms of the gear ratios of the transmission's two Planetary Gears. Determine the numerical values for these drive ratios for gear settings 1, 2, and 3, and check them against the values displayed in the Drive Ratio scope.

The drive ratio sequence should be 3, 5/3, 1, and 3, respectively, for the first, second, third, and fourth intervals of four seconds each.

Changing the Transmission Gear Sequence

The drive_simpson_ideal demo, when you open it, is programmed to step through Simpson gear settings 1, 2, 3, then 1, before disengaging. Modify it to step through settings 1, 2, 3, then 2, then disengage. The first gear requires A, B, C, and D to be locked, free, free, and locked, respectively. You will modify the clutch pressure signal sequence from 12 to 16 seconds so that the transmission is set in second, not first, gear. The second gear requires clutches A, B, C, and D to be locked, free, locked, and free, respectively.

1 Open the Clutch Control and Simpson transmission subsystems. Within the transmission, open the Clutch Schedule block and review the clutch lockings for each gear setting.

2 Open the Signal Builder block, labeled Clutch Pressures, to view the clutch pressure signals.

Modify clutch pressure signals C and D so that, between 12 and 16 seconds, clutch C is locked (not free) and clutch D is free (not locked). Sufficient pressure will lock the clutches, while zero input pressure leaves a clutch unlocked.
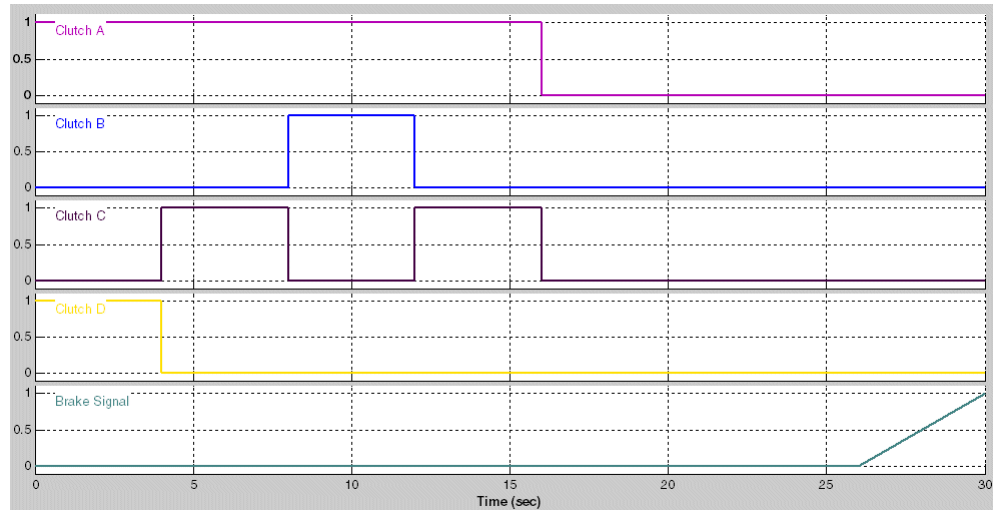
Modified Simpson 4-Speed Transmission Clutch Pressures

3 Restart the model. Observe that between 12 and 16 seconds of simulation time the clutch pressures, the clutch modes, and the driven shaft velocity are now different from the original version of the model.

Check the effective drive ratio between 12 and 16 seconds to confirm that the Simpson transmission during that time is set in gear 2, not gear 1. This fourth interval of four seconds should exhibit a drive ratio of 5/3 instead of 3.
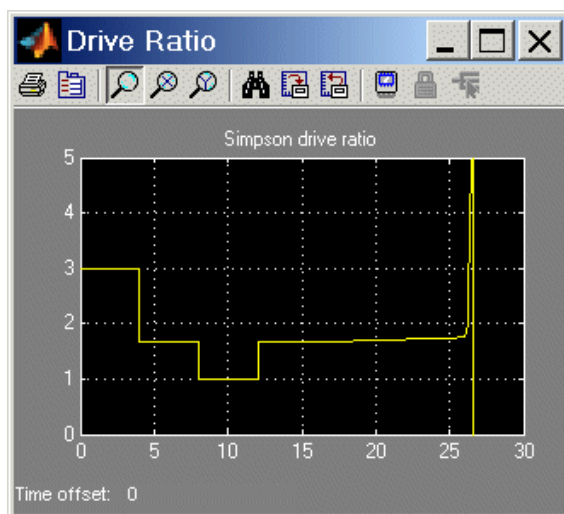
What Can You Do with SimDriveline?

SimDriveline is a set of block libraries and special simulation features for use in the Simulink environment. You connect SimDriveline blocks to normal Simulink blocks through special Sensor and Actuator blocks.

The blocks in these libraries are the elements you need to model driveline systems consisting of any number of rotating inertias, rotating about one or more axes, constrained to rotate together by gears, which transfer torque to different parts of the driveline. SimDriveline can represent machines with components organized into hierarchical subsystems, as in normal Simulink models. You can add complex dynamic elements such as clutches and transmissions, actuate bodies with external torques or motions, integrate the Newtonian rotational dynamics, and measure the resulting motions.

Modeling Drivetrains with SimDriveline

SimDriveline extends Simulink with blocks to specify a driveline's components and properties and to solve the machine's equations of motion. The blocks are similar to other Simulink blocksets, with some properties unique to SimDriveline.

These are the major steps you follow, using SimDriveline, to build and run a model representation of a driveline machine:

1 Specify rotational inertia for each body and connect the bodies with driveline connection lines representing driveline axes. If needed, ground the driveline to one or more housings fixed in space.

2 Constrain the driveline axes to rotate together by connecting them with gears. Gears impose static constraints on driveline motions and transfer torques at fixed ratios.

3 As necessary and desired, add dynamic driveline elements that transfer torque and motion among driveline axes in a nonstatic way. These elements include internal torque-generating components such as damped springs, clutches and transmissions, and torque converters. You can also construct and connect your own dynamic elements.

4 Set up actuators and sensors to initiate and record body motions, as well as apply external torques to the driveline.

5 Connect the SimDriveline motion solver to the machine and configure it. Start the simulation, calling the Simulink solvers to find the motions of the system. Display and analyze the motion.

Connector Ports and Connection Lines

Most SimDriveline blocks have special driveline ports ⊏. You connect driveline ports with driveline connection lines, distinct from normal Simulink lines. Driveline connection lines represent physical rotation axes along which torque is transferred and around which inertias rotate.

- You can connect driveline ports only to other driveline ports.

- The driveline connection lines that connect driveline ports together are not normal Simulink lines, which carry signals or indicate mathematical operations. You cannot connect driveline lines directly to Simulink inports and outports >.

- Two directly connected driveline components must corotate at the same angular velocity.

- You can branch SimDriveline connection lines. When you do so, components directly connected with one another continue to share the same angular velocity. The torque transferred along the driveline axis is divided among the multiple components connected by the branches. How the torque is divided is determined by the driveline dynamics.

  The sum of all torques flowing into a branch point equals the sum of all torques flowing out.

Inertias and Gears

SimDriveline defines a driveline as a collection of bodies rotating about driveline axes represented by connection lines. The bodies are defined by their rotational inertias. The lines carry the rotational degrees of freedom (DoF) and, unconstrained, rotate freely. Directly connecting one body to another constrains both bodies to rotate at the same angular velocity. A torque applied to one body is effectively applied to both.

You can also ground driveline axes to housings that do not move and that represent infinite effective inertia.

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:

- They constrain the connected axes to corotate at angular velocities in fixed ratio or ratios.

- They transfer the torques flowing along one or more axes to other axes, also in fixed ratio or ratios.

Complex Driveline Elements

Note These blocks serve as suggestions for developing variant or entirely new models to simulate the same components. You can study these subsystems by looking under their masks. If necessary, break the block's library link before modifying it, and then create your own version. Or create your own completely new block from scratch.

To create more realistic driveline models, you elaborate on simple drivelines consisting of inertias and gears by adding complex mechanical elements that generate torques internally within the driveline, between one axis and another. SimDriveline includes blocks that encapsulate as subsystems entire models of complex driveline elements:

- Clutches that model the locking and unlocking of pairs of driveline axes by applying kinetic and static friction

- Transmission models that incorporate multigear sets and clutches into a single subsystem

- Vehicle component models that represent engines, tires, and vehicle dynamics

- Specialized torque models, such as torque converters, bilateral stops, and damped spring-like torsion

Actuating and Sensing Motion

Sensors and Actuators are the blocks you use to interface between SimDriveline blocks and normal Simulink blocks:

- Actuator blocks impart motion to driveline axes, either at zero time or through the course of a simulation, and impose externally defined torques on the bodies of a driveline.

- Sensor blocks measure the motions of, and the torques transferred along, the axes of a driveline system.

Actuating inputs and sensor outputs are Simulink signals that you can define and use like any other Simulink signal. For example, you can connect a Sensor output to a Simulink Scope block and display the torques in a driveline as functions of time.

Simulating and Analyzing Motion

Once you specify all the rotational inertias of the bodies and interconnect the bodies with gears and other driveline elements, the dynamical problem of finding the system's motion is solvable. To finish a driveline model and prepare it for simulation, you must connect the machine to the SimDriveline environment. The environment defines the solver that 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, you can run it and analyze its motions and internal torques.

Trimming and Linearizing the Motion

In many cases, you do not know the torques necessary to produce a given set of motions. By motion-actuating your driveline and measuring the resulting torques, you can find the torques necessary to produce a specified motion trajectory. This technique inverts the canonical approach to dynamics, which consists of finding motions from torques.

A special case of inverse dynamics is *trimming*. This technique involves searching for steady-state motions of the bodies, when their angular accelerations and the torques they experience vanish. Using Simulink's

linearization tools, you can perturb such a steady motion state slightly to find how the system responds to small disturbances. The response indicates the system's stability and suitability for controllers.

Generating Code

SimDriveline is compatible with the Simulink Accelerator, Real-Time Workshop®, and xPC Target. These optional products let you generate code versions of the models you create originally in Simulink with block diagrams, enhancing simulation speed and model portability.

The presence of clutches in a driveline model creates dynamical discontinuities and triggers mode iterations in Simulink. These discontinuities and mode iterations place certain restrictions on code generation.

Learning More

You can get help online in a number of ways to assist you while you use SimDriveline.

Using the MATLAB Help System for Documentation and Demos

The MATLAB help browser allows you to access the documentation and demo models for all the MATLAB and Simulink based products that you have installed. The online help includes an online index and search system.

Consult the "Help Browser" section of the Getting Started with MATLAB documentation for more about the MATLAB help system.

Finding Special SimDriveline Help

This user's guide includes special reference chapters for use with SimDriveline.

- Appendix A, "Technical Conventions" explains special conventions, abbreviations, and units.

- Appendix B, "Bibliography" lists external references on driveline and powertrain modeling and related topics.

2

Simple Models

This chapter introduces you to modeling drivetrains with SimDriveline. After showing you how you how to access the SimDriveline block library and reviewing the essential rules of connecting blocks and transferring angular motion and torque, it moves you from modeling simple gears to simulating a full car in a series of short tutorials.

This user's guide assumes that you are familiar with building models in Simulink. If not, see the Using Simulink documentation.

Introducing the SimDriveline Block Libraries

SimDriveline is organized into a set of libraries of closely related blocks. This section shows you how to open these SimDriveline block libraries and explains the nature of each library.

- "Accessing the SimDriveline Block Library" on page 2-2
- "Using the Libraries" on page 2-4

Accessing the SimDriveline Block Library

There are several ways to open the SimDriveline block library on Microsoft Windows and UNIX platforms.

Microsoft Windows Platforms

Microsoft Windows users can access the blocks through the Simulink Library Browser. Expand the SimDriveline entry in the contents tree.
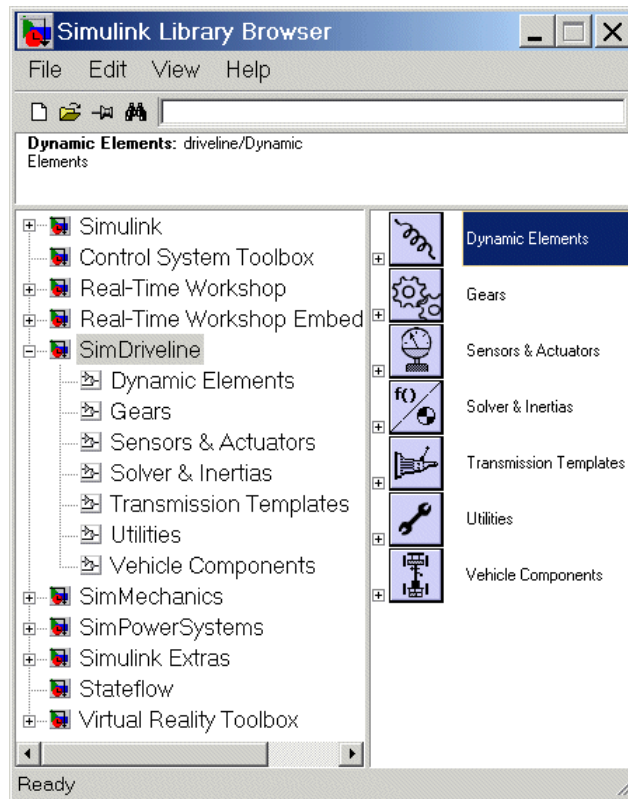
You can also access the blocks directly inside the SimDriveline library in several ways:

- In the Simulink Library Browser, right-click the SimDriveline entry and select **Open the SimDriveline Library**. The library appears.

- Click the **Start** button in the lower-left corner of your MATLAB desktop. In the pop-up menu, select **Simulink**, then **SimDriveline**, then **Block Library**.

- Enter `drivelib` at the MATLAB command line prompt.

UNIX Platforms

UNIX users can click the Simulink icon on the MATLAB menu bar, open the Blocksets & Toolboxes library and then SimDriveline. You can also enter `drivelib` at the command line.

The SimDriveline Library

Once you perform one of these steps, the SimDriveline library opens. This library displays seven top-level block groups. You can expand each library by double-clicking its icon.
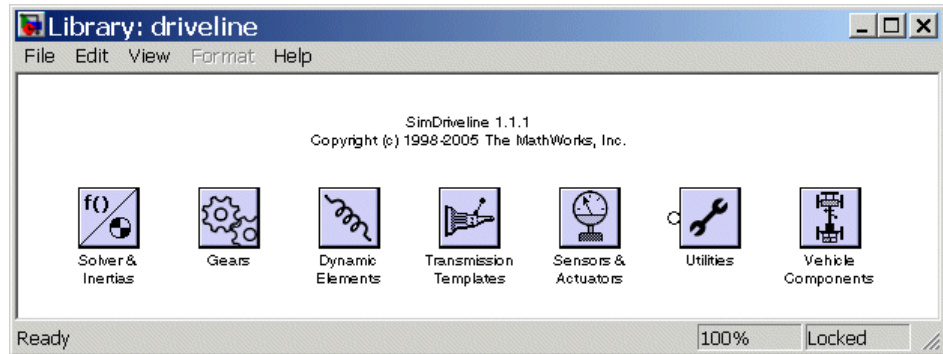
The next section summarizes the blocks of each library and their use. For explanations of individual blocks, consult the complete SimDriveline "Blocks — Categorical List".

Using the Libraries

The SimDriveline block library is organized into seven separate libraries, each with a different type of driveline block.

Solver & Inertias

The Solvers & Inertias library provides the `Inertia` block, which represents a user-defined rotating body specified by its moment of inertia, the fundamental unit of driveline modeling. It also contains the `Housing` block, which represents an immobile rotational ground.

Finally, the library contains the `Driveline Environment` block, which configures the driveline settings of a SimDriveline block diagram, and the `Shared Environment` block, which allows you to connect two driveline block diagrams in a nonphysical way so that they share the same driveline environment settings.

Gears

The Gears library contains blocks that represent simple and complex gears, driveline elements that couple distinct driveline axes and constrain their relative motions. The Gear blocks range from simple two-wheel gear couplings with fixed and variable gear ratios, to complex multiwheel and multiaxis gears such as planetary and differential gears.

Dynamic Elements

The Dynamic Elements library contains blocks that model such critical drivetrain components as clutches, torque converters, damped springs, and stops. Dynamic elements generate internal driveline torques.

The blocks of this library serve as suggestions for developing variant or entirely new models to simulate the same components. Look under the block mask, break the block's library link before modifying it, and create your own version.

Transmission Templates

The Transmission Templates are a set of predesigned transmission examples constructed from gears, clutches, and inertias. You can copy and use these examples in your drivetrain models.

Transmission templates copied into your model are not linked to the block library. You can modify and rebuild these template copies at will.

Sensors & Actuators

The Sensors & Actuators library provides blocks for sensing and initiating the motions of driveline axes and applying and sensing torques along those axes.

Utilities

The Utilities library contains miscellaneous blocks useful in building models.

Vehicle Components

The Vehicle Components library contains blocks that represent components of a full vehicle beyond the drivetrain itself. It includes models of engines, wheeled vehicles, and tires in contact with the ground.

The blocks of this library serve as suggestions for developing variant or entirely new models to simulate the same components. Look under the block mask, break the block's library link before modifying it, and create your own version.

Essential Steps to Building a Driveline Model

The demo model of Chapter 1, "Introducing SimDriveline" illustrates a typical drivetrain system you can model with SimDriveline. It also illustrates the key rules for connecting driveline blocks to each other and the dual roles of driveline connection lines: transferring torque and enforcing angular velocity constraints. You should review these rules before building and running the tutorial models of this chapter.

- Driveline blocks, in general, feature both driveline connector ports ⊏ and regular Simulink inports and outports >. You connect connector ports to one another and Simulink ports to one another. But you cannot connect a driveline port to a Simulink port.

- The driveline connection lines interconnecting driveline connector ports ⊏ represent driveline axes and enforce physical relationships. Unlike 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. A driveline connection line between two ports enforces the constraint that the two driveline components so connected rotate at the same angular velocity. The connection line also transfers any torque applied to a driveline component at one end to the component at the other end.

- You can branch driveline connection lines. You must connect the end of any branch of a driveline connection line to a driveline connector 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 at the same angular velocity. The torque transferred along the input driveline axis is split up among the branches. How the torque is split depends on the dynamical details of the system that you are modeling.

- Driveline connection lines satisfying the angular velocity constraint must have the same initial angular velocities.
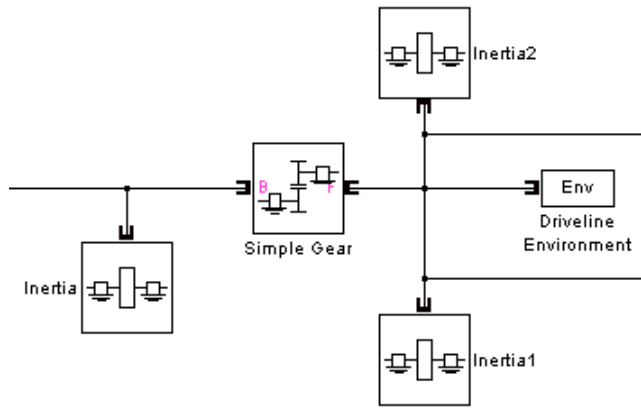
Branching Driveline Connection Lines

The Driveline Environment block 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

$\omega = \omega_1 = \omega_2 = \omega_3 \ldots$

$\tau = \tau_1 + \tau_2 + \tau_3 \ldots$

The driveline axes have an implicit directionality. Torque and motion are transferred "down" the driveline from input or drive shafts to output or driven shafts. Certain SimDriveline 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). Relative motion of driveline axes or shafts, when needed, is measured as follower relative to base.

All motion in SimDriveline models, except when relative motion is explicitly required, is measured in implicit absolute coordinates. An absolute orientation defines zero angle, and an absolute reference frame defines zero angular velocity. The Housing block implements the absolute zero angular velocity and, if connected to a driveline axis, enforces this zero-motion state on that axis.

Coupling Motion and Transferring Torque with Gears

The purpose of a gear set is to transfer rotational motion and torque at a known ratio from one driveline axis to another. This section introduces you to modeling gears and using them to couple bodies rotating on driveline axes.

- "Coupling Rotational Motion with Gears" on page 2-9
- "Coupling Two Spinning Inertias with a Simple Gear" on page 2-10
- "Coupling Two Spinning Inertias with a Variable Gear" on page 2-15
- "Coupling Three Spinning Inertias with a Planetary Gear" on page 2-17

Coupling Rotational Motion with Gears

A gear set consists of two or more meshed gears corotating at some specified gear ratios. The ratios might or might not be 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 corotate 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: $g_{21} = r_2/r_1$. 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 $g_{21} = N_2/N_1 = r_2/r_1$.

The fundamental conditions on the simple gear coupling of rotational motion are $\omega_2/\omega_1 = \pm 1/g_{21}$ and $\tau_2/\tau_1 = \pm g_{21}$. 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 (corotating on their respective outside surfaces), they rotate in opposite directions. If the gears are internal to one another (corotating with the outside of the smaller gear meshing with inside of the larger gear), they rotate in the same direction.

Caution   Gear ratios should always be strictly positive. If a gear ratio in SimDriveline vanishes or becomes negative, the simulation stops with an error.

Generalized Gear Coupling Rules

You need the general ideal gear coupling conditions if you are coupling gears that are not constant in radii, not lying in the same plane, or not circular.

The general velocity constraint requires that the linear velocities of the gears at the point of contact be the same. This is a vector condition on the angular velocities ω_1 and ω_2 and the radius vectors r_1 and r_2: $\omega_1 \times r_1 = \omega_2 \times r_2$. The alternative form in terms of the number of gear teeth is equivalent to this linear velocity constraint. For the gear teeth to mesh, the number of teeth per unit length of gear circumference must be the same on the two gears.

The general torque condition arises from the force equilibrium at the point of contact. If there is no linear motion of the whole gear assembly, the forces at contact F must be equal and opposite. The ratio of torques is then:

$$|\tau_2|/|\tau_1| = |r_2 \times F|/|r_1 \times F|$$

The power transferred along either shaft is conserved across ideal gear couplings:

$$\omega_2 \cdot (r_2 \times F) = \omega_1 \cdot (r_1 \times F)$$

Coupling Two Spinning Inertias with a Simple Gear

In this example, you couple two spinning inertias, first, along a single shaft (driveline axis), so that they spin with the same angular velocity; then spinning along two shafts and coupled by a gear so that they spin at different velocities; and finally, coupled by a gear and actuated by an external torque,

spinning at different rates and experiencing different torques. You use the most basic blocks in SimDriveline, such as `Inertia`, `Simple Gear`, and `Driveline Environment`.

Modeling Two Spinning Inertias

Here you create the first version of the simplest driveline model, two inertias spinning together along the same axis. Open the SimDriveline and Simulink block libraries and a new Simulink model window.

1 Drag and drop two Inertia, two Motion Sensor, and one Initial Condition blocks into the model window.

2 Every topologically distinct driveline block diagram requires exactly one Driveline Environment block, found in the Solver & Inertias library of SimDriveline. Copy one such block into your model.

3 From the Simulink library, drag and drop a Scope and a Mux block. Then connect the blocks as shown.
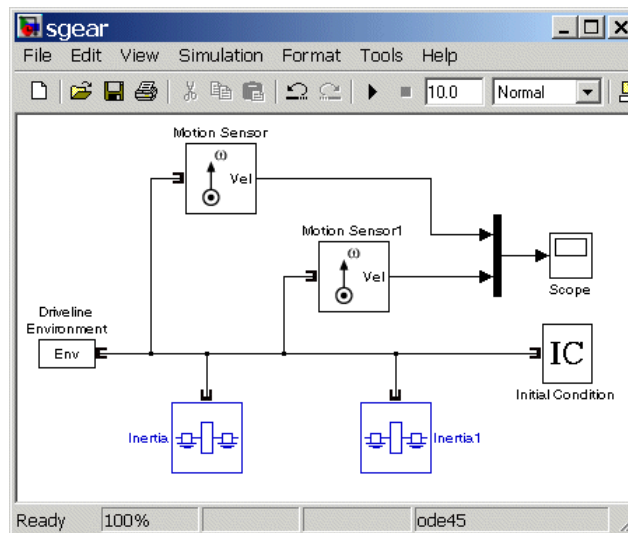
Two Spinning Inertias

4 Open the Initial Condition block. In the **Initial angular velocity** field, replace its default 0 entry with pi radians/second (rad/s). Click **OK**.

If you do not connect an Initial Condition block to a driveline axis, SimDriveline assumes by default that the axis starts the simulation with zero angular velocity. You must ensure that the initial angular velocities of your coupled driveline axes are consistent with one another. If they are not, the simulation stops with an error.

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

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

1 From the SimDriveline block library, drag and drop a Simple Gear block into your model. Open the block. Leave the default follower-base gear ratio value at 2. Clear the **Follower and base rotate in opposite directions** check box and click **OK**. The simple gear then represents two gear wheels corotating in the same direction, with the smaller wheel inside the larger. Reconnect the blocks as shown.
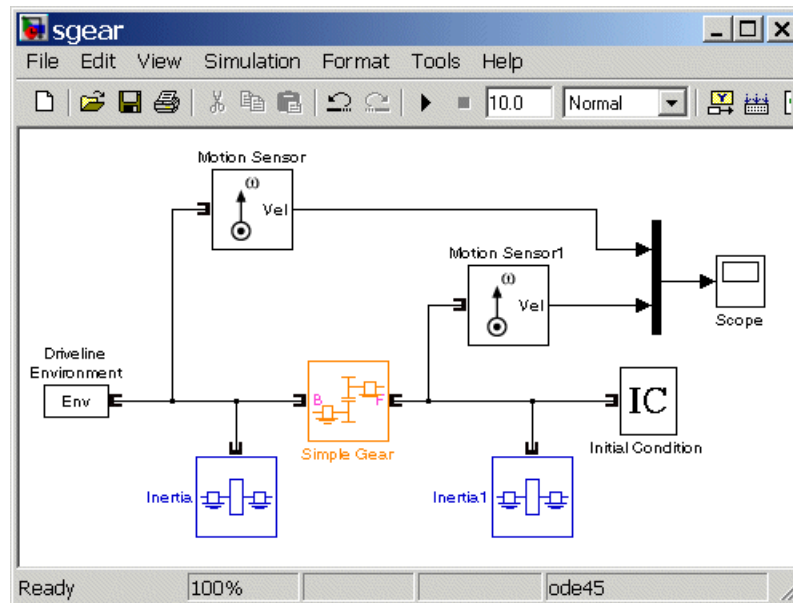
Two Spinning Inertias Coupled by a Gear

Leave the initial angular velocities at `pi` in the Initial Condition block. SimDriveline automatically sets the correct initial angular velocity for Inertia.

2 Open the Scope and start the simulation. The two angular velocities are constant at 3.14 and 6.28 radians/second for Inertia1 and Inertia, respectively. The follower-base gear ratio is 2, and the angular velocity of Inertia is twice that of Inertia1, with the same sign, because the two bodies are spinning in the same direction.

3 Now select the **Follower and base rotate in opposite directions** check box. The simple gear then becomes two wheels corotating in opposite directions, with the two wheels meshed on their respective outer surfaces.

4 Restart the simulation. The two angular velocities are 3.14 and -6.28 radians for Inertia1 and Inertia, respectively. The second angular velocity is twice the first and with opposite sign, because the two bodies are spinning in opposite directions.

5 Finally, again clear the **Follower and base rotate in opposite directions** check box.

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 this completed model in the demo drive_sgear.

1 From the SimDriveline block library, copy a Torque Actuator and two Torque Sensor blocks. From the Simulink block library, drag and drop a second Scope block, a second Mux block, and a Sine Wave block.

2 Disconnect the Inertia blocks from the Simple Gear and insert the Torque Sensors. Disconnect and delete the Initial Condition block. SimDriveline will now impose zero angular velocities on the two axes.

Connect the other blocks as shown.
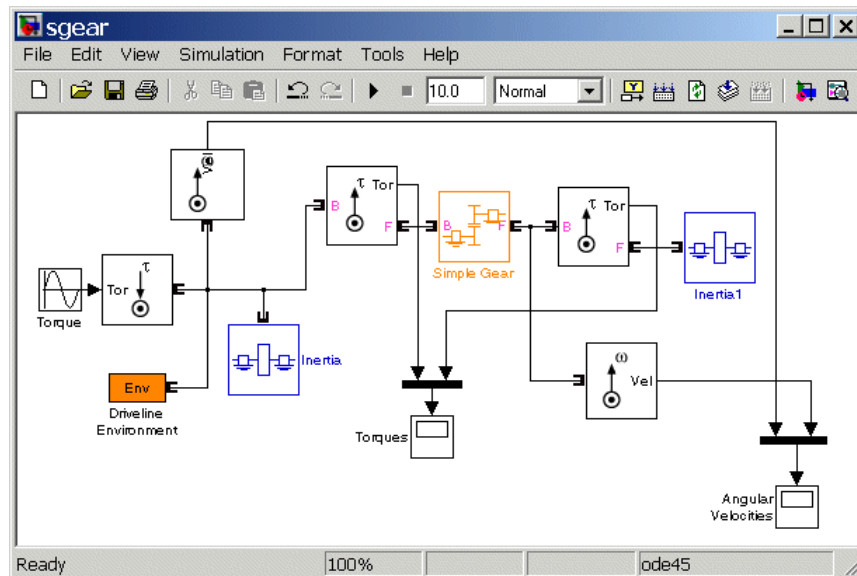
Two Spinning Inertias Coupled by a Gear and Actuated with Torque

3 Open both Scope blocks and start the simulation.

The measured torques and angular velocities vary sinusoidally. The angular velocity of Inertia1 is half that of Inertia, as you saw in the previous models. But the torque in the second shaft is twice that in the first, as required by the laws of gear coupling.

If you select the **Follower and base rotate in opposite directions** check box in Simple Gear and restart the simulation, the same angular velocities and torques result, except that the values associated with Inertia1 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 Sensor and Actuator blocks you use in the preceding models illustrate their dual nature: they act as driveline components themselves, but also let you connect driveline blocks with the rest of Simulink.

- Sensor & Actuator blocks have both driveline connector ports ⊏ and normal Simulink ports >. You can extract sensor signal information with a block's Simulink outports. You can actuate motion or apply external torques by feeding in actuation signals with a block's Simulink inports.

  Many other SimDriveline blocks feature Simulink ports for inserting and measuring signals.

- You connect a Torque Sensor *along* a driveline axis, by placing it in series with other driveline components.

- You connect the other Sensor and Actuator blocks *across* a driveline axis, by branching the driveline connection line off to one side and connecting this secondary line to the block; or by connecting the block to the end of a driveline axis.

Coupling Two Spinning Inertias with a Variable Gear

You can modify the simple gear model further by replacing the fixed-ratio gear with a gear whose gear ratio varies in time. You specify the gear ratio variation with a Simulink signal. Start with the simple gear model you built in the preceding section or by opening and editing the `drive_sgear` demo.

1 From the SimDriveline block library, drag and drop a Variable Ratio Gear block and replace the Simple Gear block with it. Open Variable Ratio Gear

and ensure that the **Follower and base rotate in opposite directions** check box is selected (the default). The two shafts will spin in opposite directions.

2 The Variable Ratio Gear block accepts the continuously varying gear ratio as a Simulink signal through the extra inport labeled r. For this example, create a Simulink signal for the gear ratio with a Signal Builder block from the Simulink block library. Build a signal that rises with constant slope from 1 to 2 over 10 seconds. Then connect the Signal Builder block to the r port.
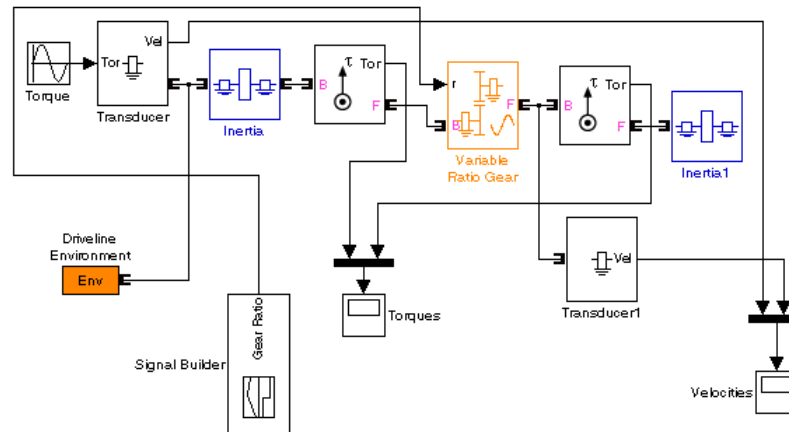
Simple Variable Ratio Gear Model

3 Leave the other, original settings of the simple gear model unchanged. Open both Scopes and start the simulation.

The two shafts' angular velocities and torques have opposite signs. Apart from this sign difference, the ratios of angular velocities and torques start at 1, because the initial gear ratio is 1. But as the gear ratio increases toward 2, the angular velocity of Inertia1 becomes smaller than that of Inertia, while the associated torque in the second shaft (apart from the opposite sign) becomes larger than that 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 drive_vgear demo is a full model of this type. To learn more about how to use variable gears, including the Coriolis acceleration, consult the Variable Ratio Gear block reference page.

Coupling Three Spinning Inertias with a Planetary Gear

You can further modify the simple gear model and use it as a starting point for studying more complex gear sets. One of the most important 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 interesting in its own right, but also important because it is a common component in complex, realistic transmissions.

1 Replace the Simple Gear in your model with a Planetary Gear from the SimDriveline block library. A planetary gear splits input angular motion from the carrier between the ring and sun wheels, each connected to their respective bodies.

2 Copy another Inertia and another Motion Sensor as well. Connect the blocks to form the new diagram as shown.
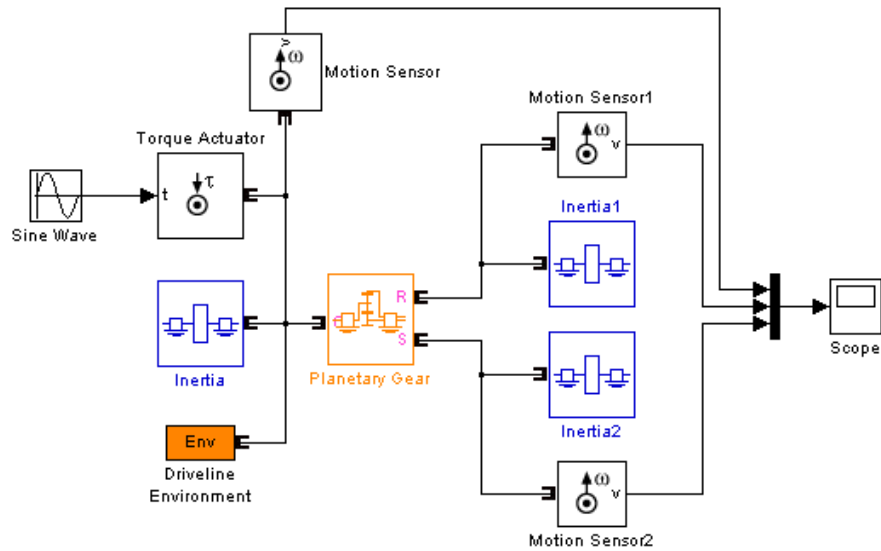
Simple Planetary Gear Model

3 Enter 2 for the **Ring/Sun gear ratio** in Planetary Gear. Open the Scope and start the simulation to observe the angular velocities of the ring, carrier, and sun, from largest to smallest.
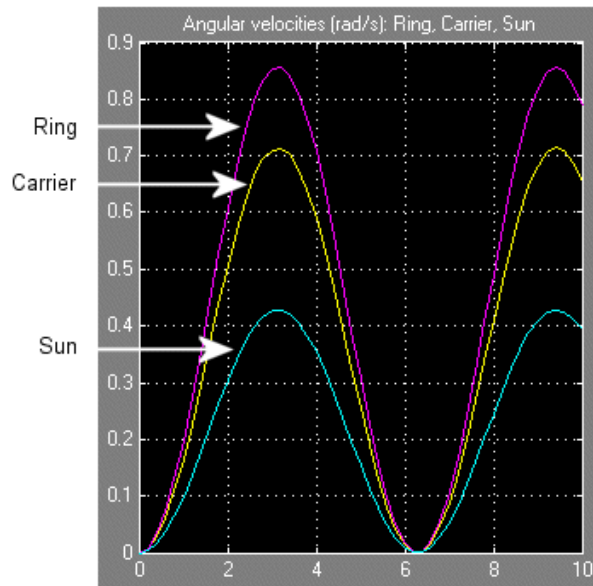
4 To see the ring and sun wheels spinning alone, you must lock the carrier. In this case, you switch the torque actuation to the ring wheel. Copy a Housing block from the SimDriveline block library. Disconnect and delete Inertia, replacing it on the carrier driveline axis with Housing, and reconnect the Driveline Environment block to this connection line.

5 Insert a Torque Actuator and move the Sine Wave block next to it. Connect it to the inport.
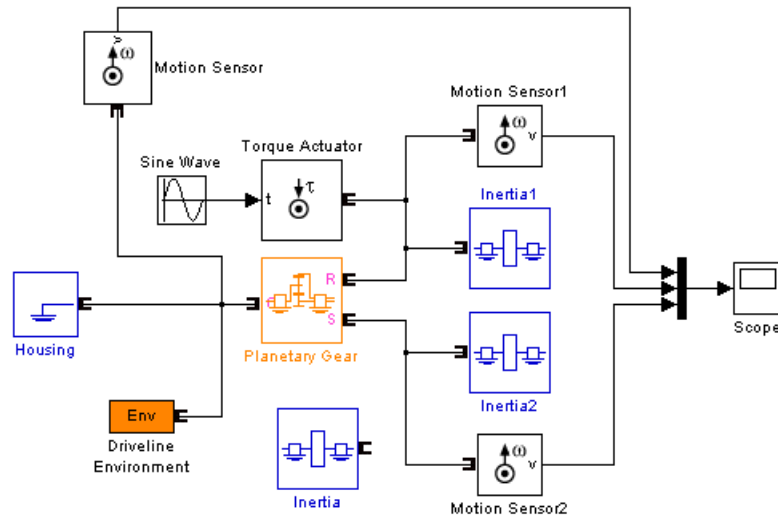
Simple Planetary Gear Model with Locked Carrier

6 Open the Scope and start your model. Observe the angular velocities of the ring, carrier, and sun.
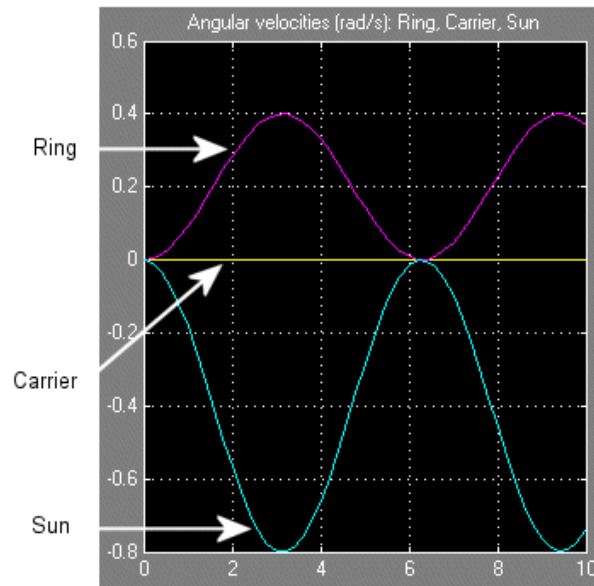
The carrier, connected to Housing, does not move. The ring is driven with a sinusoidal torque, and the sun responds by spinning in the opposite direction (ring and sun gear wheels are external to one another) at twice the rate. The ring wheel has twice the radius (or twice the number of teeth) as the sun, so it spins half as fast.

To learn more about modeling planetary gears with SimDriveline, see the Planetary Gear block reference page.

Controlling Gear Couplings with Clutches

The most important requirement of a practical drivetrain is the ability to transfer rotational motion and torque among spinning components at different speeds and gear ratios. A single set of gears is usually not sufficient to accomplish this. Clutches are the critical components that allow the drivetrain to selectively transfer motion and torque at different gear ratios under manual or automatic control.

This section explains how to model and use clutches in driveline models.

- "Engaging and Disengaging Gears with Clutches" on page 2-22 explains the essential purpose of a clutch and how to model a simple clutch that couples two shafts with a gear.

- "Modeling Realistic Clutch Systems with Loss" on page 2-27 adds greater realism with nonclutch frictional losses.

- "Braking Motion with Clutches" on page 2-30 shows how to gradually halt spinning motion with a clutch.

Engaging and Disengaging Gears with Clutches

A common problem in drivetrain design is transferring motion and torque at different fixed gear ratios. Drivetrains are typically designed to switch among a set of discrete 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 gradually engage and disengage driveline shafts from one another. The Controllable Friction Clutch block of SimDriveline represents a standard surface friction-based clutch that models this behavior.

You can also model gradual motion-torque transfer with the Torque Converter block, which models fluid viscosity instead of surface friction.

How a Clutch Works

A clutch makes two shafts spinning at different rates spin at a single rate by applying forces that tend to accelerate one shaft and decelerate the other. The most common way for a clutch to accomplish this is with surface friction. Such a clutch can operate in one of three 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, a moderate amount of contact between two surfaces is induced by applying clutch pressure (a force normal to the surfaces). The two surfaces in contact and moving relative to one another experience kinetic friction, which causes them to narrow their relative velocity. The faster surface tends to slow down (unless an external torque is acting) and the slower one to speed up. 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 locking of the shafts is controlled by static friction, which holds the shafts together as long as sufficient normal force is applied and no relative torque is large enough to overcome the locking. If the clutch unlocks but is still engaged, it again applies kinetic rather than static friction.

The transition between the unlocked and locked modes is a discontinuous change of motion. Modeling a clutch locking or unlocking requires searching for the correct combination of pressure and torque acting on the clutch. SimDriveline determines the instant of locking and unlocking during a simulation by accurate zero-crossing detection and repeated *mode iteration*.

Engaging and Disengaging a Gear with a Clutch

Here you construct a simple model that simulates a gear being engaged, then disengaged, by a clutch. Torque and motion are transferred from one shaft to another over a finite time interval. Start with the simple gear model of the last section or with the drive_sgear demo. The completed clutch model is the drive_sclutch demo.

1 From the SimDriveline block library, you need a Controllable Friction Clutch block. Also copy a Signal Builder and a Constant block from the Simulink block library.

2 Remove the Torque Sensor blocks, insert the Clutch between Inertia and Simple Gear, then reconnect the connection lines.

3 In the Clutch dialog, select the **Show mode signal port** check box, but leave the other defaults. Rearrange and connect the blocks as shown here.
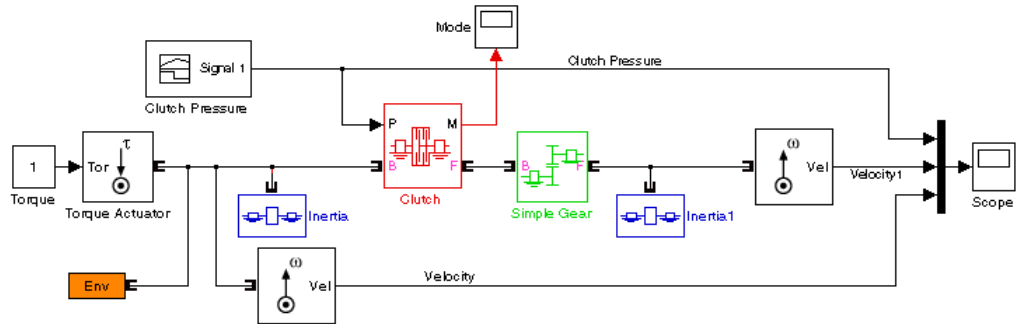
Simple Clutch Model with Programmed Clutch Pressure

4 Use the Constant block as the input torque signal in place of the sinusoidal signal. Reconfigure Mux and the first Scope blocks to accept three signals, the two angular velocities and the clutch pressure. Connect the second Scope to display the Clutch mode signal.

5 Open Signal Builder and construct the following signal. Signal Builder specifies the clutch pressure signal, which is normalized between 0 and 1. (The **Peak normal force** field in Clutch determines the maximum clutch pressure.)

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

6 Open the Scopes and start the simulation.

The normalized clutch pressure signal follows the profile you created in Signal Builder. From 0 to 2 seconds, the velocity of Inertia increases linearly because it is subject to a constant torque. At 2 seconds, the clutch begins to engage, and Inertia1 begins to spin. The velocity of Inertia continues to rise, although at a slower rate, because the two inertias now share the external torque. At 4 seconds, the pressure reaches its maximum, and the clutch locks. Inertia1 continues to speed up at a constant acceleration. At 6 seconds, the clutch begins to disengage as the pressure drops. Inertia and Inertia1 continue to accelerate with the applied torque. The clutch unlocks at 6.77 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 is still accelerating. But Inertia1, now free of the drive shaft and its torque, no longer accelerates and instead spins at a constant rate without frictional loss.
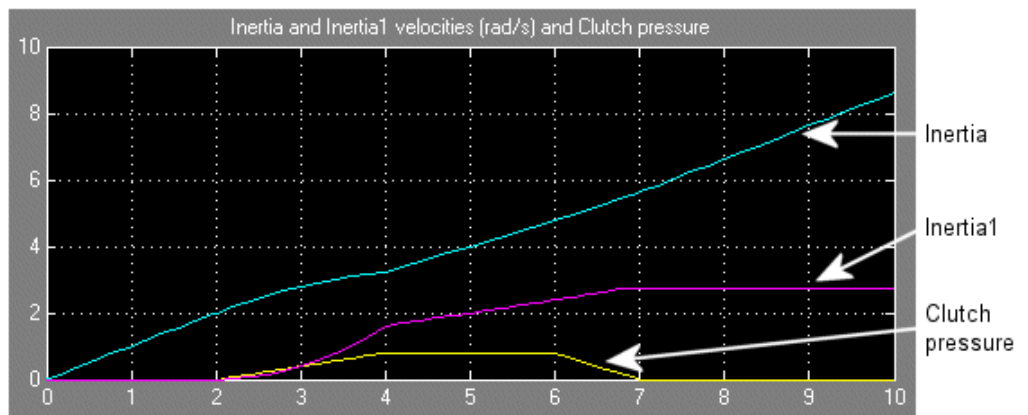
The Clutch mode signal indicates the relative motion of its two connected shafts. From 0 to 4 seconds, the two shafts are moving relative to one another. The follower (driven) shaft is slower than the base (drive) shaft, so the mode signal is -1. Once the two shafts lock, their relative velocity is 0, and the mode signal switches to 0. At 6.77 seconds, they unlock, and the drive (base) shaft starts accelerating faster than the driven (follower) shaft. The mode signal switches back to -1.
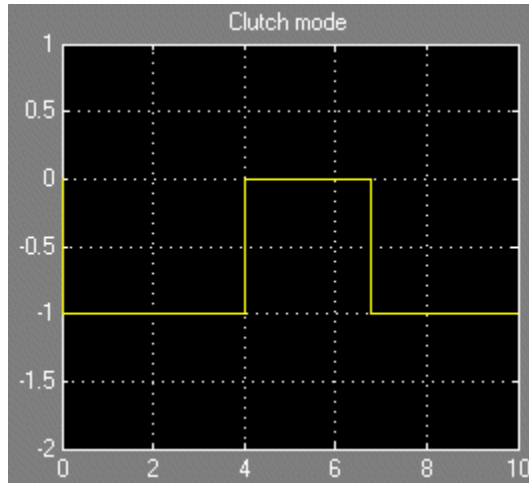
While the two shafts are locked, between 4 and 6.77 seconds, Inertia and Inertia1 spin together. But the Simple Gear, with a gear ratio of 2 between follower and base, transforms Inertia1's velocity to half that of Inertia. To see the two Inertias locked and spinning at the same rate,

1 Remove Simple Gear and connect Inertia1 directly to the Clutch. Change the **Peak normal force** value in Clutch to 2.5 (newtons).

2 Restart the simulation. Inertia and Inertia1 now spin at the same rate while the clutch is locked.
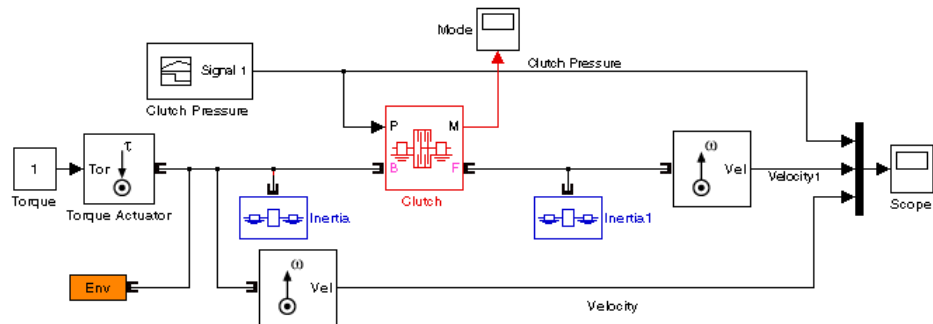
Simple Clutch Model with No Gear
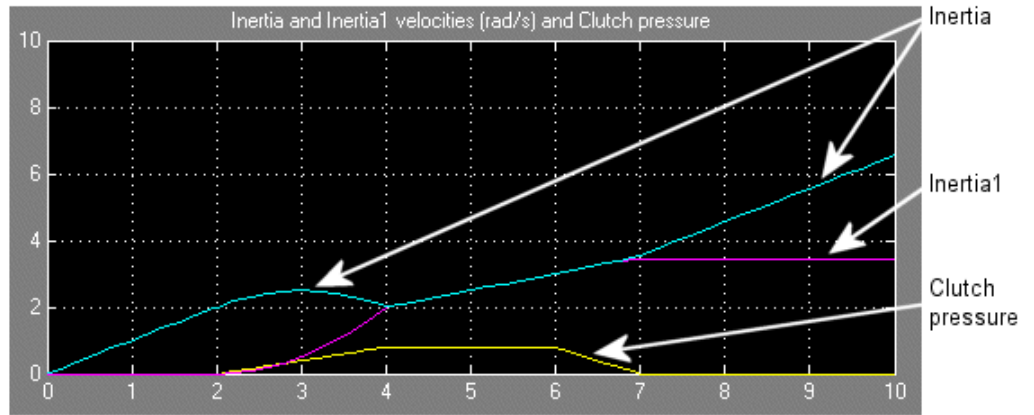
Modeling Realistic Clutch Systems with Loss

To make your clutch system model more realistic, you should add frictional damping to the spinning shafts. Here you add a kinetic friction torque proportional to the angular velocity to both sides of the clutch. A simple way to do this is to create a friction subsystem that applies such a torque to any driveline axis it is connected to. Then you can copy the subsystem and modify your existing clutch model by connecting the two copies on either side of the clutch.

Note The velocity used for this damping is the absolute velocity of a single shaft relative to rest (as defined by a Housing block, for example). If you had *two* driveline shafts and wanted to exert a relative damping between them as a function of their *relative* velocities, you could use the Torsional Spring-Damper block of SimDriveline. In general, this block applies a mixture of spring-like and damping torques between the two connected axes. But you can apply a pure damping torque by simply setting the spring constant to zero.

Creating a Torque Damping Subsystem

The frictional torque is $\tau_{fric} = -\mu\omega$, where μ is the frictional proportionality constant. To apply the frictional torque proportional to the velocity, you need to

1 Measure the angular velocity of the driveline axis

2 Multiply it by $-\mu$, because the frictional torque opposes the motion

3 Apply the resulting torque back to the driveline axis

To implement kinetic damping torque:

1 Copy Motion Sensor and Torque Actuator blocks and, from the Simulink library, a Gain block, into your model window.

2 Connect the angular velocity port Vel to the inport of the Gain block and the outport of the Gain block to the torque inport of the Torque Actuator block. Enter -0.3 for the **Gain** value in the Gain dialog, leaving the other defaults.

3 With your cursor, select the connected Sensor-Gain-Actuator block set, and create a subsystem. Call it Damper. When you create the subsystem, the port appearing on its block is a driveline connector port ⊏, not a Simulink port >.

Now create a second copy of Damper.
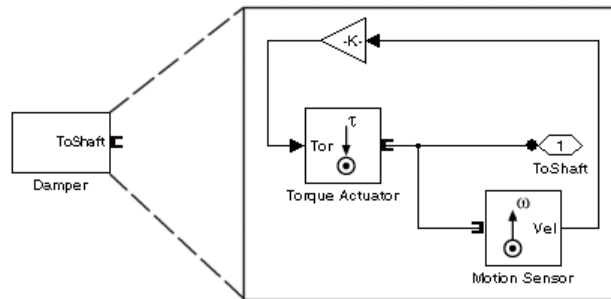
Rotational Kinetic Damping Subsystem

Connecting and Simulating the Damped Clutch System

Complete and run the model.

1 Connect the two Damper subsystems to the driveline of your previous clutch model as shown.
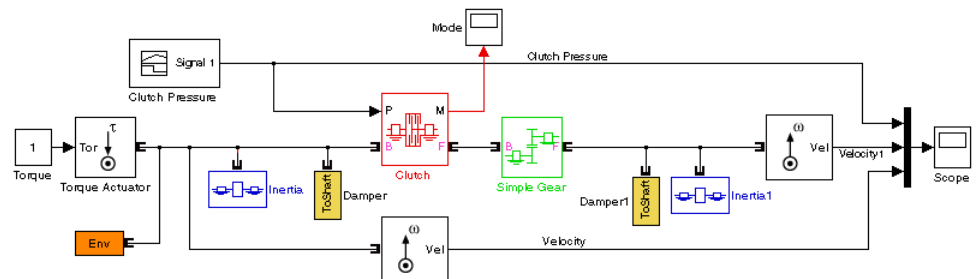
Damped Simple Clutch Model

2 Change the simulation time to 20 seconds. Then open the Scope blocks and click **Start**.

Readjust the horizontal axes of the Scopes with **Autoscale** to see the full plots. The clutch pressure and external torques are applied as before. But the shaft rotations are different now because of the damping.

Inertia   Clutch   Inertia1
           pressure
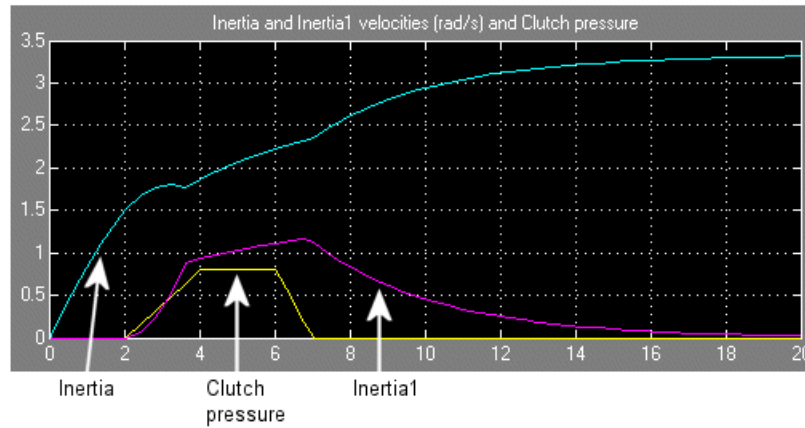
Inertia1, as before, begins to spin when the clutch starts to engage at 2 seconds. After the clutch locks at 4 seconds, the body continues to accelerate, but at a slower rate than it did without damping. At 6 seconds, the clutch begins to disengage and completely disengages at 7 seconds. Unlike the friction-free case, Inertia1, subject to friction, now starts to slow down. Its angular velocity drops exponentially with time once the external torque is removed.

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

Braking Motion with 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 fixed housing, which effectively has infinite inertia and is represented in SimDriveline by a Housing block. Because the housing cannot move, a

driveline axis locked to a housing also cannot move. You can implement the gradual engagement or disengagement of a driveline component with a housing using a clutch, just as you use a clutch to gradually couple or uncouple two spinning shafts.

Braking with a Double-Clutch System

The drive_clutch_engage demo model is an elaboration on the preceding models of this chapter and 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).
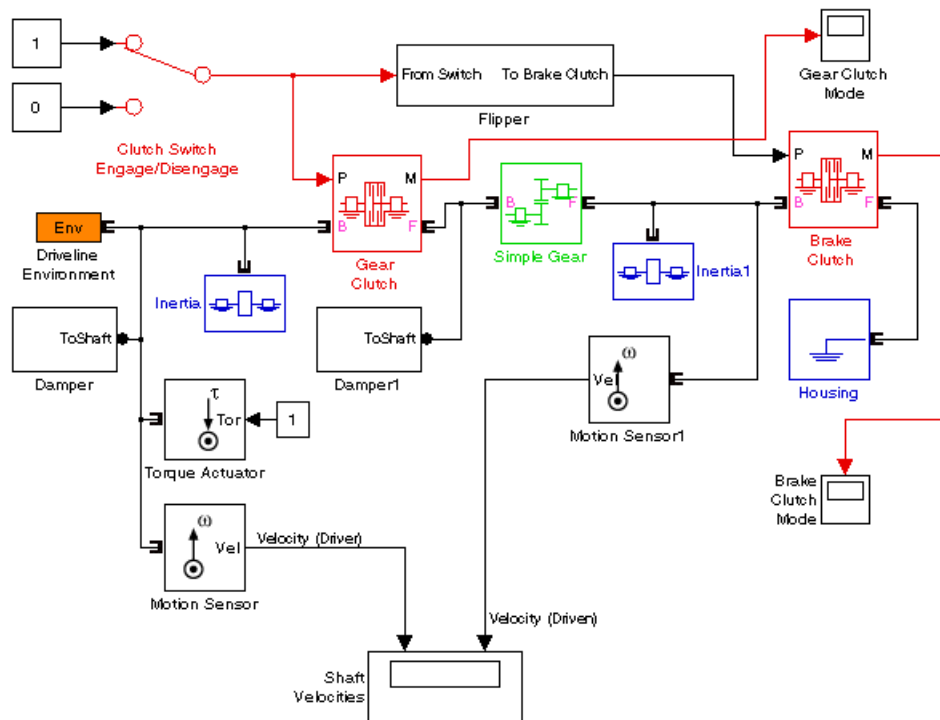
Simple Clutch Model with Brake Clutch

This model again uses the basic structure of inertia-clutch-gear-inertia. The first body, Inertia, is still driven by an external torque, and the initial velocities are still 0. There is, however, another clutch for the second body, Inertia1,

that can couple Inertia1 to the Housing and bring it to a stop. Another new feature, compared to the preceding models, is the switching assembly made of the Clutch Switch and Flipper blocks. You can flip this switch to apply a constant clutch pressure signal to either Gear Clutch or Brake Clutch. The two Damper subsystems are identical to those you constructed in "Modeling Realistic Clutch Systems with Loss" on page 2-27, except that the frictional constants, the **Gain** values of the Gain blocks, are set to -0.1.

1 Start the model with the Clutch Switch set to 1. The clutch pressure is then applied to Gear Clutch, which engages and locks the driver and driven shafts and causes Inertia and Inertia1 to rotate together.

The angular velocity of Inertia1 (2.5 radians/second) is half that of Inertia (5 radians/second) 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. The mode of Brake Clutch is then -1, because Brake Clutch's follower, the Housing block, is at rest, while the base, Inertia1, is spinning. The mode of Gear Clutch is 0, because its base and follower, the driver and driven shafts, are locked together.

After an initial transient, the system settles into a steady state of motion where the external torque is exactly balanced by the frictional losses. The effective frictional constant, with two dampers, is 0.2. With an external torque of 1 newton-meter, the terminal angular velocity of Inertia is then ω = 1/0.2 = 5 radians/second.

2 With the simulation running, now change the Clutch Switch to 0 to disengage Gear Clutch and engage Brake Clutch. The system undergoes another transient while Gear Clutch disengages and Brake Clutch engages.

The angular velocity of Inertia and the driver shaft settles down to a new steady state of 10 radians/second, twice its old speed. The mode of Gear Clutch is now -1, because the driven shaft (follower) is not moving, while the driver shaft (base) continues to spin.

Because Gear Clutch is now disengaged, Inertia is no longer subject to the second frictional damping block, Damper1. The effective frictional constant drops in half, to 0.1, and the terminal velocity doubles. At the same time, Inertia1 is no longer receiving torque through Gear Clutch. But Brake Clutch is engaged and couples Inertia1 to the immobile Housing. Once engaged, the kinetic friction of Brake Clutch and Damper1 bring

the driven shaft and Inertia1 to a stop. Because it locks, Brake Clutch's mode becomes 0.

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 another 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 will be 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 about 682 seconds of simulation time. Gear Clutch disengaged and Brake Clutch engaged, braking the driven shaft. The driver shaft's angular velocity rose from 5 to 10 radians/second. The driven shaft's angular velocity dropped to 0.
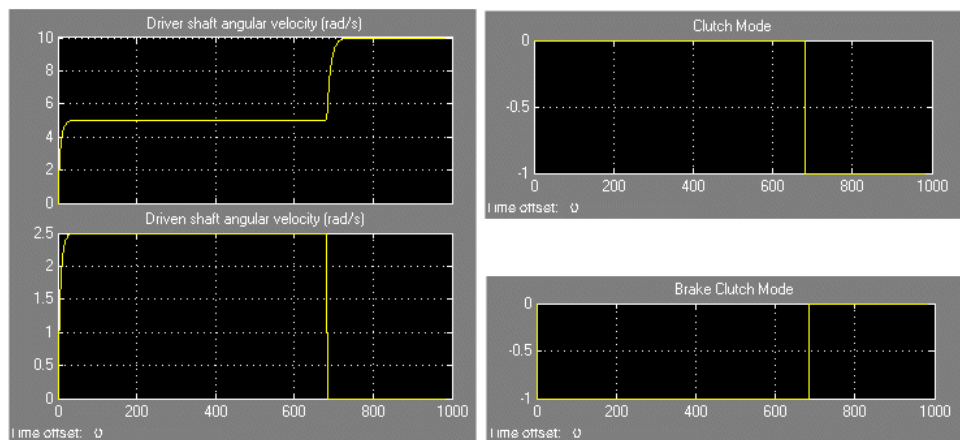
Modeling Transmissions

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. Realistic 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 specify the set of clutches to engage and disengage for each desired 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.

This section explains how to model transmissions, first by creating a transmission model from gears and clutches, then by using the SimDriveline library of predesigned transmission subsystems. One such predesigned transmission, the Simpson 4-speed transmission, is the basis of another example.

Note The examples in this section contain unrealistic clutch pressure signals that rise and fall sharply. A realistic transmission is controlled by clutch pressure signals that rise and fall smoothly. For a car demo with smoothed clutch pressure signals, see "Simulating a Complete Car" on page 2-49. "Improving Performance" on page 3-2 discusses the relationship of sharp and smooth clutch pressure signals to the Simulink solver choice and settings.

- "A Simple Two-Speed Transmission with Braking" on page 2-35
- "Introducing the Transmission Templates Library" on page 2-42
- "A Simpson 4-Speed Transmission Driveline with Braking" on page 2-44

A Simple Two-Speed Transmission with Braking

The demo model `drive_strans_ideal` contains a driveline system that makes up a simple but complete transmission.
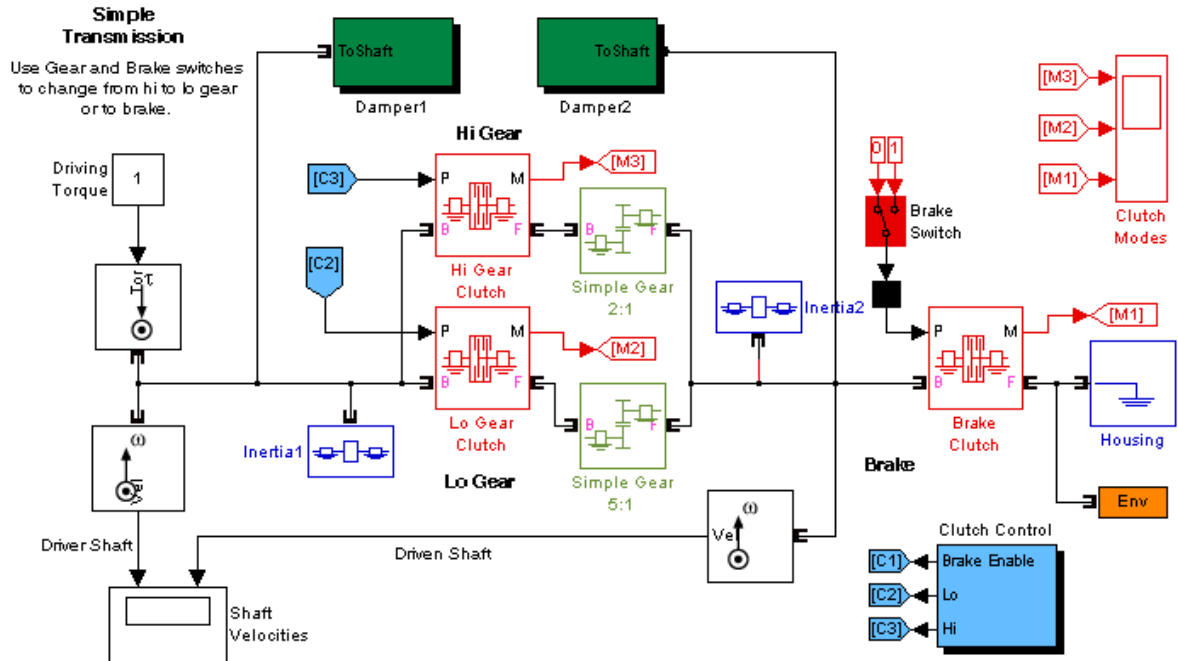
Simple Transmission with Two Gear-Clutch Pairs and Braking

The model is an elaboration of the `drive_clutch_engage` demo model presented in "Braking Motion with Clutches" on page 2-30. This model also contains two driveline shafts or axes, with an actuating torque applied to the driven shaft. Both the driver and the driven shafts are subject to, respectively, large and very small kinetic damping torques. (The kinetic torque constants μ are 0.1 and 10^{-4} newton-seconds/radian, respectively, in Damper1 and Damper2.) In the steady state, the driving and damping torques balance one another, and the two shafts spin at constant rates. (If braking is engaged, the driven shaft is stopped, as before.) But there are now two selectable gears to couple the two axes, instead of one.

This transmission model couples the gears in a simple way, with each gear and the brake associated with its own respective clutch. Coupling one gear requires engaging and locking its corresponding clutch, while ensuring that the other two clutches are disengaged. Switching on the brake requires disengaging the two gear clutches and locking the brake clutch.

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, and 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. The "low" and "high" labels, following common usage for automobile gears, refer, not to the gear ratios, but the angular velocity ratios.

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

- The "low" gear is the Simple Gear 5:1 block, which can be coupled by engaging its corresponding clutch, modeled by the Lo 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. Hence the name "low gear." Such a gear, by the same token, has a high torque transfer ratio of 5, from base to follower. In an automobile, a "low" gear like this is used to accelerate the vehicle from a stop by transferring a large torque down the drivetrain from the engine.

- The "high" gear is the Simple Gear 2:1 block, coupled by engaging its own clutch, represented by the Hi 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. Hence the name "high 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 that generated by the "low" gear.

While either Gear Clutch is engaged, the Brake Switch is disabled. You can start braking and bring the driven shaft to a stop by engaging Brake Clutch. This clutch, once locked, holds the driven axis fixed relative to the Housing. The driver shaft continues to spin, subject to the competing driving and

damping torques. In this transmission, the brake is completely disabled if either gear clutch is engaged. Disengaging the gears puts the transmission into "neutral" and allows you to use the Brake Switch to apply or not apply brake clutch pressure.

Clearly, 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 Housing. In a real transmission, such conflicts generate internal stresses and might destroy the machine. In a SimDriveline, such conflicts cause the simulation to stop with an error.

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   You must check every transmission's clutch schedule to implement the various transmission states correctly and to avoid motion conflicts among gear sets. You must also check clutch pressure signal profiles to make sure that any transmission's clutches are engaged, locked, unlocked, and disengaged in a realistic and conflict-free manner. In SimDriveline, unphysical or conflicting clutch schedules and clutch pressure signals lead to simulation errors.

Avoiding such conflicts leads, for the drive_strans_ideal model, to a unique clutch schedule.

Clutch Schedule for the Simple Two-Speed Transmission

Transmission State	Clutch1 State	Clutch2 State	Clutch3 State
Neutral/Braked	D/L	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 (or, in the case of the brake, enable) the clutch pressure signals to lock each clutch as needed.
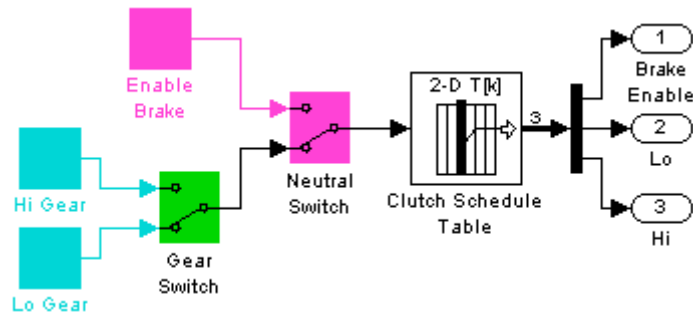
Clutch Control Subsystem for Simple Transmission Model

In this simplified and unrealistic clutch control model, the clutch pressure signals are just constants: 1 to engage and lock a clutch, and 0 to disengage it. (A clutch pressure signal is normalized to equal 1 when the surface friction force equals the peak normal force specified in the Controllable Friction Clutch dialog.) The brake signal is not a pressure, but only an enabling signal for the Brake Switch in the main model. The table of these constants, for each transmission state, is contained in the Clutch Schedule Table block, customized from the Simulink LookupND Direct block, which is discussed in the Using Simulink documentation. Open the dialog to see this table.
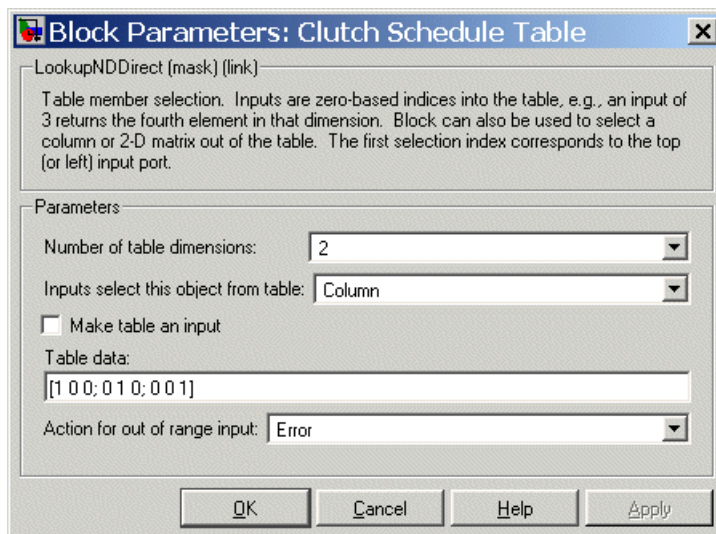
The table is indexed, starting from zero, in both row and column. An input signal of 0 causes the block to output the first column of table values; a value of 1 outputs the second column; and a value of 2, the third column. The first column applies zero pressure to the two gear clutches and enables the Brake Switch in the main model. Turning this switch on applies full pressure to the brake clutch. Turning it off releases the brake pressure. The second column applies full pressure to the low gear clutch and zero pressure to the high gear and the brake clutches. The third column applies full pressure to the high gear clutch and zero pressure to the other two.

These different table columns are activated by changing the positions of the two Manual Switch blocks, labeled Gear Switch and Neutral Switch. Putting the transmission into "neutral" and enabling the brake (upper position of Neutral Switch) feeds a zero signal to Clutch Schedule Table and activates the braking schedule. Switching the brake to off (lower position) allows the Gear Switch schedule signal to pass through instead. This signal has value 1 for the low gear and 2 for the high gear.

This clutch control subsystem is adequate for a simple model like this one, but not realistic. A full clutch control model requires realistic clutch pressure signals that rise from and fall back to zero in a smooth way. See "Shaping

Realistic Clutch Pressure Signals" on page 2-48 for more about modeling realistic clutch control pressures.

Running the Model, Switching Gears, and Braking

To see how gear switching works,

1 Start the model.

Its initial transmission state is low gear. The driven shaft spins at one-fifth the rate of the driver shaft.

2 Change the Gear Switch from Low to High, and observe how the driven shaft velocity increases in the Shaft Velocities scope.

The driven-to-driver ratio is now one-half. (The driver shaft velocity decreases slightly, because it experiences the damping torque on the driven shaft differently depending on which gear is engaged.)

3 Change the Gear Switch back to Low, then observe that the driven shaft again spins more slowly.

At the same time, while you switch the gears back and forth, the clutches, as shown in the Clutch Modes scope, switch from Lo Gear Clutch being locked, to the Hi Gear Clutch being locked, and back. When one is locked, the other is unlocked.

4 Now enable the brake by changing the Neutral Switch to the upper position.

The two gear clutches unlock and disengage. The driven shaft, subject to very light damping, now slows gradually. The Brake Clutch remains unengaged.

5 By turning the Brake Switch to on, you can switch the Brake Clutch to the locked mode and bring the driven shaft to an immediate and complete stop. The driver shaft continues to spin at 10 radians/second.

Running the Model Without Clutch Mode Iteration

In realistic transmissions, the pressure signal applied to one clutch is often determined by the locked/unlocked mode of another clutch. Simulating such a system requires SimDriveline to stop simulation time briefly and begin

mode iteration to search for a self-consistent state of all clutches across the entire driveline.

This transmission is simple and non-self-referential, insofar as each clutch is controlled by external signals only. No clutch is controlled by the mode of another clutch. In such a case, you do not need mode iteration for the clutches, because SimDriveline does not have to search for a collective self-consistent state of all clutches. The externally imposed clutch schedule does that automatically.

To turn off clutch mode iteration,

1 Open the model's Driveline Environment block (the block with the "Env" icon).

2 Select the **Disable mode iteration for clutch locking** check box.

3 Start the model again. The model runs faster and without mode iterations.

Disabling clutch mode iteration to avoid algebraic loops is sometimes necessary when you are using code generation-based simulation options in Simulink and Real-Time Workshop. See the `Controllable Friction Clutch` and `Driveline Environment` block reference pages for more details.

Adding Realistic Clutch Signals

The most critical addition you can make to the `drive_strans_ideal` model 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. A variant model, `drive_strans`, has smoothed clutch pressure signals. The price of this greater realism is a potentially more complex model. It is critical for Simulink to determine transmission motion for exactly two clutches to always remain locked, or for all four to be unlocked, at any instant. Changing the transmission's gear settings while maintaining this requirement is an example of the central problem of transmission design.

The final case study of the chapter, "Simulating a Complete Car" on page 2-49, implements smoothed clutch pressure signals. See "Shaping Clutch Pressure Signals" on page 2-59.

Introducing the Transmission Templates Library

The Transmission Templates library of SimDriveline provides examples of complete, working multiclutch transmission subsystems. The blocks in this library are unmasked. If you copy one into a model and double-click it, the subsystem opens directly, allowing you to inspect the component blocks.
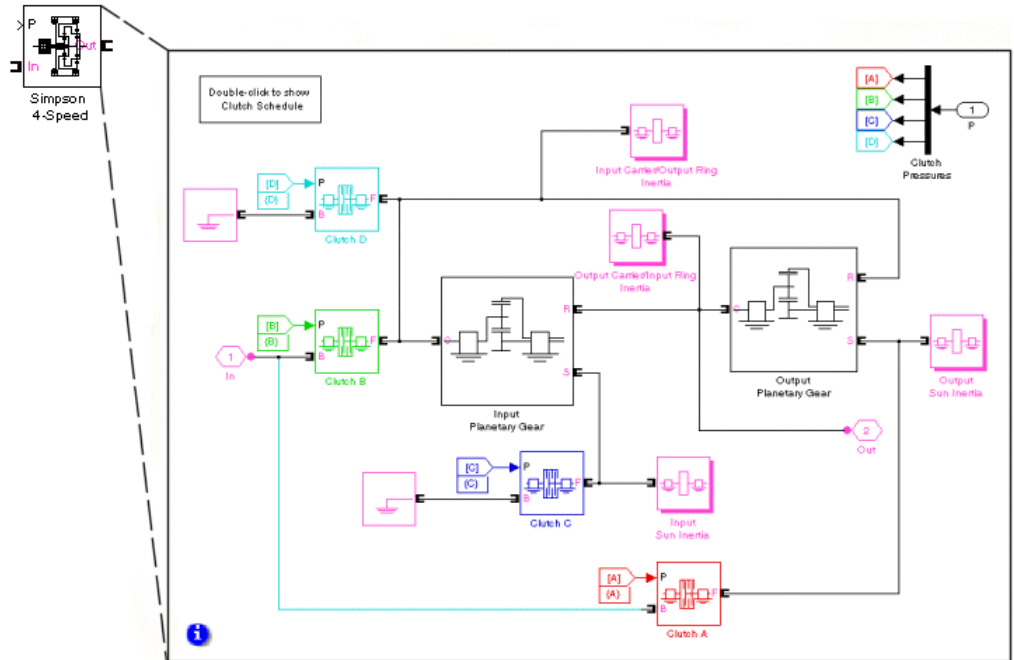
Simpson 4-Speed Transmission Template Subsystem

Note The Transmission blocks are not library-linked. Once you make a copy from the library to your model, you are free to modify your copy.

Each type of transmission block has its own clutch schedule, which you can view by opening the subsystem, then opening the clutch schedule block inside. (The corresponding block reference pages also list the clutch schedules for each Transmission block.) Properly engaging a transmission in a particular gear setting requires engaging a certain number of clutches, no more and

no fewer. Locking too few or too many clutches, or engaging the wrong clutches, will lead to conflicting gear meshings and simulation errors. You can disengage a transmission by setting all clutch pressure signals to 0.

Customizing and Using Transmission Blocks

Because you do not have to take any extra steps to unlink a transmission block from its library, you can easily modify the Transmission block copies in your models. You will typically need to change gear ratios, clutch pressures, and gear shaft inertias in any case. If you open the Transmission block to view the underlying subsystem, you can proceed to modify blocks at will.

Caution   Observe certain cautions when modifying the transmission subsystem component blocks:

- Do not remove any of the gear shaft inertia blocks or set their inertia values to 0. These inertias are needed for realistic simulation and preventing acceleration singularities when torques are applied.

- The clutch schedule for any transmission type specifies those clutches that must be engaged and those that must be free at any instant for the transmission to be properly in gear. Make sure that your clutch pressures respect this requirement. Set all clutch pressures to 0 only if you want to disengage the transmission completely (place it in neutral). Do not engage any more or fewer clutches than needed, at any time during simulation.

- If you want to redesign the transmission, by adding or removing gears, you must consider whether you need as well to add or remove clutches and redesign the clutch schedule. You also might need to add or remove gear shaft inertias.

The next section presents a driveline model based on the Simpson 4-speed transmission model of the Transmissions library.

A Simpson 4-Speed Transmission Driveline with Braking

The drive_simpson_ideal demo model builds on the previous clutch and transmission models with a more realistic transmission. (This is the same model presented in "Running a Demo Model" on page 1-4.) It uses the Simpson 4-Speed transmission block from the Transmissions library to transfer motion and torque from one shaft and inertia to another. The model is otherwise similar to drive_strans_ideal.
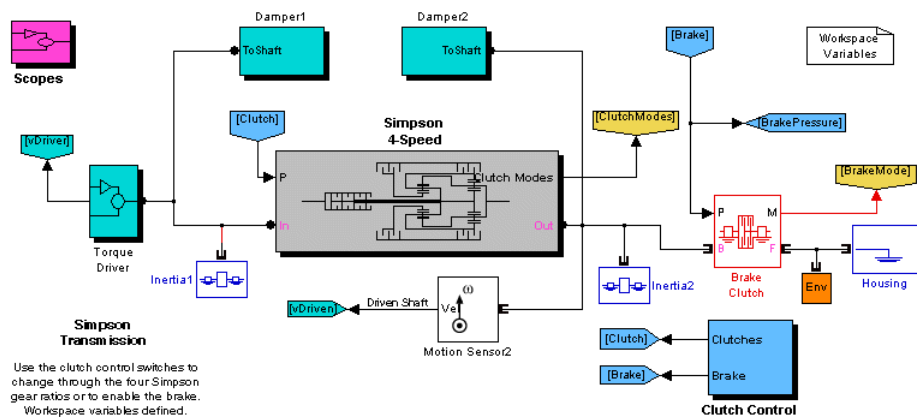
Simpson 4-Speed Transmission Model

A Torque Driver subsystem feeds a constant driving torque to the driver shaft (Inertia1). Two damping subsystems apply heavy and light kinetic friction to the driver and driven shafts, respectively. The three Scopes measure the shaft velocities, clutch pressures, and clutch modes, respectively. The model pre-load function defines essential parameters in the workspace. You can view these by opening the Workspace Variables block or opening the **Callbacks** tab of the **File** > **Model Properties** dialog. The Simpson 4-Speed transmission subsystem couples the driver to the driven shaft (Inertia2). A brake clutch and fixed housing allow you to brake the driven shaft if the transmission is disengaged. When you first open the model, the Clutch Control subsystem contains a set of programmed clutch signals for shifting the Simpson transmission through a preconfigured gear and braking sequence over 30 seconds.

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

Replacing Programmed with Controllable Clutch Pressures

To achieve manual control over the clutch pressures,

1 Change the simulation time in the Simulink model toolbar from 30 to inf.

2 Open the drive_simpson_clutch_control_switch model and convert the entire model to a replacement Clutch Control subsystem. (This model is not intended to be run by itself.)

3 Delete the original Clutch Control subsystem in drive_simpson_ideal and replace it with this new subsystem.

When you have completed these steps, you can run the model without stopping and manually switch the transmission into different gear settings.

Configuring the Simpson Transmission Subsystem

The Simpson 4-Speed transmission block used in drive_simpson_ideal has default settings for its component Gears, Clutches, and Inertias, with some exceptions. Certain parameters are changed for greater realism and are referenced to variables defined in the workspace when the model opens. These variables are used in the four Simpson Clutch blocks A, B, C, and D.

Simpson 4-Speed Transmission Clutch Variables

Workspace Variable	Meaning
num_fric_surf	Number of frictional surfaces in each clutch
eff_tor_rad	Effective torque radius in each clutch (m)
peak_normal	Peak normal force on clutch surfaces (N)
fric_coeff (matrix)	Kinetic friction coefficient as a function of the relative angular velocity of the clutch shafts

Within the Simpson transmission subsystem,

1 Open the Clutch Schedule block to see the table of gear settings, clutch lockings, and gear ratios. (The `Simpson 4-Speed` block reference page also discusses the clutch schedule.)

There are four distinct gear settings, each with a different effective gear ratio. For the transmission to be properly engaged and transmit torque and motion, exactly two clutches must be locked at any instant. Unlocking all the clutches simultaneously puts the transmission into neutral (no motion or torque transfer).

2 Close the transmission subsystem and return to the main model window. The main model's Damping subsystems use these variables for frictional damping of the driving (engine) and driven shafts coupled across the transmission.

Drive Shaft Damping Coefficients

Workspace Variable	Meaning
eng_damping	Driver (engine) shaft kinetic friction coefficient (N·m·s/rad)
driven_damping	Driven shaft kinetic friction coefficient (N·m·s/rad)

Programming the Clutch Schedule Logic

Note This and the following sections assume you have replaced the original programmed Clutch Control subsystem with the manually switchable replacement subsystem. See "Replacing Programmed with Controllable Clutch Pressures" on page 2-45.

From the main model, open the Clutch Control subsystem. The Clutch Schedule Logic block embodies the Simpson 4-Speed clutch schedule as a truth table. Each row represents a different gear setting. You select a particular row for output by inputting a set of 1's and 0's that specify the row value as a binary number.

Simpson 4-Speed Clutch Schedule Logic

Gear Setting	Truth Table Row	Truth Table Value
1	$00_2 = 0$	1 0 0 1
2	$01_2 = 1$	1 0 1 0
3	$10_2 = 2$	1 1 0 0
4	$11_2 = 3$	0 1 1 0

In the order of the Simpson Clutches — A, B, C, and D, respectively — the sequence of 1's and 0's indicates which clutches are locked (1) and which are free (0). These Boolean values are then converted into normalized clutch pressure signals.

Running the Simpson Transmission Model — Changing Gears

You are now ready to run the model.

1 Open the Scopes subsystem, then the individual Scope blocks. Close the Scopes subsystem.

With the Scopes, you can observe the angular velocities of the driving and driven shafts, and the pressures and modes of the four clutches.

2 Start the model.

Open the Clutch Control subsystem. (This should be the manually switchable subsystem.) You can change the gear settings by flipping the Bit0 and Bit1 Switch blocks and moving through truth table entries corresponding to each setting. (See the table, Simpson 4-Speed Clutch Schedule Logic on page 2-47.) Switching from one gear setting to another unlocks some clutches and locks others, but always leaves two clutches locked. As you flip between gear settings, the transmission transfers motion and torque at different ratios.

3 You can disengage the Simpson transmission completely by flipping on the Neutral Switch. This step also enables the Brake Switch in the main model. (If the transmission is engaged, not in neutral, the Brake Switch is disabled.)

If the transmission is disengaged but without braking, the driven shaft velocity slowly decreases under the influence of frictional damping. If you brake, however, by switching on the Brake Switch, the driven shaft velocity immediately drops to 0. The braking here works the same way as in the previous examples.

Shaping Realistic Clutch Pressure Signals

The `drive_simpson_ideal` model allows you to switch gear settings without placing the Simpson transmission in neutral. Of course, controlling a real manual transmission requires moving the transmission out of gear and into neutral, picking a new gear setting, then putting the transmission into the new gear. You can mimic these steps by flipping the Neutral Switch on, changing the gear setting, then slipping the Neutral Switch off.

The most critical addition you can make to this model 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. A variant model, `drive_simpson`, has smoothed clutch pressure signals. The price of this greater realism is a potentially more complex model. It is critical for Simulink to determine transmission motion for exactly two clutches to always remain locked, or for all four to be unlocked, at any instant. Changing the Simpson transmission's gear settings while maintaining this requirement is an example of the central problem of transmission design.

The following case study, "Simulating a Complete Car" on page 2-49, implements smoothed clutch pressure signals. See "Shaping Clutch Pressure Signals" on page 2-59.

Simulating a Complete Car

The full car drivetrain simulation of the `drive_full_car` demo encompasses all the points of this chapter and many of the features of SimDriveline. It includes engine and transmission models and a simplified 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 and neutral before a braking torque is applied. The clutch pressure signals are smooth and more realistic than the sharp clutch pressure signals used in the preceding studies.

Open the demo. The model pre-load function defines a set of workspace variables in MATLAB used by some of the blocks. Note the major systems of this car model.
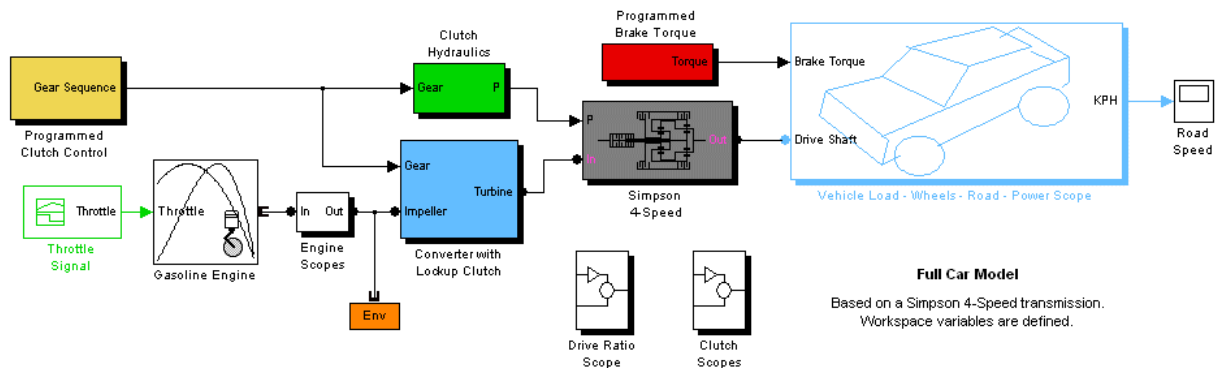
Full Car Model

The main driveline subsystems are

- Engine
- Torque converter
- Transmission

The large subsystem to the right represents the final part of the drivetrain: the vehicle inertia, the wheels, their coupling to the road, and braking. All the other subsystems in the model represent inputs that control the drivetrain or outputs that measure its behavior.

The following sections explain these subsystems in greater detail:

- "Modeling the Engine" on page 2-50
- "Modeling the Transmission" on page 2-52
- "Coupling the Engine to the Transmission" on page 2-52
- "Modeling the Wheel Assembly and Road Coupling" on page 2-53
- "Controlling the Clutches and Braking" on page 2-57

Finally, in "Running the Model" on page 2-60, you simulate the car and interpret the model simulation results.

Modeling the Engine

SimDriveline is primarily devoted to modeling the rotational dynamics of drivelines, accepting rotational power from any source that can be modeled in Simulink and converted to a connection line transferring torque. In most applications, your modeled driveline power and torque sources will represent engines and motors. For the purposes of system modeling, an engine or motor specifies an output torque as a function of driveline speed. However you specify the behavior of the engine or motor, in SimDriveline, the output is a connector port ⊏ transferring torque to the rest of the system.

Using an Engine Block from Vehicle Components

The Vehicle Components library of SimDriveline contains blocks representing simple engine models. You control these engine models with a Simulink throttle signal. The heart of the engine model is a function that specifies the maximum engine torque possible for each engine speed. The throttle signal controls how much torque, out of this maximum possible, the engine can deliver. The maximum possible torque itself is a function of the engine speed at any instant.

The `drive_full_car` demo uses a Gasoline Engine block from Vehicle Components. The block's properties specified in its dialog include the engine's maximum power, its speed at maximum power, and its maximum possible speed. The throttle signal is programmed by a Signal Builder block that specifies a time-depend throttle profile over the course of the simulation. Open these block dialogs to view these settings and the throttle profile. The

throttle signal is programmed to produce a realistic acceleration profile and to be consistent with the gear shifting sequence discussed in "Controlling the Clutches and Braking" on page 2-57.
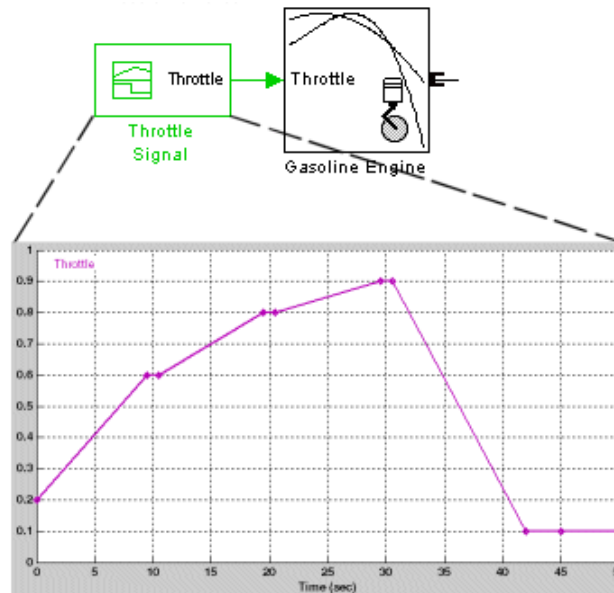
Engine Throttle Signal Profile

Learn more about the Engine block models from their block reference pages. See also "Vehicle Components" on page 2-6.

Alternative and Advanced Methods for Modeling Engines

The engine models of the Vehicle Components library are simple. You can create your own, more complex, engine models by elaborating on the basic pattern of engine speed determining engine torque output. The complete engine model involves a feedback loop because the output torque, once connected to the external load, determines how fast the output driveshaft spins. The engine model then uses this output speed to set the maximum possible torque.

Several important engine features to consider in a more complete model would be

- 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, or electromagnetic induction

- Additional controls beyond what can be represented by a single throttle signal

Modeling the Transmission

The Simpson 4-speed transmission subsystem in the `drive_full_car` model is similar to the previous example, "A Simpson 4-Speed Transmission Driveline with Braking" on page 2-44. The clutch and planetary gear properties are set in the block dialogs with workspace variables.

Workspace Variable	Meaning
inPlanetRatio	Gear: input planetary gear ring/sun ratio
outPlanetRatio	Gear: output planetary gear ring/sun ratio
numFricSurf	Clutch: number of surface friction surfaces
effTorqueRadius	Clutch: effective torque radius (m)
peakNormForce	Clutch: peak normal force on friction surfaces (N)
coeffFricTable (matrix)	Clutch: surface friction function (tabulated discrete function)
staticFricPeak	Clutch: static (locking) friction peak factor
velTol	Clutch: clutch velocity locking tolerance (rad/s)

For more about gears, clutches, and transmissions, see the `Controllable Friction Clutch` and `Simpson 4-Speed` block reference pages, as well as "Gears" on page 2-5 and "Transmission Templates" on page 2-5.

Coupling the Engine to the Transmission

The `drive_full_car` model couples the engine and the transmission through a torque converter subsystem.
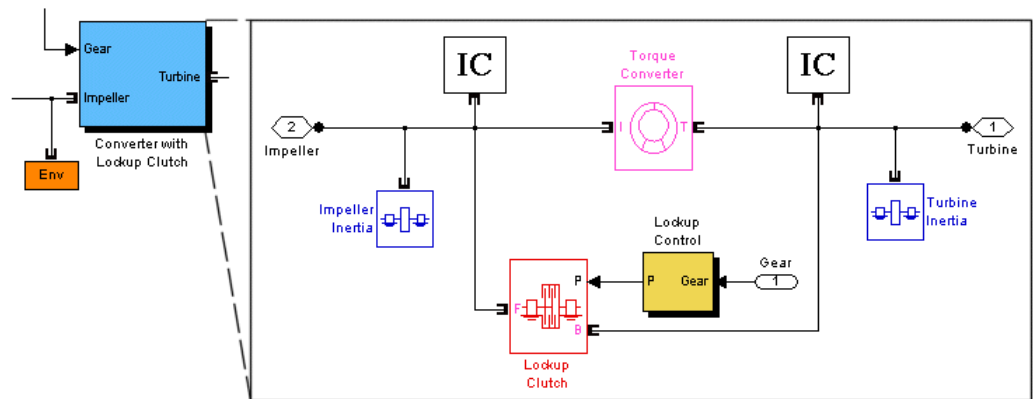
Torque Converter Subsystem

A torque converter, like a clutch, couples two independent driveline axes in such a way as to transfer angular motion and torque from an input to an output shaft. However, unlike a clutch, a torque converter never locks and the output shaft never exactly reaches the speed of the input. (The torque converter transfers motion by hydrodynamic viscosity, not by surface friction.) So 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 (IC) actuators start the input and output shafts at nonzero velocities.

The clutch in this subsystem is present to lock the input and output shafts together once the main clutches of the transmission have reached the highest gear. See "Controlling the Clutches and Braking" on page 2-57.

See the `Torque Converter` and `Initial Condition` block reference pages for more details about these blocks.

Modeling the Wheel Assembly and Road Coupling

The Simpson 4-speed transmission feeds its output torque to the final drive subsystem, Vehicle Load - Wheels - Road - Power Scope, which represents the vehicle inertia (the load on the transmission), the wheels, and the wheel contact with the road. This subsystem also incorporates a brake model that can impose, with the appropriate input signal, a brake torque on the wheels.

This torque acts on the driveline in addition to the reaction stress imposed by the wheel contact with the road. The wheel-road load model is implemented with Simulink alone.

This final drive subsystem is masked. Open it by right-clicking it, then selecting **Look Under Mask**.
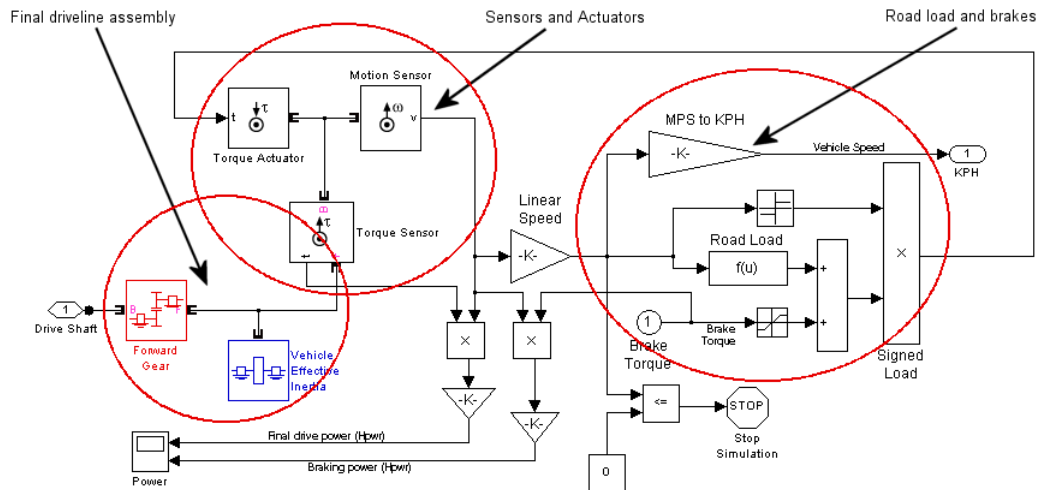
Final Drive Subsystem: Vehicle Load, Wheels, and Road Coupling

The subsystem has three major areas:

- On the leftmost part of the diagram is the terminus of the driveline proper, where the driveline connection lines end.

- At the upper left are a collection of sensor and actuator blocks coupling the driveline proper to the wheels and brakes.

- In the center and right are the Simulink blocks that model the wheel-road coupling and braking.

Modeling the Final Driveline Assembly and Vehicle Load

The driveline connection line sequence of the whole full car model ends with the first set of blocks on the left of the subsystem. The torque and motion are transferred forward through the Forward Gear and are loaded down with

the Vehicle Effective Inertia. Two workspace variables specify the relevant variables.

Workspace Variable	Meaning
ratioForwDrive	Gear: follower-to-base gear ratio in the final driveline stage
inertiaVehicle	Inertia: effective rotational inertia of the final driveline assembly (kg-m^2)

The final driveline connection emerges from the Forward Gear-Vehicle Effective Inertia assembly into a sensor-sensor-actuator set.

- The Torque Sensor measures the final torque along the driveline. This torque is used to calculate the power transferred along the drivetrain.

- The Motion Sensor measures the final angular motion of the driveline. This angular motion is used to calculate the driveline and braking power and to model the wheel-road contact.

- The Torque Actuator feeds the torques from the road load and the braking back to the driveline.

The road load and braking are discussed in "Modeling the Road Load — Adding Brakes" on page 2-55 and "Measuring the Driveline and Braking Power" on page 2-56.

Modeling the Road Load — Adding Brakes

The road load-braking part of the subsystem is modeled in Simulink only. It requires the final drivetrain angular velocity to compute the road load, which is the reaction torque of the road back on the driveline via the wheel-tire assembly. This road load torque is a function of the vehicle's linear speed implemented by the Road Load block. (This function is controlled by a set of workspace variables. Open the Road Load block dialog to view the functional form.) The Linear Speed (Gain) block converts the driveline speed to a linear velocity with an effective wheel radius, the workspace variable radiusWheel.

The braking torque is provided by the Brake Torque input signal. The total torque is computed from the road load torque and the brake torque. The

overall sign of the total torque is set by the sign of the linear vehicle speed. This total torque is fed back as a reaction torque on the end of the driveline by the Torque Actuator to the left.

If the applied braking torque is large enough and acts for long enough, it brings the vehicle speed to zero. At that instant the Stop Simulation block ends the simulation, if it has not already reached the end of the simulation time specified in the model toolbar.

Measuring the Driveline and Braking Power

The power transferred along a driveline axis is the angular velocity ω of the driveshaft multiplied by the torque τ transferred along the shaft: $P = \omega \cdot \tau$.

The final drive subsystem computes the final drive power and braking power by multiplying the angular velocity by the final driveline torque (measured by the Torque Sensor) and by the brake torque, respectively, then converting the MKS result to horsepower units. The Power scope displays both horsepower values. Open the Power scope, and close the final drive subsystem.

Alternative Differential, Wheel, Road, and Braking Models

The `drive_full_car` model uses the purely Simulink part of the final drive subsystem to model the end of the driveline and its connection to the wheels and road.

In the Vehicle Components library, SimDriveline provides a set of specialized blocks to model the end of a driveline and the resulting motion of the wheels and vehicle. You could create an alternative model to the final drive subsystem of `drive_full_car` by using these specialized blocks. The model uses one of these blocks already, the Gasoline Engine, as discussed in "Modeling the Engine" on page 2-50. For wheel-vehicle modeling, you can use the `Tire` and `Longitudinal Vehicle Dynamics` blocks instead of creating such models in Simulink. The `drive_4wd_dynamics` and `drive_vehicle` demo models illustrate the use of these blocks.

The braking model of `drive_full_car` applies a brake torque directly to the final driveshaft with a Torque Actuator. You could build an alternative brake model around a clutch. See "Braking Motion with Clutches" on page 2-30 and "A Simple Two-Speed Transmission with Braking" on page 2-35.

Controlling the Clutches and Braking

Return to the main model window of drive_full_car. Like any engineering system, the full car model requires control signals. One of these signals controls the throttle, as explained in "Modeling the Engine" on page 2-50. The other signals control the clutches and the braking. "Running the Model" on page 2-60 presents the full interplay of these control signals and how they determine the simulation results.

Programming the Transmission and Lockup Clutches

The Programmed Clutch Control is the central subsystem for the clutch signals. Its master signal controls two clutch sets, the four clutches of the Simpson 4-Speed transmission and the one lockup clutch of the Converter subsystem.

1 Open the subsystem. Four Step blocks contribute to the total or master clutch signal, which steps from 0 up to 5, through the intermediate integers, each step at a 10-second interval, from 0 to 40 seconds of simulation time.
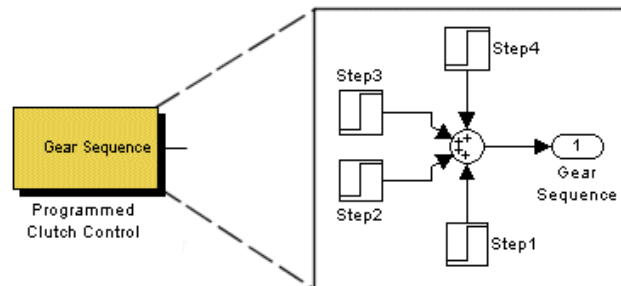
Master Clutch Control Signal

2 Close the Programmed Clutch Control subsystem.

3 Open the Clutch Hydraulics subsystem.
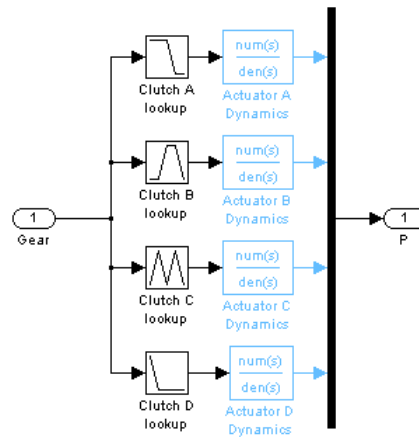
Clutch Hydraulics Subsystem

This subsystem, consisting of Simulink Lookup and Transfer Fcn blocks, converts the single clutch control signal to a set of four clutch signals that shift the Simpson 4-speed transmission through a fixed gear sequence: first gear, second gear, third gear, and fourth gear, at 0, 10, 20, and 30 seconds, respectively, of simulation time. Each gear is reset when the master clutch signal from Programmed Clutch Control steps up to the next higher integer, from 1 through 4. At 40 seconds, the master clutch signal reaches the value of 5. All the transmission clutches then unlock, and the Simpson transmission shifts into neutral. (See "Shaping Clutch Pressure Signals" on page 2-59 for more about the transfer function blocks.) You can view the clutch schedule for the Simpson 4-speed transmission by opening the Simpson 4-Speed subsystem, then double-clicking the Clutch Schedule block.

4 Close the Clutch Hydraulics subsystem.

5 Now open the Converter with Lockup Clutch subsystem, which was discussed in "Coupling the Engine to the Transmission" on page 2-52. This subsystem waits for the master clutch signal to reach the value of 5 before locking the two shafts and bypassing the motion transfer through the Torque Converter. Before 40 seconds of simulation time (before the transmission shifts into neutral), the driveline motion is transferred through the Torque Converter. The Lockup Clutch is triggered to lock by

the Lockup Control subsystem. Like the transmission clutch pressures, the clutch pressure signal is reshaped by a transfer function.
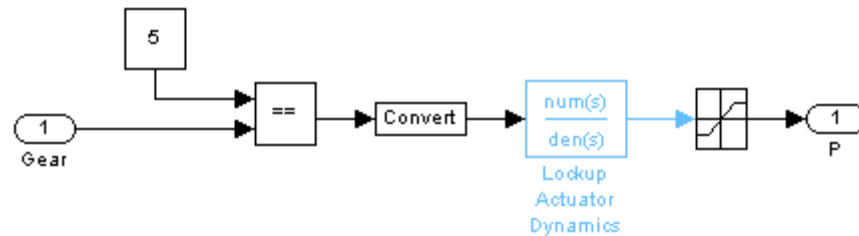
Lockup Control Subsystem

This behavior mimics a common drivetrain behavior. When the engine speed reaches a critical value or the transmission reaches the highest gear and the driveline coasts, a lockup clutch closes and forces the engine and the transmission input shafts to spin at the same rate. (A torque converter requires the two shafts to have different velocities to transfer any torque.) You could redesign the lockup control logic by instead implementing a conditional locking of the clutch, testing for a critical value of the engine shaft angular velocity.

6 Close all the torque converter-related subsystems.

Shaping Clutch Pressure Signals

Both the Clutch Hydraulics subsystem and the Lockup Control subsystem filter the clutch signals through Simulink Transfer Fcn blocks in order to shape the pressure signals from sharp steps to smooth rises or falls.

The characteristic rise/fall time of the transfer functions in the Clutch Hydraulics subsystem is set by the workspace variable `clutchRise`, with units of seconds. If s_0 = 1/clutchRise, the transfer functions have the form $s_0/(s+s_0)$.

You can determine the rise/fall time of the lockup clutch by inspecting its Transfer Fcn block.

Applying the Brake Torque

A separate control subsystem, Programmed Brake Torque, designed around a Signal Builder block, generates the Brake Torque signal fed into the final drive subsystem, Vehicle Load - Wheels - Road - Power Scope. Open the subsystem.
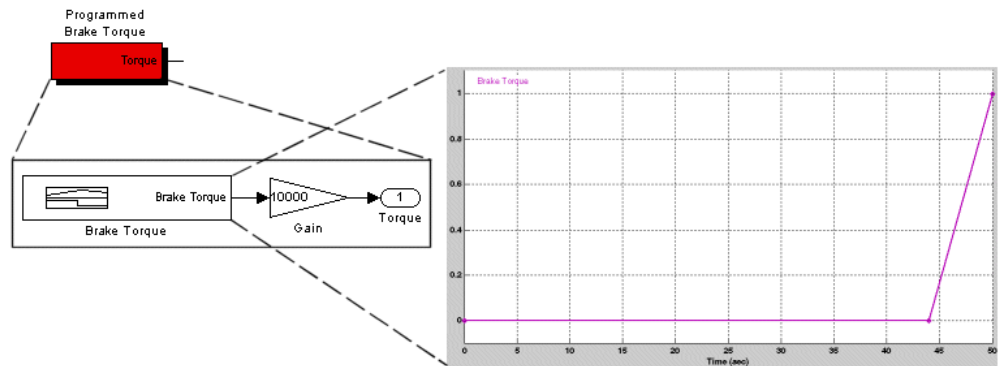
Brake Torque Signal Profile

The brake torque signal is designed not to be applied until after the Simpson 4-speed transmission is set into neutral and the lockup clutch is triggered. The signal is 0 N-m until 44 seconds of simulation time. It then rises to a large value at the end of the simulation at 50 seconds. The brake torque is strong enough to stop the driveline completely before 50 seconds. Close the Programmed Brake Torque subsystem and related windows.

Running the Model

Now simulate the car.

1 After closing all subsystems, open the Road Speed scope.

2 Also open the Engine Scopes, Drive Ratio Scope, and Clutch Scopes subsystems. Then open the Engine RPM & Torque, Drive Ratio, Clutch Modes, and Clutch Slippages scopes. Close the subsystems.

3 Review the simulation sequence before starting the model. The Brake Torque signal stops the vehicle before the nominal 50 seconds of simulation time.

Time Range (s)	Simpson Gear Setting	Brake Torque (N-m)
0 – 10	1	0
10 – 20	2	0
20 – 30	3	0
30 – 40	4	0
40 – 44	Neutral	0
44 – 50	Neutral	0 – 10,000

4 Start the simulation, then review the scope outputs.

Clutch Modes and Slippages

The Clutch Modes scope shows the sequence of clutch locking and unlocking that moves the transmission through its gear stages, first through fourth gear. The Clutch Slippages scope displays how much each clutch slips (relative angular velocity between the follower and base shafts in each clutch) when it is unlocked.
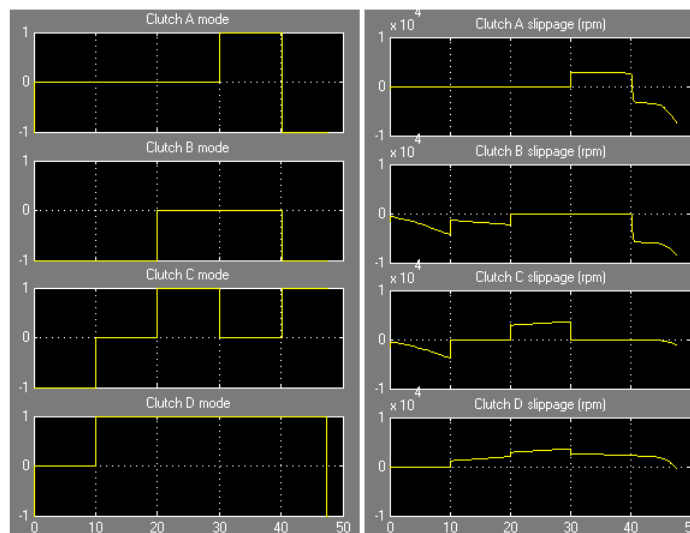
Engine Speed & Torque — Road Speed

The Engine RPM & Torque scope shows the engine speed in revolutions per minute (rpm), as well as the engine output torque, in newton-meters (N-m), delivered to the Converter subsystem. When the transmission shifts into neutral at 40 seconds, the engine speed jumps to its maximum and its output torque to zero.

The Road Speed scope displays the vehicle's linear velocity in kilometers per hour (km/h). Note that the braking stops the driveline and the vehicle before 50 seconds — to be exact, at 47.7 seconds.
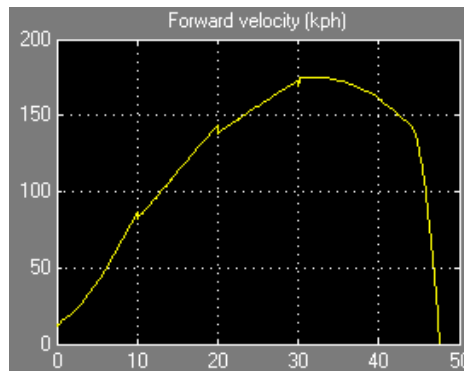
Drive Ratio

The Drive Ratio scope measures the effective gear ratio of the Simpson 4-speed transmission by computing the ratio of the input shaft to the output shaft angular velocities, respectively. (Open the Simpson 4-Speed subsystem to locate the Motion Sensor blocks.) As the transmission shifts through each gear from 1 to 4, its drive ratio goes down. (The speed ratio is the inverse of the drive ratio.) After 40 seconds, when the transmission shifts to neutral, the drive ratio is no longer well defined.
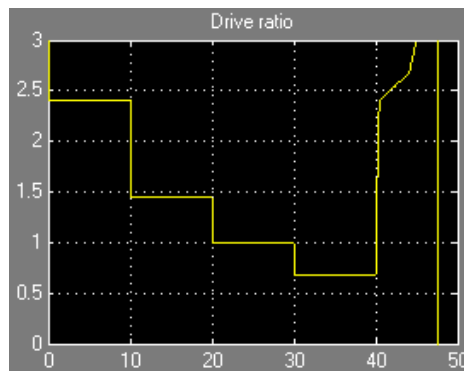
Driveline and Braking Power

Finally, the Power scope displays the total power exerted by the driveline, measured at the output of the final drive subsystem, and the braking power exerted by the brake torque necessary to bring the driveline to a stop. The braking power is negative, as the brake torque opposes the forward motion of the driveline.
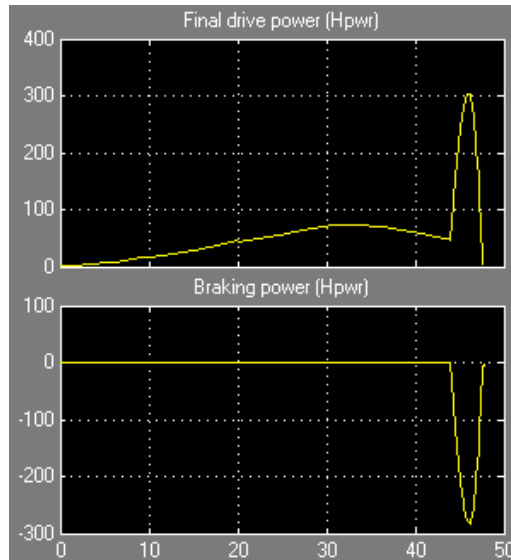
The braking power is zero until 44 seconds, when the brake torque begins to rise from zero. Both the total and the braking power then quickly peak and drop to zero as the braking stops the driveline motion in just under 4 seconds.

3

Advanced Methods

This chapter uses driveline examples to explore some of the deeper issues associated with driveline modeling, as well as powerful techniques that can extend your driveline simulations.

Improving Performance

SimDriveline uses the Simulink solver suite, and many of your choices for these solvers and their settings are no different for driveline simulations than they are for general Simulink models. See the Using Simulink documentation for a general discussion.

This section examines the difficulties specific to simulating driveline systems. The most important are the dynamical discontinuities caused by the locking and unlocking of clutches.

- "Increasing Accuracy and Speed" on page 3-2
- "Clutch-Shifting and Fixed-Step Solvers" on page 3-4
- "Troubleshooting Simulation Failures" on page 3-8

Increasing Accuracy and Speed

Once you have chosen a solver for your driveline system, your most important solver choices are *step size* and *tolerance*. Assume a fixed physical time, which the model simulates. The real time the model needs to finish is less if you demand less accuracy of the simulation. The real time needed is more if you demand more accuracy. The fundamental tradeoff in numerical simulation is speed versus accuracy. The step size and tolerance control this tradeoff. A special case of this tradeoff occurs with "stiff" systems.

Mode changes associated with clutches require further consideration if you use fixed-step solvers. See "Clutch-Shifting and Fixed-Step Solvers" on page 3-4.

Variable- Versus Fixed-Step Solvers

Except when you want to generate a code version of your model, you typically use variable-step solvers as the default. A variable-step solver automatically adjusts its step size as it moves forward in time to adapt to how well the solution is converging. You control the accuracy and speed of the variable-step solution by adjusting the solver tolerance. You can also limit the minimum and maximum step size in a variable-step solver, but you should change these settings only after trying other expedients.

Larger tolerances are associated with less simulation accuracy but, in general, greater speed. If a system undergoes sudden, rapid changes, larger tolerances can cause major inaccuracies. If your variable-step simulation is not accurate enough, or if it reaches the minimum step size allowed without converging, consider reducing the tolerances. The locking and unlocking of clutches automatically induce sudden changes in the driveline and might require you to reduce solver tolerances.

Fixed-step solvers are necessary if you are generating a code version of your model. A typical application in this case is a hardware-in-the-loop simulation. With a fixed-step solver, you can adjust the step size directly and thus control the accuracy and speed of your simulation.

Larger step sizes are associated with less simulation accuracy but, in general, greater speed. A fixed-step simulation with a large step size can miss sudden, rapid changes and lose accuracy. If your fixed-step simulation is not running accurately, try reducing the step size. Clutch mode changes automatically induce sudden changes in the driveline and might require you to reduce your simulation step size. You should set the fixed-step solver step size to the smallest value you can that produces an acceptable simulation speed (not too slow).

For a further discussion of how to simulate clutch mode changes accurately with fixed-step solvers, see "Clutch-Shifting and Fixed-Step Solvers" on page 3-4.

Solving Stiff Drivelines

A *stiff* system is one whose dynamics have several intrinsic time scales of very different magnitudes. These time scales are characteristic of the different but coupled degrees of freedom (DoFs) in the system or result from the contrast between initial conditions and later evolution. Moreover, these time scales might change during the simulation.

Generally, drivelines are not stiff systems, with one broad class of exceptions. While the internal dynamics of a driveline is typically not stiff, its coupling to its external load — the wheel-tire-road load, in the case of an automobile — is often stiff. (A tire is "stiff" in the colloquial sense of responding slowly to imposed forces. It also has a broad range of frequency responses.) For example, driving and road conditions typically change over seconds or tens of

seconds. But the internal changes of an automobile's drive system can change over fractions of a second, especially if clutch-shifting and braking are at work. In addition, clutch changes create dynamical discontinuities. "Clutch-Shifting and Fixed-Step Solvers" on page 3-4 discusses these dynamical discontinuities further.

Regular solvers can have difficulty accurately simulating such systems. The Simulink solver suite contains a set of special variable-step solvers designed to solve stiff systems. If you are simulating a stiff driveline with a variable-step solver and obtaining unsatisfactory results, consider

- Reducing the tolerances. This makes the simulation run more slowly.
- Using a special stiff variable-step solver.

If you are simulating a stiff driveline with a fixed-step solver and encountering unacceptable performance, try

- Reducing the step size. This makes the simulation run more slowly.
- Using the Simulink `ode14x` solver.

Reference
Moler, C. B., *Numerical Computing with MATLAB,* Philadelphia, Society for Industrial and Applied Mathematics, 2004, Chapter 7.

Clutch-Shifting and Fixed-Step Solvers
The discontinuities associated with the locking and unlocking of clutches can create significant obstacles to accurate simulation if you are using a fixed-step solver. See the `Controllable Friction Clutch` block reference page for details.

- Mode changes cause the number and nature of the degrees of freedom (DoFs) of the system to change during the simulation. See "Analyzing Degrees of Freedom" on page 3-11.
- Because clutch mode changes are simulated as idealized discrete events in SimDriveline, they cause the system's dynamics (torques) to change as well, as static and kinetic friction switch roles. The switch is triggered by

one or more clutch pressures rising high enough or falling low enough to induce locking or unlocking.

- In the default mode setting, SimDriveline uses mode iteration to determine when to lock and unlock clutches. Mode iteration consists of suspending the flow of simulation time and repeatedly testing the locking and unlocking conditions for all the clutches simultaneously. You can turn this mode iteration off.

- A fixed-step solver, unlike a variable-step solver, cannot adaptively reduce its step size to better resolve the system dynamics when clutches undergo mode changes. Instead, you need to make other changes to your model to compensate.

Smoothing and Offsetting Clutch Pressure Control Signals

You exert dynamic control on the locking and unlocking of clutches through their input clutch pressure signals. The simplest way to force a locking is to abruptly change a clutch pressure 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, as discussed in "Shaping Realistic Clutch Pressure Signals" on page 2-48. The best solution is to model a full clutch actuator. However, you can use simplified models instead of a complex clutch simulation.

The important criterion is to make sure the clutch pressure signal rises and falls smoothly and 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 demo models illustrate smoothed clutch pressure signals:

- `drive_sclutch`
- `drive_full_car`
- `drive_vehicle`

Staggering Multiple Clutch Signals. In multiclutch systems (transmissions) where multiple clutches lock and unlock together, you can further optimize your simulation by slightly staggering the clutch pressure signal changes, which ideally should be simultaneous. Slightly offset the rise or fall of each clutch pressure from the others in time, so that the clutches changing their modes together do not experience the dynamical discontinuities simultaneously.

Adjusting Clutch Parameters

You can adjust internal parameters within each `Controllable Friction Clutch` block to control when and how it locks and unlocks.

Changing the Pressure Threshold. The pressure signal coming into a clutch is dimensionless and normalized to 1 for a specified physical friction torque. You can also specify a pressure threshold 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.

Raising the threshold makes it harder for the clutch to engage. If you find a clutch in your simulation is engaging too easily, consider raising its pressure threshold. If the threshold is too high and the clutch has difficulty engaging, consider lowering the pressure threshold.

Changing the Velocity Tolerance. Each clutch has a velocity tolerance ω_{Tol} that controls when the clutch locks. SimDriveline determines whether to lock a clutch if the relative shaft velocity ω lies in the range $-\omega_{Tol} < \omega < \omega_{Tol}$ or whether to unlock a clutch if ω lies outside that range. In default operation, SimDriveline determines a value for ω_{Tol} based on the solver settings.

If the clutch is locking too easily, consider reducing the velocity tolerance. If the clutch has difficulty locking, consider increasing the velocity tolerance.

Adjusting Solvers for Discrete Mode Changes

The Simulink documentation discusses the adjustment of the Simulink solvers in general, while specific driveline solver issues are explored throughout the section, "Improving Performance" on page 3-2.

With locking and unlocking clutches in your driveline simulation, the most critical misadjustment of Simulink solvers to avoid is excessively loosening the accuracy of your solver. If the tolerances are too large in a variable-step solver, SimDriveline finds it difficult or impossible to accurately track the dynamical change associated with the change of friction torques acting on the driveline. If the step size is too large in a fixed-step solver, SimDriveline cannot accurately resolve abrupt changes such as clutch lockings and unlockings.

If you encounter convergence failures at or around the instant of clutch mode changes, consider reducing the solver tolerances for a variable-step solver and the step size for a fixed-step solver.

Controlling Mode Iteration

In the default mode, SimDriveline suspends simulation time flow and repeatedly tests if the clutches of your model should lock or unlock. This loop constitutes *mode iteration*.

You can turn off mode iteration through the Driveline Environment dialog. (Every driveline in your model must have exactly one Driveline Environment block attached.) The effect of turning mode iteration off is to spread the mode change calculation over multiple time steps, instead of concentrating it all at one time step. This change typically makes the simulation run somewhat faster but with some loss of accuracy.

With fixed-step solvers and in the generated code versions of SimDriveline models, mode iteration is automatically turned off. See the `Driveline Environment` and `Controllable Friction Clutch` reference pages for more details. See "Generating Code" on page 3-45 to learn more about generating code from SimDriveline models.

Reference

Higham, D. J., and Higham, N. J., *MATLAB Guide,* Philadelphia, Society for Industrial and Applied Mathematics, 2000, Chapter 12.

Troubleshooting Simulation Failures

You can encounter a variety of errors that cause model simulation to stop. Some of these errors arise from unphysical motions, actuations, and configurations of the driveline itself.

Overconstrained and Conflicting Degrees of Freedom

Analyzing and counting the driveline degrees of freedom (DoFs) are often essential to fixing simulation errors. For more about driveline DoFs, see "Analyzing Degrees of Freedom" on page 3-11.

To run successfully, your driveline simulation must, throughout the simulation, have a positive number of independent DoFs. Furthermore, the model's DoFs must not be in conflict with each other.

If you encounter a simulation error where the driveline cannot move, check to see whether the number of independent DoFs is positive. If N_{DoF} is not positive, consider

- Removing one or more constraining blocks, such as Gears, Clutches, or Housings

- Removing one or more Motion Actuator blocks

Try one or both of these steps repeatedly until you locate the origin(s) of the simulation failure and make N_{DoF} positive.

Consider also whether two or more DoFs are in conflict. For example, check whether two Motion Actuators are trying to move a single DoF in two different ways. Such a configuration creates a motion conflict and leads to a simulation error.

Clutch and Transmission Errors

Faulty clutch and transmission configurations generate many driveline motion failures.

To avoid or cure driveline DoF conflicts and errors, you should pay especially close attention to the collective mode 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. See "Clutch-Shifting and Fixed-Step Solvers" on page 3-4 and "Modeling Transmissions" on page 2-34.

It is easy to make these mistakes:

- Locking too many clutches simultaneously, leading to too many constraints. This leads to conflicting and/or overconstrained (not enough) DoFs.

- Locking conflicts among clutches, leading to redundant 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. Strictly speaking, this error does not overconstrain DoFs or put them in conflict. However, it typically puts the transmission in a state where it cannot transmit any torque or motion.

Inconsistent Initial Conditions

Like Motion Actuators, `Initial Condition` actuators can also cause motion conflicts. Unlike Motion Actuators, they do not remove DoFs from the driveline because they act only at the start of the simulation. However, if you configure them incorrectly, Initial Condition actuators can cause errors when you begin the simulation.

- Initial Condition actuators can conflict with one another.

  Example: Suppose 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 an initial velocity, and the follower cannot respond with half that velocity at the start of the simulation, the simulation stops with an error.

- Initial Condition actuators can conflict with Motion Actuators. When the simulation starts, the velocity signal controlling a Motion Actuator and the initial velocity value specified by the Initial Condition actuator must agree, if they act on the same DoF. 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 (e.g., branched lines, lines in closed loops) must have the same initial angular velocities.

Analyzing Degrees of Freedom

In a SimDriveline model, the mechanical motions are all rotational. Because absolute angles are not used in SimDriveline, it is simplest to identify a driveline *degree of freedom* (DoF) with an angular velocity. (SimDriveline sometimes uses the relative angle between two driveline shafts to determine the torques generated by internal driveline dynamic elements.) A DoF represents a single, distinct angular velocity. Each DoF responds to the torques acting on the inertias making up the driveline. Integrating Newton's equations of rotational motion determines the angular motions. In fundamental terms, mechanical DoFs are properties of rotating inertias. In SimDriveline, it is consistent and simpler to identify a single DoF as a driveline axis (idealized driveshaft) with its connected inertias, because the inertias are rigidly attached to their idealized shaft.

Thus, to identify and count DoFs in a driveline, you need to look at a SimDriveline diagram starting with its Physical Modeling driveline connection lines first, before considering its blocks. Driveline blocks modify the DoFs represented by connection lines by

- Imposing torques that act relatively between driveline axes

- Adding constraints among the driveline axes

- Imposing externally actuated torques and motions

Identifying rotational degrees of freedom is important for building and analyzing a driveline, particularly a complex machine with many constraints and external actuations. This section discusses how to identify driveline DoFs, take constraints into account, and extract the true or *independent* DoFs from a complete driveline diagram. It includes these interrelated topics:

- "Fundamental Degrees of Freedom" on page 3-12 discusses driveline connection lines.

- "Connected Degrees of Freedom" on page 3-15 discusses dynamic elements and internal torques.

- "Constrained Degrees of Freedom" on page 3-16 discusses how constraints remove DoFs.

- "Actuating, Sensing, and Terminating Degrees of Freedom" on page 3-20 discusses how Simulink signals import and export information into and from your driveline.

- "Counting Independent Degrees of Freedom" on page 3-22 puts the previous topics together and explains how to enumerate all the DoFs of a complete driveline.

- "Counting Degrees of Freedom in a Simple Driveline with a Clutch" on page 3-23 demonstrates the DoF counting rules by analyzing a SimDriveline model.

Relationship to Driveline States

Simulink represents driveline DoFs as *states*, among all states of a model, including the pure Simulink states. See "Finding and Using Driveline States" on page 3-31 for more about states and how they are related to DoFs.

Fundamental Degrees of Freedom

The basic unit of driveline motion is the degree of freedom (DoF) represented by an unbroken driveline connection line. Such lines represent idealized massless and perfectly rigid driveshafts. Rotating bodies with rotational inertias, represented by Inertia blocks, are rigidly attached to these lines and rotate with the axes.

Driveline Axes as Fundamental Degrees of Freedom

In SimDriveline, a driveline connection line anchored by driveline connector ports ⊏ represents an idealized driveline axis. The connection line enforces the constraint that the two connected driveline components rotate at the same angular velocity.

⊏─────⊐
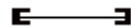
You measure the angular velocity of an axis with a `Motion Sensor` block. For its driveline analysis of a single axis, SimDriveline keeps track of only angular velocity. The absolute angle of an axis is internally undefined.
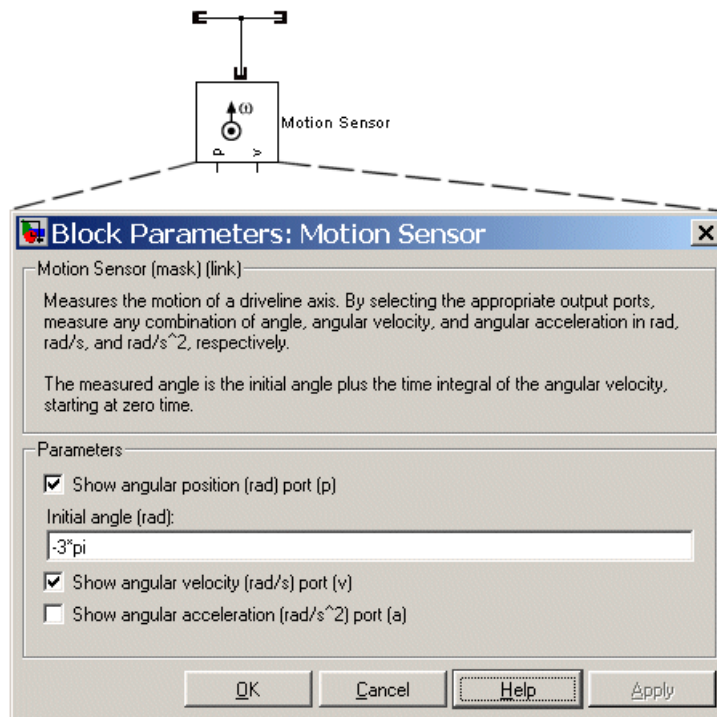
Measuring Driveline Axis Motion with a Motion Sensor

Defining Relative and Absolute Angles. Relative angle is sometimes necessary to compute internally generated torques between pairs of axes (see "Connected Degrees of Freedom" on page 3-15). To determine a relative angle, SimDriveline integrates the relative angular velocity of the pair of axes and adds the result to the initial relative angle that you specify in the cases where it is needed.

You can define an absolute angle of rotation for a single axis only when you measure its motion with a Motion Sensor block. The sensor defines the absolute angle by integrating the angular velocity of the axis and adding an arbitrary absolute reference angle that you provide in the Motion Sensor dialog.

Rigidly Rotating Inertias Attached to Driveline Axes

By itself, a driveline connection line represents a single DoF. You cannot subject this DoF to any torques, because it lacks rotational inertia. The other basic element needed to construct a functioning driveline model is one or more Inertia blocks. In a real mechanical system, the spinning bodies carry both inertia and DoFs. In SimDriveline, the spinning bodies are rigidly attached to a driveline axis. It is simpler to take the equivalent point of view that the driveline axis is the fundamental DoF and the bodies carry only inertia.

You attach Inertias to driveline connection lines by branching the lines. The attached inertias are subject to whatever torque is transmitted by the connection line, which imposes the constraint that everything attached to a single line must be spinning at the same rate.
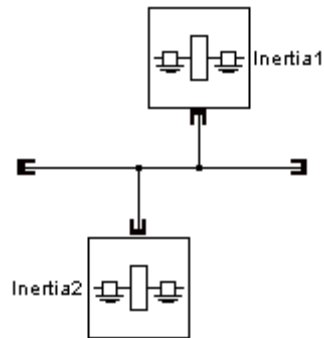
Driveline Axis Branching Rules and Constraints

You can branch driveline connection lines. You can only connect the end of any branch of a driveline connection line to a driveline connector port ⊏. All driveline components connected to the ends of a set of branched lines rotate at the same angular velocity. A set of unbroken, branched connection lines represents a single DoF.
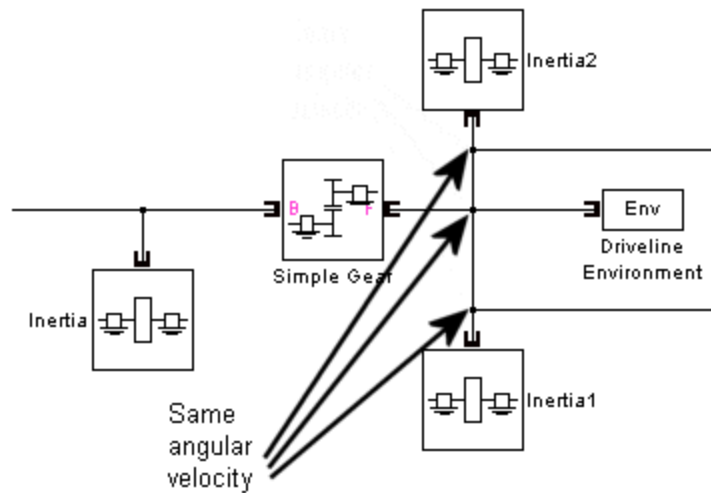
Branched Connection Lines and Angular Velocity Constraints

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 relative torque from the relative angle and/or motion of the two axes. This relative torque acts between the two axes, which remain independent DoFs and which transmit the relative torque (with equal magnitude and opposite sign on the base and follower axes) to their respective attached inertias.

Dynamic Elements and Internal Torque Generation

The Dynamics Elements library of SimDriveline contains blocks representing driveline elements that generate internal torques. (See "Modeling Dynamic Elements" on page 4-4.) A single relative torque is applied with positive sign to the follower (F) axis and negative sign to the base (B) axis.

Torsional
Spring-Damper
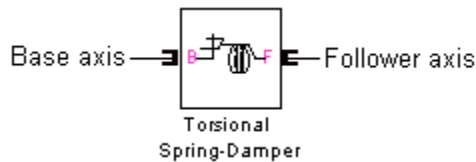
- `Hard Stop` and `Torsional Spring-Damper` generate spring-like and damping torques acting on the driveline axes connected to them. These torques are a function of the relative angle and angular velocity of the two axes.

- `Torque Converter` generates a viscous torque acting on the driveline axes connected to it. This torque is a function of the relative angular velocity of the two axes. In normal operation (forward power flow), the impeller (I) is equivalent to the base (B), and the turbine (T) to the follower (F).

Clutches and Conditional Connections

A clutch is a *conditional* or *dynamic* constraint.

A clutch, if unlocked, also connects two driveline axes and can impose a relative torque between them, leaving the two axes independent. The unlocked clutch is either completely unengaged, imposing no torque at all, or engaged, imposing kinetic friction as a function of the relative velocity of the two connected axes. The `Controllable Friction Clutch` block models such a clutch.

If a clutch locks, applying 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 "Constrained Degrees of Freedom" on page 3-16.

Constrained Degrees of Freedom

Certain driveline elements couple driveline axes in such a way as to eliminate 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 machine. 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 contained 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.

Most constraints are *unconditional* or *static*, that is, unchanging over the simulation. Clutches are the exception and constitute *dynamic* or *conditional* constraints.

Locking a Driveline Axis

Connecting a driveline connection line to a Housing block freezes the motion of the corresponding driveline axis. It cannot move, and its angular velocity is constrained to be zero during a simulation. Such an axis has no associated independent DoF.
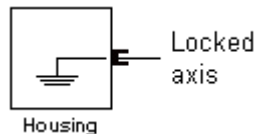
Locking Two Driveline Axes Together with a Clutch

A locked clutch, as long as the conditions for locking are valid, constrains the two connected driveline axes to spin together. The two axes remain distinct, but only one represents an independent DoF. The other is dependent. See `Controllable Friction Clutch` for more details.
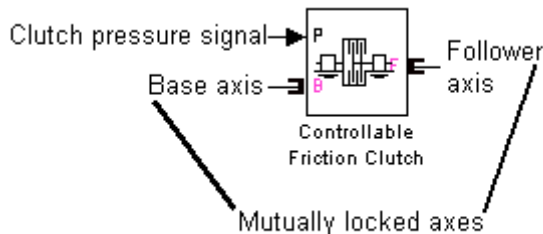
An unlocked clutch, even if it continues to apply a relative kinetic friction torque between the axes, no longer imposes a constraint. Instead, it acts as a dynamic element. See "Connected Degrees of Freedom" on page 3-15.

Coupling Driveline Axes with Gears

A gear coupling between two or more driveline axes reduces the independent DoFs of the machine by imposing constraints. The nature of those constraints depends on the gear being used. Gear blocks with two connected axes impose one such constraint and reduce the two axes to a single independent DoF.
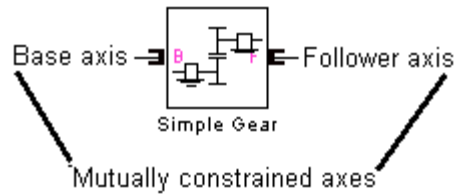
Multiaxis gears impose more than one constraint. For example, a planetary gear imposes two constraints on three axes, reducing the axes to one independent DoF. (This count does not include the fourth, internal DoF, the planetary wheel, which is not connected to an axis.)
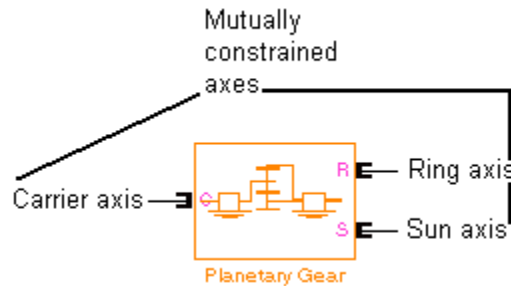
See "Modeling Gears" on page 4-3 in the block reference for more examples of gear constraints.

Closed Loops, Effective Constraints, and Constraint Consistency

The actual constraint count used to determine the number of DoFs is the number of effective or *independent* constraints. You must take special care in counting constraints in a driveline diagram when connection lines form closed loops. The presence of closed loops in a diagram reduces the effective constraint count by rendering some of the constraints redundant:

Number of independent constraints = Number of constraints from blocks – 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 clearly has two independent loops.
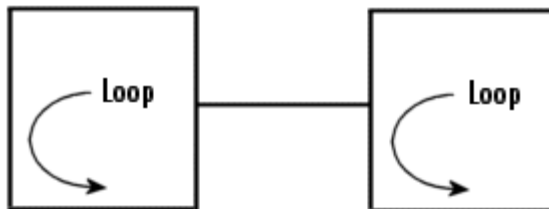
In this diagram, you can draw three loops: two inner loops, left and right, and the outer loop. The outer loop encompasses both inner loops.
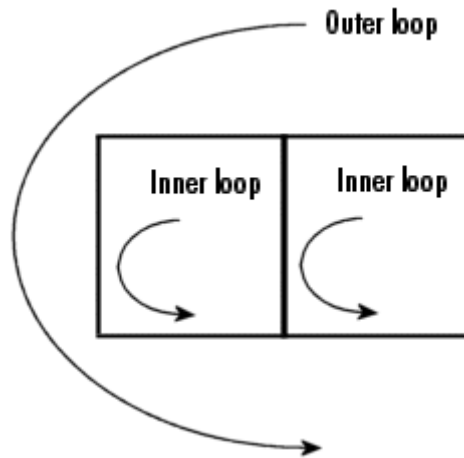
There are two independent loops in this diagram, because only two are fundamental. The outer loop is not fundamental.

Consistency of Constraints. A closed loop renders redundant one of the constraints contained within it as long as all the angular velocities constrained by line branchings are equal over the whole loop. (See "Driveline Axis Branching Rules and Constraints" on page 3-14.) The angular velocities not directly connected by lines must also be consistent if, for example, they are transferred through gears.

If the angular velocities along a closed loop cannot be made consistent, the driveline is overconstrained and cannot move.

Actuating, Sensing, and Terminating Degrees of Freedom

You can use SimDriveline blocks with only one driveline connector port ⊏ to originate and/or terminate degrees of freedom (DoFs) because they can end a driveline connection line. Such blocks include:

- Driveline Environment
- Inertia
- Housing

- Sensors and Actuators (see "Sensors & Actuators" on page 2-5), except `Torque Sensor`

- Vehicle Components (see "Vehicle Components" on page 2-6) such as Tire and Engines that use sensors and actuators in a subsystem

These blocks do not have to end a connection line.

They can instead be branched like this:

Terminating a connection line does not actually create or destroy a DoF, of course, but it does limit the DoF. If the termination is an actuator, the termination can modify the DoFs of the driveline. On the other hand, sensors have no effect on driveline DoFs.

Directionality, Actuating, and Sensing

Driveline connection lines have no inherent directionality. The direction of motion and torque flow is determined by the driveline dynamics once you simulate. You should contrast this with the inherent directionality of Simulink ports > and signal lines.

Although driveline connection lines are nondirectional, directionality is implicitly introduced into a driveline model when you attach actuator blocks to the diagram (see "Sensors & Actuators" on page 2-5), because these blocks interface pure SimDriveline blocks with the rest of Simulink. The actuator's effect on the driveline is determined by the (signed) Simulink input signal entering on one side.

The motion that results from the driveline simulation in turn determines the sign of the Simulink output signals that emerge from sensor blocks.

The Effect of Torque Actuation on Degrees of Freedom

Connecting a `Torque Actuator` to a driveline applies the torque specified by a Simulink input signal to the driveline. Such an actuation has no effect on the number of system DoFs. The driveline axes transmit the torque to their connected Inertias, and the driveline is free to respond to the imposed torques. The motion is simulated by integrating the driveline accelerations (a result of the imposed torques) to obtain the driveline velocities.

The Effect of Motion Actuation on Degrees of Freedom

Connecting a `Motion Actuator` to a driveline axis removes the freedom of that axis to respond to torques and instead specifies the axis motion during the simulation from the actuator's Simulink input signal. Motion actuation, unlike torque actuation, removes an independent DoF from the system.

Counting 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 (lumping together connected sets of branched lines) in the SimDriveline portion of your model diagram. Call the total of such lines N_{CL}.

These lines connect two driveline connector ports or terminate on one driveline connector port ⊏, as discussed in "Fundamental Degrees of Freedom" on page 3-12 and "Actuating, Sensing, and Terminating Degrees of Freedom" on page 3-20.

2 Count all the constraints arising from blocks that impose constraints on their connected driveline axes. Call the total of such constraints N_{bconstr}.

In most cases, each such block imposes one constraint, but complex gears impose more than one. See "Constrained Degrees of Freedom" on page 3-16 for details.

3 Count the number of independent loops N_{loop}. The effective number of constraints is $N_{\mathrm{constr}} = N_{\mathrm{bconstr}} - N_{\mathrm{loop}}$. Refer to "Closed Loops, Effective Constraints, and Constraint Consistency" on page 3-19 for more information.

4 Count all the motion actuations in your driveline, by counting each Motion Actuator block. Consult the preceding section, "Actuating, Sensing, and Terminating Degrees of Freedom" on page 3-20, for further discussion. Call the total of such motion actuations N_{mact}.

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} be positive.

Conditional Degrees of Freedom with Clutches

Unlike other driveline components, clutches can undergo a discrete mode change during the course of a simulation. The number of independent DoFs of a driveline is not, in general, constant during its motion. Each mode change of one or more clutches changes the independent DoF count. Different collective states of a driveline's clutches, taken as a whole, can have different total net DoFs. To understand a driveline completely, you must examine each possible collective state of its clutch modes to identify its independent DoFs and possibly invalid configurations.

See "Troubleshooting Simulation Failures" on page 3-8 and "Modeling Transmissions" on page 2-34.

Counting Degrees of Freedom in a Simple Driveline with a Clutch

Consider the simple transmission model `drive_strans_ideal`.
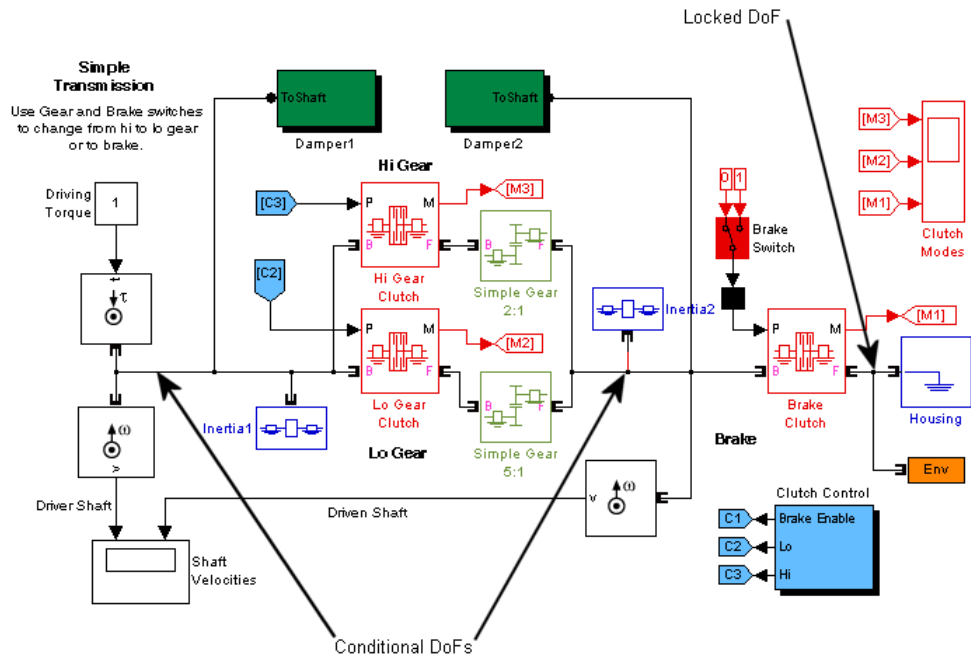
Simple Transmission

This system has five apparent DoFs, represented by these driveline axes:

- Branched axis with Inertia1

- Branched axis with Inertia2

- Axis connecting the Hi Gear Clutch to Simple Gear 2:1

- Axis connecting the Lo Gear Clutch to Simple Gear 5:1

- Axis connecting the Brake Clutch to the Housing

There is an apparent closed loop formed by the Gears and Gear Clutches. This loop is real only if both Gear Clutches are locked.

The actual number of independent DoFs depends on the state of the clutches. The model has no Motion Actuators, so we need consider only Gears and Clutches as constraints.

- The two Gears are always acting, thus yielding two ever-present constraints.
- The fifth axis is always connected to the Housing. These three constraints reduce five DoFs to two.

Now consider the clutches.

- Consider first the case where the Brake Clutch is disabled (free).
  - If both Hi Gear Clutch and Lo Gear Clutch are unlocked, the system has two independent DoFs, essentially one on the left of the Gear Clutches and the other between the Gear Clutches and the Brake Clutch.
  - If one of these Gear Clutches is locked, the additional constraint reduces the system to one independent DoF, essentially everything to the left of the Brake Clutch. (The clutch control schedule is set up to prevent both of these clutches from being locked at the same time.)
- If the Brake Clutch is enabled, then the clutch control schedule keeps the two Gear Clutches disabled.
  - If the Brake Clutch is unlocked, the driveline has two independent DoFs, the same two as above: essentially, to the left of the Gear Clutches and between the Gear Clutches and the Brake Clutch.
  - If the Brake Clutch is locked, the system is reduced to one DoF, essentially to the left of the Gear Clutches. Everything to the right of the Gear Clutches is locked to the Housing in this case.

This table and abstract diagram summarize the possibilities available in this model.

Brake Enabling	Clutch Locking	Independent DoFs
Brake disabled	Both Gear Clutches unlocked	Two: On the left and on the right of the Gear Clutches
	One Gear Clutch locked	One: On the left of the Brake Clutch

Brake Enabling	Clutch Locking	Independent DoFs
Brake enabled	Brake Clutch unlocked	Two: On the left and on the right of the Gear Clutches
	Brake Clutch locked	One: On the left of the Gear Clutches
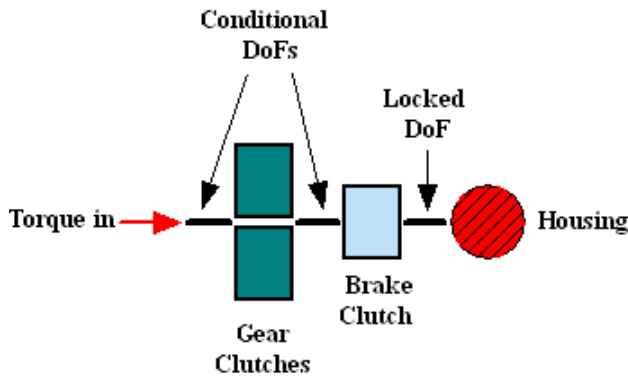
Degrees of Freedom in the Simple Transmission

Possible But Nonphysical Configurations

It is also worth considering possibilities not available in the model because the clutch schedule design, implemented in the Clutch Control subsystem, excludes them.

Both Gear Clutches Locked, Brake Clutch Unlocked. This configuration creates a conflict of DoFs and reduces the independent DoFs to one. The driveline axis to the right of the Gear Clutches 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 SimDriveline 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, Brake Clutch 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. In addition, the driveline axis between the Gear Clutches, driven by the driveline axis to the left, tries to spin but finds itself locked to the Housing. Attempting to simulate such a configuration leads to a SimDriveline error.

Both Gear Clutches Locked, Brake Clutch 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 Clutches tries to spin at two different nonzero rates, while at the same time remaining locked to the Housing, creating two distinct DoF conflicts.

Trimming and Linearizing Driveline Models

Note This section assumes some familiarity with advanced Simulink modeling techniques.

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. Trimming and linearization are the formal steps of such an analysis.

This section explains how to trim and linearize your driveline model and the related concepts of states and inverse dynamics.

The section concludes with three interrelated examples:

Trimming, Inverse Dynamics, and Linearization

Trimming a driveline means finding a state of motion that satisfies certain conditions on some combination of the degrees of freedom (DoFs) and their derivatives. Trimming usually starts with a guess. The solution to the trimming problem is the *trim* or *operating point*.

Linearizing a system means finding the response of the system to small perturbations in its motion. These perturbations can be small changes in initial conditions or in the applied torques. Simulink provides a basic command (among others) for linearization called linmod.

Driveline and Simulink States

Simulink represents driveline DoFs and other information about a model's dynamics with *states*. Trimming a driveline requires extracting its driveline states, which are part of the complete set of Simulink states of your model. Although the number of 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 angular velocities, not the angular velocities of particular driveline axes. This DoF-to-state transformation is not known beforehand, so you must trim your driveline without specifying which state is which.

Caution Simulink and other products based on Simulink and MATLAB contain trimming commands that require prior state specification; Simulink itself provides the `trim` command. Because SimDriveline allows counting, but not identification, of driveline states, such state-based trimming methods are difficult to use with driveline models.

State information is also useful for analyzing a driveline's *inverse dynamics*. Normally, you apply torques to a driveline and then determine the motions. Inverse dynamics means specifying motions to determine what torques are needed to produce those motions.

You can extract state and model output data from your simulation by selecting the appropriate check boxes in the **Data Import/Export** panel of your model's **Configuration Parameters** dialog. The default state and output vectors are `xout` and `yout`, respectively.

Relation Between Trimming and Linearization

Trimming and linearization are closely intertwined. Trimming a system leads to a trim or operating point. That configuration of the system is then often the system trajectory around which you linearize the motion. If, after linearization, the operating point proves to be unstable, you typically attempt another trim and locate a different trim point.

Role of Discrete Driveline Modes

The collective mode of the driveline is the set of all its clutch modes. If clutch mode changes are possible, trimming requires determining which clutches are locked and unlocked, as well as finding the state of continuous motion.

During linearization, SimDriveline starts with the clutch modes you specify and performs a clutch mode iteration to find a consistent state of all clutches. It then implements the perturbation of continuous states, holding the clutch modes fixed.

Inverse Dynamics and Trimming

A typical driveline simulation is based on torque-actuating the driveline axes, then measuring the resulting motions. This analysis is called *forward dynamics*. If you motion-actuate (instead of torque actuating) some parts of your driveline, those axes and the equivalent states are no longer independent. There is no point in measuring their angular velocities because you already specify them.

If you want outputs from these axes, measure instead the torques flowing along them. Knowing these torques is the starting point of *inverse dynamics* analysis.

Trimming involves a mixture of forward and inverse dynamics. You apply torques to some axes and specified motions to others. You have trimmed the model once you have determined a desired and consistent driveline motion state. Knowing its state in that trajectory, you can linearize the model's motion.

Relation of Trimming and Linearization to Control Design

A complete driveline system usually consists of the driveline part (the plant) and a controller that changes the plant state in response to human or automatic commands. In designing a controller for a driveline, it is essential to locate stable operating points and analyze the system's response at those operating points to small changes in control inputs. These are precisely trimming and linearization.

These products provide specialized and advanced methods that help you design controllers with MATLAB and Simulink:

- Control System Toolbox

- Simulink Control Design

In addition, Stateflow is a powerful tool for designing and analyzing transitions among discrete states such as those found in clutches and transmissions.

Finding and Using Driveline States

This section helps you locate and use states with SimDriveline. See the Using Simulink documentation for more about states in general.

See "Counting Driveline States in a Full Car" on page 3-35 for an example.

Relationship of States to Degrees of Freedom

Understanding degrees of freedom (DoFs) in your driveline model is essential for advanced analysis, simulation, and troubleshooting. (See "Analyzing Degrees of Freedom" on page 3-11.) Simulink represents these DoFs as part of the model's states.

The driveline states are a subset of the model's total states. You can count the number of driveline states in your model by counting the independent DoFs with all clutches unlocked. Both DoFs and states are based on the angular velocities of the associated axes. See "Analyzing Degrees of Freedom" on page 3-11 for a complete discussion of DoFs.

The independent DoFs are essentially the unbroken driveline axes or branched axis groups, as well as axes connected by true constraints like gears. Axes on either side of a dynamic element are independent. (For the purpose of identifying states, Simulink treats clutches as dynamic elements, not as constraints.)

When all clutches are unlocked, the driveline states are all independent. Once a clutch locks, one of the associated DoFs becomes dependent. The two associated states become dependent because the two angular velocities are necessarily equal.

Locating Driveline States in Simulink

Your driveline model consists of a mixture of SimDriveline and ordinary Simulink blocks. The model in general has Simulink states associated with the Simulink blocks. The driveline states of a single driveline system are associated with the Kinematic subsystem of the `Driveline Environment` block of that driveline. You can list all model states with the Simulink `model` command:

1 Open a model. In this example, use `drive_model`.

2 At the command line, enter

```
[sys,xO,str,ts] = drive_model([],[],[],'sizes');
sys
str
```

The relevant elements of `sys` are

- `sys(1)` = number of continuous states

- `sys(3)` = number of outputs

- `sys(4)` = number of inputs

`str` lists all model states, including the driveline states associated with each Driveline Environment block.

Trimming a Driveline with Inverse Dynamics

For a specific example of trimming, see "Trimming a Full Car to Rest" on page 3-40.

Consider trimming a model with a mixture of forward and inverse dynamics. In this example, the model is called `drive_model`. Open the model.

Inserting Sensors, Actuators, Scopes, Inports, and Outports

First set up driveline sensors and actuators and model inputs and outputs, as necessary and desired.

1 Attach Torque Actuators (or the equivalent, such as Engines) to whichever axes you want to torque-actuate. If you want to measure their resulting motions, add Motion Sensors as well.

2 Attach Motion Actuators to whichever axes you want to move according to prior specifications. If you want to measure their resulting torques, add Torque Sensors as well.

3 For whatever measurements you want to save to your workspace, feed the corresponding Motion and Torque Sensor output signals to Outport blocks.

4 For any external input signals you need to feed into the model, add Inport blocks.

If you want to visually examine sensor output, add and connect Scopes as desired.

Counting the States

Now count the states, inputs, and outputs by using the `'sizes'` option in the `model` command. See "Finding and Using Driveline States" on page 3-31.

From `sys`, read the number of states, outputs, and inputs. From `str`, read the subset of driveline states.

Configuring and Initializing the Model

Configure the model to extract state and output data.

1 From the **Simulation** menu, open the **Configuration Parameters** dialog. Select the **Data Import/Export** node.

2 In that panel, under **Save to Workspace**, select the **States** and **Output** check boxes. The default workspace variables for this data are `xout` and `yout`, respectively.

3 Close the dialog.

Check the model's blocks to impose the initial conditions as you want them. If you are feeding input and/or initial state data into the model, check those values as well.

Finding a Trim Point

You can now run your model as needed. Each time you do, for all time steps, the array xout receives the state values and the array yout receives all the output values.

1 As necessary, adjust the torque and motion actuators in the model until you locate the desired motion state. The torques as you impose and measure them are the torques needed to produce the motion that you see.

2 Once you reach the desired motion state in a simulation, examine the components of xout, especially those representing the driveline states.

   If you want to use this state as a linearization point, extract the state vector from xout at a time step when the driveline is moving exactly as you want it to. Store the vector as x0, for example.

Linearizing a Driveline Model

For a specific example, see "Linearizing a Full Car at Rest" on page 3-42.

Consider linearizing a model with the Simulink linmod command. In this example, the model is called drive_model. If you need to locate a trim point before linearizing, first see "Trimming a Driveline with Inverse Dynamics" on page 3-32.

By default, the linmod command extracts linear models of the state space (A, B, C, D) form. If x is the state vector, u is the input vector, and y is the output vector, the matrices play the following roles in the linearized state space dynamics. The order of states, inputs, and outputs is the same in the linear model as in the full model.

$$\frac{d\boldsymbol{x}}{dt} = A \cdot \boldsymbol{x} + B \cdot \boldsymbol{u}$$
$$\boldsymbol{y} = C \cdot \boldsymbol{x} + D \cdot \boldsymbol{u}$$

Note During linearization, SimDriveline holds all input signals fixed at their values for the simulation time specified, including the gear ratios specified by Variable Ratio Gear blocks. The default linearization time is $t = 0$.

Linearizing at the Null State and Null Inputs

To linearize your model at zero simulation time and the combination x = []
and u = [], enter

```
[A,B,C,D] = linmod('drive_model');
```

This approach usually produces a trivial state space. A nontrivial state space
typically requires nonnull inputs u.

Linearizing at a Predefined State

If you have predefined state and input vectors x0 and u0, you can linearize in
that configuration and at zero simulation time by entering

```
[A,B,C,D] = linmod('drive_model',x0,u0);
```

You can save a predefined x0 vector from a trim operation. See "Trimming a
Driveline with Inverse Dynamics" on page 3-32.

A nonnull u0 is usually needed to obtain a nontrivial state space.

Other Linearization Options

Consult the command reference for linmod and the Using Simulink
documentation for complete linearization options.

Caution   If you apply explicitly time-dependent motion or torque actuators
to your driveline axes, your model is time dependent. Linearizing at the same
state but at different times leads to different results in that case.

Counting Driveline States in a Full Car

The full car model of "Simulating a Complete Car" on page 2-49 serves as an
example of degrees of freedom (DoFs) and states. Here you use a variant
version of the model called drive_full_car_trim.

"Finding and Using Driveline States" on page 3-31 discusses state counting
in general. This example is the starting point for the subsequent example,
"Trimming a Full Car to Rest" on page 3-40.

Identifying the Degrees of Freedom

First count the independent DoFs of the system. With all clutches unlocked, the model has four independent driveline DoFs.

1 Open `drive_full_car_trim`. Three subsystems, Converter with Lockup Clutch, Simpson 4-Speed, and Vehicle Load-Wheels-Road-Power Scope, are part of the driveline system. (There is also a Torque Actuator hidden in the Gasoline Engine block.)
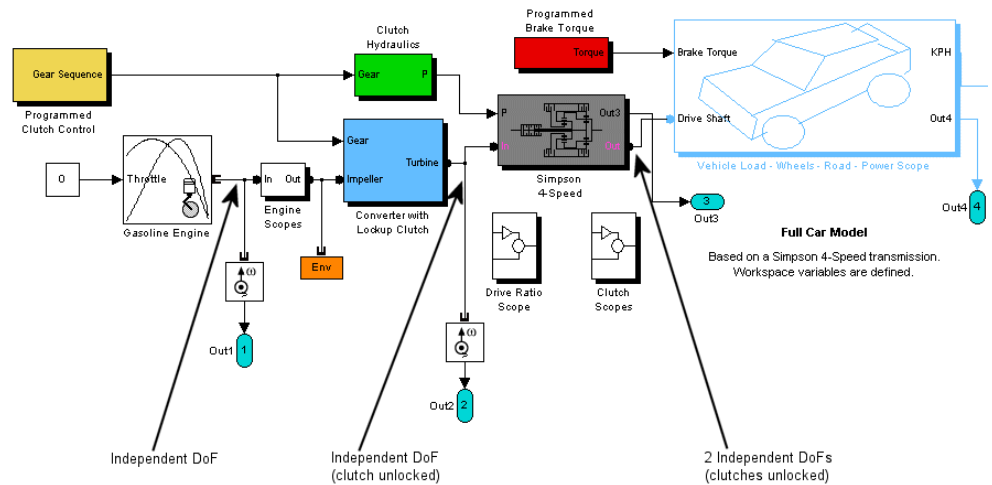
Full Car Model: Degrees of Freedom

2 Identify the driveline axis running from the Engine to the Converter subsystem. This is one independent DoF. Then open that subsystem.
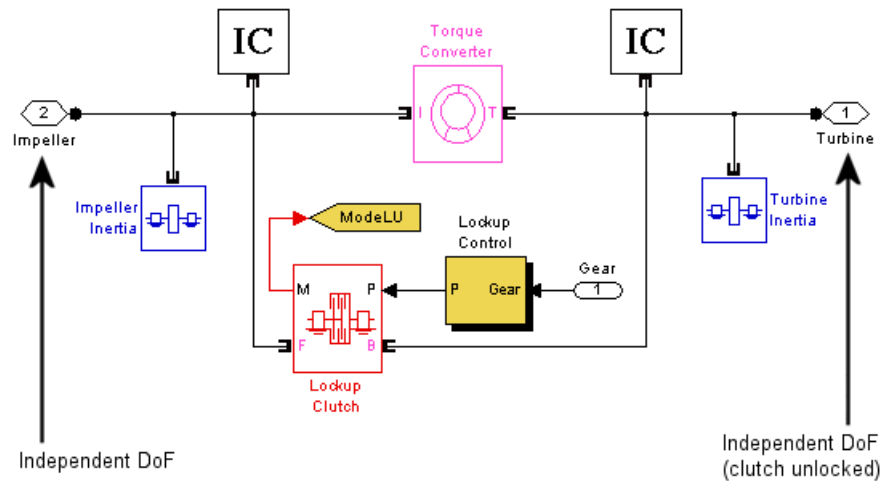
Full Car: Converter with Lockup Clutch Subsystem

The Torque Converter introduces a new independent DoF to the right. This axis then connects to the transmission. Close the Converter subsystem.

3 Open the Simpson 4-Speed transmission subsystem.
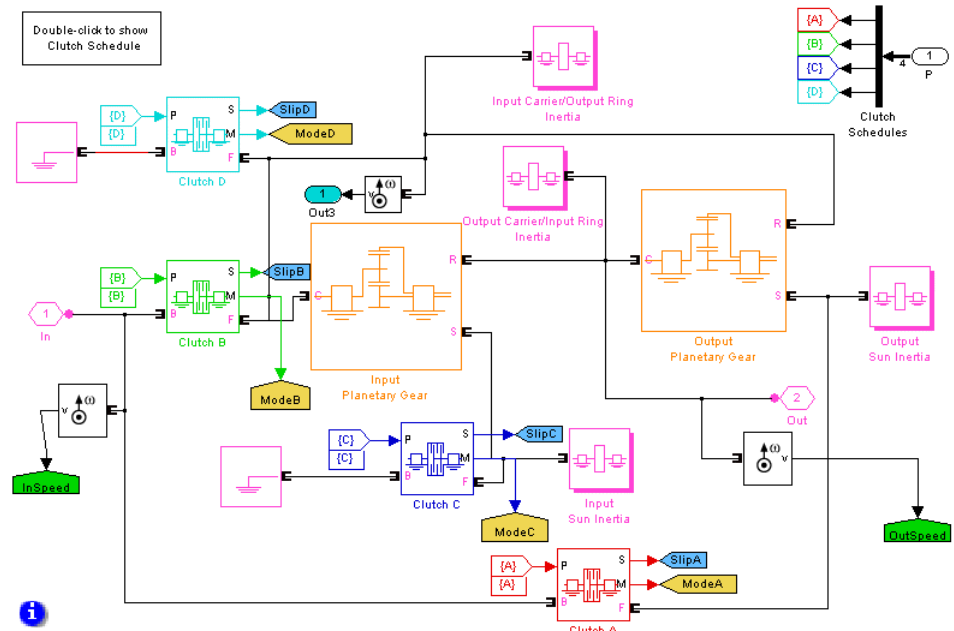
Full Car: Simpson 4-Speed Transmission Subsystem

This subsystem adds six new driveline connection lines. The two Housing and two complex Gear blocks add six apparent constraints. (The two Planetary Gear blocks are complex and impose two constraints each.) But these constraints are embedded into two independent loops, so only 6 − 2 = 4 constraints are independent. Thus the transmission introduces 6 − 4 = 2 new independent DoFs.

Close the transmission subsystem.

4 Look under the Vehicle Load subsystem mask.
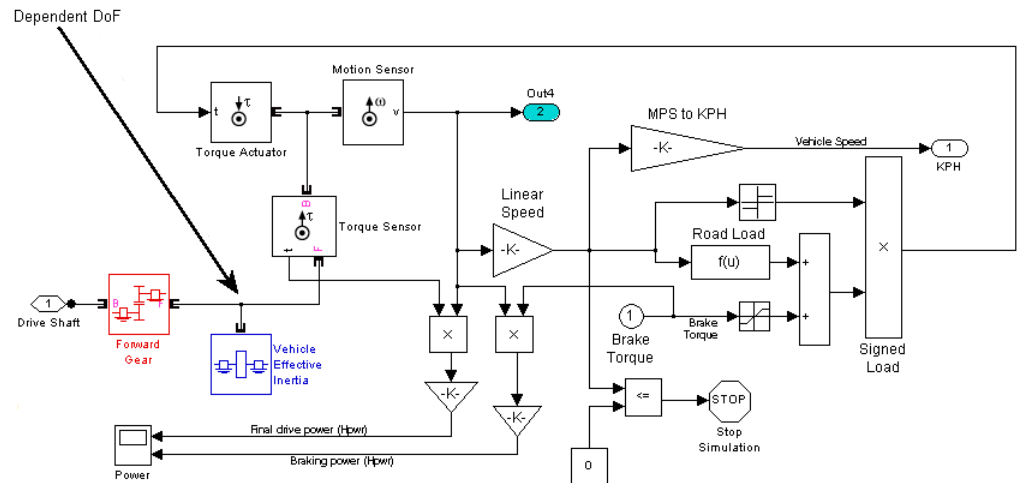
Full Car: Vehicle Load Subsystem

With the Forward Gear constraint, there are no new independent DoFs in this subsystem. Close it.

This model has four independent DoFs with all clutches unlocked:

- The first runs from the Engine to the Torque Converter.
- The second runs from the Torque Converter to the Simpson 4-Speed transmission.
- The transmission (in neutral) adds the third and fourth DoFs.

When the Transmission Locks. When you engage the Simpson 4-Speed transmission, two of its clutches lock, removing its two independent DoFs and reducing the total from four to two. They are the engine-to-converter axis and the converter-to-road axis.

There are still four driveline states, but two become dependent.

When the Lockup Clutch Also Locks. When the transmission is engaged and the lockup clutch in the Converter subsystem is also locked, only one independent DoF remains. This DoF is the entire drivetrain from engine to road.

Of the four driveline states, only one is independent.

Finding the States

Now list the model states.

```
[sys,xO,str,ts] = drive_full_car_trim([],[],[],'sizes');

sys
sys =
     9
     0
     4
     0
     0
     0
     2

str
str =
    [1x88 char]
    'drive_full_car_trim/Clutch Hydraulics/Actuator B Dynamics'
    'drive_full_car_trim/Clutch Hydraulics/Actuator D Dynamics'
    'drive_full_car_trim/Clutch Hydraulics/Actuator A Dynamics'
    'drive_full_car_trim/Clutch Hydraulics/Actuator C Dynamics'
    'drive_full_car_trim/Driveline Environment/Kinematic Block'
    'drive_full_car_trim/Driveline Environment/Kinematic Block'
    'drive_full_car_trim/Driveline Environment/Kinematic Block'
    'drive_full_car_trim/Driveline Environment/Kinematic Block'
```

The model has a total of nine states and four outputs. The driveline states are the last four, associated with the Kinematic subsystem of the Driveline Environment block.

Trimming a Full Car to Rest

Here you use the variant full car model called drive_full_car_trim to obtain a trim or operating point. In this example, the car's operating point is at rest.

"Trimming a Driveline with Inverse Dynamics" on page 3-32 discusses trimming in general. This example is based on the preceding example,

"Counting Driveline States in a Full Car" on page 3-35, and forms the starting point for the subsequent example, "Linearizing a Full Car at Rest" on page 3-42.

Configuring the Model for Trimming

Open and examine drive_full_car_trim.
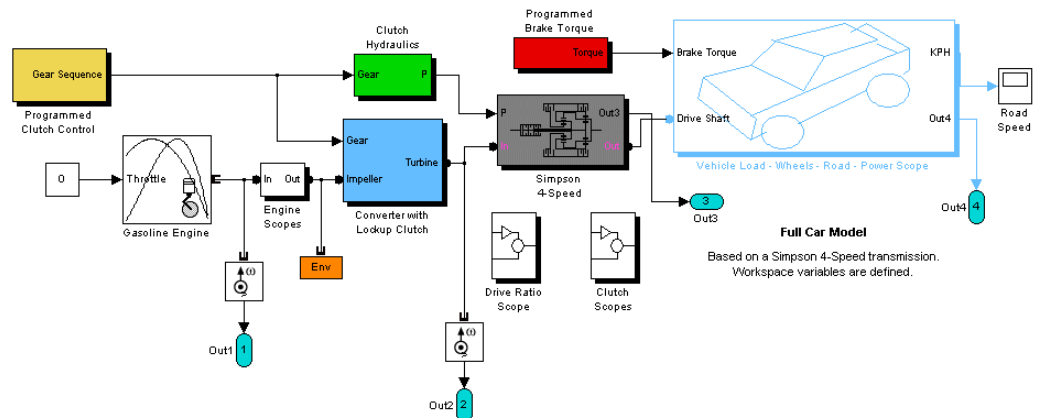
Full Car Model Customized for Trimming

- The Controllable Friction Clutch blocks A and D of the Simpson 4-Speed transmission subsystem are set to lock when the simulation starts. Thus the transmission starts engaged.

- The four driveline degrees of freedom (DoFs) are reduced to two by transmission locking. The lockup clutch in the Converter with Lockup Clutch subsystem does not lock.

- The model contains four Motion Sensor and four Outport blocks to capture the angular velocities of four driveline axes, corresponding to the four model DoFs. (If you want to measure torques instead, replace Motion Sensors with Torque Sensor blocks.)

- The Gasoline Engine throttle signal is a Constant block with value 0. Thus the car runs without any engine power. It starts with a small Torque Converter impulse and rolls to a stop.

- By examining **Simulation** > **Configuration Parameters** > **Data Import/Export**, you can confirm that the time, model states, and model outputs are logged to `tout`, `xout`, and `yout`, respectively.

Reading the Model States and Outputs

Examine the simulation data.

1 Open the Road Speed scope and run the model. The simulation stops when the car reaches rest.

2 In your workspace, read the state data from `xout` and angular velocity data from `yout`. Each row of these arrays represents a time step; the columns are the vector components.

Defining an Operating Point from the State

To linearize the model at the rest state, you need to first define that state. At the last simulation time step, the car is not exactly at, but very close, to rest. Save this state as a vector `x0` and use it as the linearization point.

1 Determine the number of time steps by examining `tout` in the workspace or entering at the command line

```
length(tout)
```

2 Define the quasi-rest state in your workspace by extracting the last entry in `xout`. At the command line, enter

```
x0 = xout(length(tout),:);
```

`x0` and each row of `xout` have nine components, corresponding to the nine states obtained in "Counting Driveline States in a Full Car" on page 3-35.

Linearizing a Full Car at Rest

Here you use the variant full car model called `drive_full_car_trim` to linearize the full car at the operating point of rest.

"Linearizing a Driveline Model" on page 3-34 discusses linearization in general. This example is based on the previous example, "Trimming a Full Car

to Rest" on page 3-40, which you need to carry out before proceeding here. The result of that example is a saved linearization point, the quasi-rest state x0.

Creating Model Inputs

To generate a valid and nontrivial linearized model, you need to insert at least one model input. The simplest choice is to convert the Gasoline Engine throttle signal into a model input.

1 Disconnect the zero Constant block from the Gasoline Engine block.
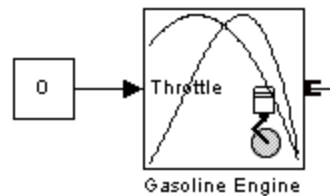
2 Replace it with an Inport block with **Port number** set to 1.
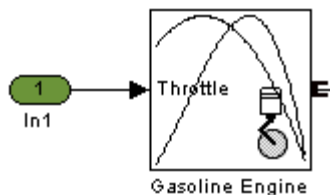
Obtaining the Linearized Model

Linearize the model with the linmod command.

1 At the command line, define the input throttle signal by entering

```
u0 = 0;
```

2 Then obtain the state space model arrays *(A, B, C, D)* by entering

```
[A,B,C,D] = linmod('drive_full_car_trim',x0,u0);
```

linmod carries out a clutch mode iteration first and locks clutches A and D of the Simpson 4-Speed transmission subsystem.

There are nine states (nine components in x), one input (one component in u), and four outputs (four components in y). (Four of the nine states are driveline states.) Thus A is 9-by-9, B is 9-by-1, C is 4-by-9, and D is 4-by-1.

Finding the Minimal Realization of the State Space

Note This step requires the Control System Toolbox.

Although there are four apparent driveline DoFs, with the Simpson transmission engaged, only two are independent. Thus not all the states of x are independent, and the state space form *(A, B, C, D)* contains redundant information.

Reduce the state space model to its minimal form by entering at the command line

```
[a,b,c,d] = minreal(A,B,C,D);
7 states removed.
```

Of nine states, state space reduction removes seven, leaving two states, corresponding to the two independent DoFs. There are still four outputs in y and one input in u. Thus a is 2-by-2, b is 2-by-1, c is 4-by-2, and d is 4-by-1.

Generating Code

You can use SimDriveline together with Real-Time Workshop to generate stand-alone C or C++ code from your mechanical models and enhance simulation speed and portability. Certain features of Simulink itself make use of generated or external code. This section explains these features.

- "Related Simulink Code Generation Documentation" on page 3-45
- "Reasons for Generating Code" on page 3-45
- "Using Code-Related Products and Features" on page 3-46

Some SimDriveline features are restricted when you translate a model into code. See "Restrictions with Generated Code" on page 3-46.

Note Code generated from SimDriveline models is intended for rapid prototyping and hardware-in-the-loop applications. It is not intended for use as production code in embedded controller applications.

Related Simulink Code Generation Documentation

Consult the documentation for Real-Time Workshop, xPC Target, Simulink Accelerator, and Simulink S-functions for general information on installing and using these products.

Reasons for Generating Code

There are many purposes and methods associated with code generation in Simulink. In SimDriveline, there are three essential rationales:

- Compiled code versions of Simulink models run faster than the original block diagram models. The time savings for regular Simulink models can be dramatic. For the SimDriveline portions of a model, the performance improvements are also significant.

- A more important consideration for SimDriveline models is that converting the SimDriveline part of a model to code frees your model from requiring SimDriveline to run. For example, converting a SimDriveline subsystem to an S-function block allows you to run a model with Simulink alone.

- An equally important consideration for SimDriveline is the stand-alone implementation of generated/compiled code. Once you convert part or all of your model to code, you can deploy the stand-alone executable program on virtually any platform, independently of MATLAB.

  Converting a model or subsystem to code also hides the original model or subsystem.

Using Code-Related Products and Features

Simulink, Real-Time Workshop, and xPC Target, using several code-related technologies, enable you to link existing code to your models and generate code versions of your models.

Code-Related Task	Component or Feature
Link existing code written in C or other supported languages to Simulink models	Simulink S-functions to generate customized blocks
Speed up Simulink simulations	Simulink Accelerator
Generate stand-alone fixed-step code from Simulink models	Real-Time Workshop
Generate stand-alone variable-step code from Simulink models	Real-Time Workshop Rapid Simulation Target (RSIM)
Convert Simulink model to code and compile and run it on a target PC	Real-Time Workshop and xPC Target
Generate a block representing a Simulink model	S-function Target
Generate code for designated models or subsystems	Model Reference

Restrictions with Generated Code

There are certain limitations on code generated from models containing SimDriveline blocks.

Mode Iteration Disabled for Code Generation

In the default simulation mode, SimDriveline uses mode iteration to determine the locking and unlocking of clutches, suspending the simulation in time and entering an algebraic loop. Through the `Driveline Environment` block, present in each distinct driveline, you can change the simulation mode to turn off mode iteration while running your Simulink model. In that case, SimDriveline determines clutch modes over multiple time steps while the simulation continues. Turning off mode iteration increases simulation speed somewhat but at the possible cost of accuracy.

However, in the generated code versions of SimDriveline models, mode iteration is turned off automatically. (This is also true of Simulink Accelerator.) Clutch locking and unlocking are determined over multiple time steps.

Code Generation and Fixed-Step Solvers

Most code generation options for SimDriveline models require the use of fixed-step Simulink solvers. This table summarizes the available solver choices, depending on how you generate code. See "Improving Performance" on page 3-2 for more about adjusting Simulink solvers for driveline models.

Code Generation Option	Solver Choices
Simulink Accelerator	Variable-step or fixed-step
Real-Time Workshop: RSIM Target	Variable-step or fixed-step
Real-Time Workshop: Targets other than RSIM	Fixed-step only

Fixed-Point Not Supported by SimDriveline Code Generation

You must run code generated from models containing SimDriveline blocks on floating-point processors.

4

Blocks — Categorical List

This chapter lists all SimDriveline blocks arranged by library. The next chapter contains reference information for all blocks in SimDriveline, arranged in alphabetical order by block name.

Use the Simulink Library Browser or the SimDriveline library to access the blocks directly, guided by this hierarchical library list.

Creating a Model

Driveline Environment	Represent the SimDriveline environment
Housing	Rotationally lock the connected driveline axis and prevent it from turning
Inertia	Represent a body with rotational inertia
Shared Environment	Connect two driveline components so that they share the same driveline environment

Modeling Gears

Differential	Represent a differential gear with specified gear ratio
Dual-Ratio Planetary	Represent a set of carrier, sun, planet, and ring gear wheels with specified ring-planet and planet-sun gear ratios
Planet-Planet	Represent a set of carrier, inner planet, and outer planet gear wheels with specified planet-planet gear ratio
Planetary Gear	Represent a set of carrier, sun, planet, and ring gear wheels with specified ring-sun gear ratio
Ravigneaux	Represent a Ravigneaux planetary set of carrier, sun, planet, and ring gear wheels with specified ring-sun gear ratios
Ring-Planet	Represent a set of carrier, planet, and ring gear wheels with specified ring-planet gear ratio
Simple Gear	Represent a gear with fixed gear ratio
Variable Ratio Gear	Represent a gear with controllable, variable gear ratio

Modeling Dynamic Elements

Controllable Friction Clutch	Represent a frictional clutch with kinetic and static friction and controlled by a pressure signal
Hard Stop	Model the restriction on the relative angular motion of two driveline axes to a free gap with elastic upper and lower limits
Torque Converter	Transfer torque between two driveline axes as a function of their relative angular velocity
Torsional Spring-Damper	Represent a damped torsional spring torque, with a free play gap, acting between two rotating axes

Modeling Transmissions

Lepelletier 6-Speed	Model a six-speed Lepelletier transmission based on a planetary gear and a Ravigneaux gear
Lepelletier 7-Speed	Model a seven-speed Lepelletier transmission based on a planetary gear and a Ravigneaux gear
Ravigneaux 4-Speed	Model a Ravigneaux four-speed transmission based on a Ravigneaux gear
Simpson 4-Speed	Model a Simpson four-speed transmission based on two planetary gear sets

Modeling Vehicle Components

Diesel Engine	Model a diesel fuel engine with throttle control, speed governor, and driveline output
Gasoline Engine	Model a gasoline fuel engine with throttle control, speed governor, and driveline output
Longitudinal Vehicle Dynamics	Model longitudinal dynamics and motion of a two-axle, four-wheel vehicle
Tire	Model tire dynamics and motion at the end of a driveline axis

Sensing and Actuating Motion

Initial Condition	Set the initial angular velocity of a driveline axis to a nonzero value
Motion Actuator	Actuate a driveline axis with specified motions
Motion Sensor	Measure the motion of a driveline axis
Torque Actuator	Actuate a driveline axis with a specified torque
Torque Sensor	Measure the torque transferred along a driveline axis

Additional Useful Blocks

Connection Port	Create a Physical Modeling connector port for a subsystem

Blocks — Alphabetical List

Connection Port

Purpose   Create a Physical Modeling connector port for a subsystem

Library   Utilities

Description   The Connection Port block, placed inside a subsystem composed of SimDriveline blocks, creates an open SimDriveline round connector port ○ on the boundary of the subsystem. Once connected to a connection line, the port becomes a driveline connector port ⊏.
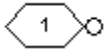
You connect individual SimDriveline blocks and subsystems made of SimDriveline blocks to one another with SimDriveline connection lines instead of normal Simulink signal lines. These are anchored at the open square driveline connector ports ⊏. Subsystems constructed from SimDriveline blocks automatically have such driveline connector ports. You can add additional connector ports by adding Connection Port blocks to your subsystem.

Dialog Box and Parameters
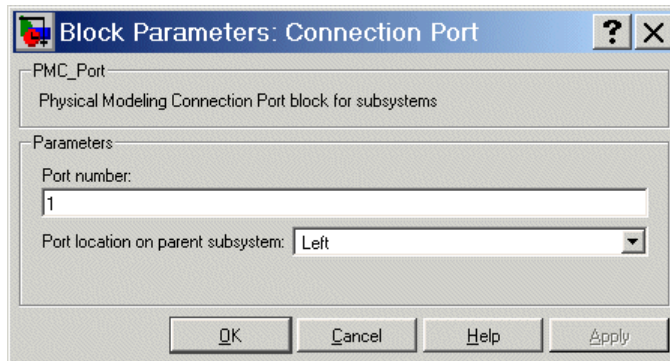
Port number

Labels the subsystem connector port created by this block. Each connector port on the boundary of a single subsystem requires a unique number as a label. The default value for the first port is 1.

Port location on parent subsystem
> Choose here on which side of the parent subsystem boundary the port is placed. The choices are `Left` or `Right`. The default choice is `Left`.

See Also   In the Using Simulink documentation, see Creating Masked Subsystems.

Controllable Friction Clutch

Purpose   Represent a frictional clutch with kinetic and static friction and controlled by a pressure signal

Library   Dynamic Elements

Description

A friction clutch transfers torque between two driveline axes by coupling them with friction. The Controllable Friction Clutch block models a standard friction clutch with kinetic friction and static (locking) friction acting on the two axes. The motion is measured as follower (F) axis relative to base (B) axis, $\omega = \omega_F - \omega_B$.
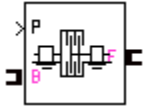
The friction clutch has two possible directionalities:

- Bidirectional ($\omega \leq 0$ or $\omega \geq 0$), allowing the follower to rotate relative to the base in either direction

- Unidirectional ($\omega \geq 0$), allowing the follower to rotate relative to the base in the forward direction only. The follower and base can also rotate locked together ($\omega = 0$), in the same direction, at the same rate.

Note If you want a unidirectional clutch that allows the follower to rotate relative to the base in the reverse direction only, connect the Controllable Friction block in your driveline with reversed orientation, follower (F) to base (B).

The clutch requires a dimensionless input pressure signal P that modulates the applied friction. This signal should be positive or zero. A signal P less than zero is interpreted as zero.

Clutch Friction and Locking

A friction clutch can be in one of three states:

- *Unengaged*, when the clutch applies no friction at all. The frictional surfaces are not in contact. The follower and base are independent, and no torque is transferred between them.

- *Engaged* (but not locked), when the clutch applies kinetic friction as the frictional surfaces slip past one another. The follower and base remain independent, but some torque is transferred between them.

- *Locked*, when the clutch applies static friction. The frictional surfaces are locked together and do not slip. The follower and base effectively form a single axis. This state transfers the maximum torque possible.

There is also a fourth, virtual state between locked and unlocked called the *wait state* (see "Friction Clutch Model" on page 5-11).

The kinetic friction torque requires that you specify five factors:

- Number of friction surfaces

- Effective torque radius

- Peak normal force

- Normalized pressure signal P and pressure threshold P_{th}

- Kinetic friction coefficient μ expressed as a tabulated discrete function of ω

Locking requires that

- The relative speed $|\omega|$ be smaller than a velocity threshold ω_{Tol}.

- The normalized pressure P exceed the pressure threshold P_{th}.

The static friction torque controls the unlocking of a friction clutch. (You can optionally lock the clutch at the start of the simulation as well.) When the clutch is locked, it remains locked unless the friction constraint torque across the clutch exceeds a static friction limit. By default, SimDriveline decides when to unlock a clutch after repeatedly suspending the simulation in time and testing a set of unlocking conditions (see "Friction Clutch Model" on page 5-11). This testing is called *mode iteration*.

Controllable Friction Clutch

Clutch Locking and Degrees of Freedom

If it locks, a Controllable Friction Clutch block imposes a constraint on your driveline system by requiring that two otherwise independent angular velocities be equal. A locked clutch thus reduces the number of independent degrees of freedom by one.

Enabling and Disabling Clutch Mode Iteration

You can turn off mode iteration for a whole driveline machine from that machine's Driveline Environment block. In that case, SimDriveline tests the friction clutches of your model without mode iteration. Instead, the locking and unlocking tests are applied over multiple time steps, improving your simulation performance, but possibly decreasing its accuracy.

Mode Iteration and Code Generation

In the default simulation mode, SimDriveline simulates clutch mode changes with mode iteration turned on. However, in the generated code versions of SimDriveline models, mode iteration is turned off automatically. Clutch locking and unlocking are determined over multiple time steps.

Setting the Velocity Tolerance

You can set the *velocity tolerance* or threshold ω_{Tol} for a clutch in a number of ways, using a value specified in the clutch itself or in the Driveline Environment block connected to its driveline, and depending on whether you are using a variable- or fixed-step solver.

- You can allow the clutch to use the driveline-wide default velocity tolerance settings specified in the Driveline Environment block. (This is the default configuration of the clutch.)

- You can override the driveline-wide default velocity tolerance settings by allowing a particular clutch to automatically compute a velocity tolerance from solver settings. (This option is available only if you use a variable-step solver.)

- You can override the driveline-wide default velocity tolerance settings by specifying an individual velocity tolerance value for a particular clutch.

Note Variable-step solvers allow you to use either automatically computed or explicitly specified values for ω_{Tol}. Fixed-step solvers require you to specify a value for ω_{Tol}, either in each clutch or in the connected Driveline Environment block.

Using Dynamic Element Blocks

Use the blocks of the Dynamic Elements library as a starting point for vehicle modeling. To see how a Dynamic Element block models a driveline component, look under the block mask. The blocks of this library serve as suggestions for developing variant or entirely new models to simulate the same components. Break the block's library link before modifying it and creating your own version.

Controllable Friction Clutch

Dialog Box and Parameters

Block Parameters: Controllable Friction Clutch

Controllable Friction Clutch (mask)

Models a clutch coupling the input base (B) and output follower (F) axes with controllable kinetic and static locking friction. The clutch can be bidirectional, with the follower axis moving relative to the base axis with either sign; or unidirectional, with the follower moving relative to the base with positive sign only.

For the clutch to lock, the relative follower-base speed must drop below the velocity tolerance, and the normalized pressure must be greater than the engagement threshold. A locked clutch remains locked unless the friction constraint torque across the clutch exceeds the static friction limit = (Static friction peak factor)*(Kinetic friction torque).

You must either specify a velocity tolerance for this clutch or select the "Use default clutch velocity tolerance" check box to get the default settings for this driveline entered in its connected Driveline Environment block. If you use a variable-step solver, selecting the "Use automatic clutch velocity tolerance" check box causes the clutch to automatically calculate a velocity tolerance based on solver settings. If you use a fixed-step solver, the velocity tolerance is either the value specified here or in the Driveline Environment block. The tolerance settings you enter here override the settings specified in the Driveline Environment block.

Parameters

Directionality: Bidirectional

Number of friction surfaces:

6

Effective torque radius (m):

0.04

Peak normal force (N):

1000

Coefficient of friction table:

[0 1; 1e8 1]

Static friction peak factor:

1.1

Engagement threshold for normalized pressure:

0

☑ Use default clutch velocity tolerance from the Driveline Environment block

☑ Use automatic clutch velocity tolerance for variable-step solvers (instead of the value below)

Velocity tolerance (rad/s):

1e-3

☐ Lock clutch at start of simulation

☐ Show clutch slip (rad/s) port (S)

OK   Cancel   Help   Apply

Directionality

Select `Bidirectional` or `Unidirectional` to determine how the follower axis can turn relative to the base, in both directions or only in the forward direction, respectively. The default is `Bidirectional`.

Number of friction surfaces

Number of friction-generating contact surfaces inside the clutch. The default is `6`.

Effective torque radius

The effective moment arm radius, in meters (m), that determines the kinetic friction torque inside the clutch. The default is `0.04`.

Peak normal force

The maximum force, in newtons (N), normal to the frictional surfaces in the clutch. This value normalizes the Simulink input signal P for clutch pressure and determines the maximum kinetic friction torque. The default is `1000`.

Coefficient of friction table

Dimensionless kinetic friction factor μ as a function of the angular velocity in tabular form. The table is a matrix whose rows are vectors of length 2, separated by semicolons. Each two-vector specifies a pair of values, an angular velocity ω and a corresponding μ value, in that order. The simulation automatically interpolates a cubic spline from these values.

The default matrix is `[0 1; 1e8 1]`, which is a constant μ of value 1. The angular velocity values are in units of radians/second.

Static friction peak factor

Ratio of the static friction limit for clutch locking to the kinetic friction. The default is `1.1`.

Engagement threshold for normalized pressure

Minimum normalized pressure P_{th} that activates clutch friction. If the normalized pressure input signal P is less than this threshold, the clutch applies no friction. The default is `0`.
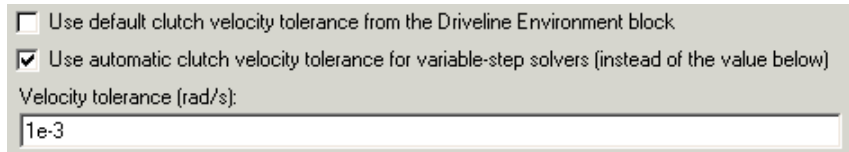
Controllable Friction Clutch

Use default clutch velocity tolerance from the Driveline Environment block

> Select to require this clutch to use the driveline-wide velocity tolerance ω_{Tol} specified in the Driveline Environment block connected to the driveline. The default is selected.

> If you unselect this check box, you enable the **Use automatic clutch velocity tolerance for variable-step solver** check box and the **Velocity tolerance** field (see following).

Use automatic clutch velocity tolerance for variable-step solver

> Select to require this clutch to compute the velocity tolerance ω_{Tol} automatically from solver settings. The default is selected.

> This check box is enabled only if the **Use default clutch velocity tolerance from the Driveline Environment block** check box is unselected.

Velocity tolerance

> Sets the minimum angular velocity ω_{Tol} above which the clutch cannot lock. Below this velocity, the clutch can lock. (See the diagram, Clutch States and Transitions on page 5-15.) The units are radians/second. The default is 1e-3.

> For a fixed-step solver, SimDriveline always uses this value for this clutch, if you do not specify a default velocity tolerance through the Driveline Environment block.

> This field is enabled only if the **Use default clutch velocity tolerance from the Driveline Environment block** check box is unselected.

☐ Lock clutch at start of simulation

☐ Show clutch slip (rad/s) port (S)

☐ Show power dissipation (W) port (L)

☐ Show mode signal port (M)

Lock clutch at start of simulation

Select to start the simulation with the clutch already locked. The default is unselected.

Show clutch slip port (S)

Select to make available the Simulink outport for the clutch slippage signal. The default is unselected.

The clutch slippage is the relative angular velocity ω of the two coupled driveline axes. The signal measures the clutch slippage in radians/second.

Show power dissipation port (L)

Select to make available the Simulink outport for the power dissipation signal. The default is unselected.

The signal measures the power, in watts (W), being dissipated by friction torques applied by the clutch to the driveline axis.

Show mode signal port (M)

Select to make available the Simulink port for the discrete clutch mode signal. The default is unselected.

The signal value is the sign ±1 of the angular velocity ω of the follower relative to the base. If the follower and base are locked and rotate together, the signal value is 0.

Friction Clutch Model

When you apply a pressure signal above threshold ($P \geq P_{\text{th}}$), the Controllable Friction Clutch block applies two kinds of friction to the driveline motion, *kinetic* and *static*. The clutch applies kinetic friction torque only when one driveline axis is spinning relative to the other driveline axis. The clutch applies static friction torque when the two

Controllable Friction Clutch

driveline axes are locked and spin together. SimDriveline iterates through multistep conditional testing to decide when to unlock the clutch.

Unlocked State: Kinetic Friction

The kinetic friction torque is a product of five factors:

$$\tau_K = \mu \cdot (\text{number of friction disks}) \cdot (\text{effective torque radius}) \cdot (\text{peak normal force}) \cdot \max[(P - P_{\text{th}}), 0]$$

The effective torque radius r_{eff} is the effective radius, measured from the driveline axis, at which the frictional forces are applied at the frictional surfaces. It is related to the geometry of the friction disk by

$$r_{\text{eff}} = \frac{2}{3} \frac{r_o^3 - r_i^3}{r_o^2 - r_i^2}$$

r_o and r_i are the outer and inner radii, respectively, of the friction disk.

You specify the kinetic friction coefficient μ as a tabulated discrete function of relative angular velocity ω. This function is assumed to be symmetric for positive and negative values of the relative angular velocity, so that you need to specify μ for positive values of ω only. The peak normal force is the normal force applied to the frictional surfaces when the normalized pressure signal P is one.

The pressure signal should be nonnegative. If P is less than P_{th}, the clutch applies no friction at all.

Locked State: Static Friction

Once a friction clutch locks, it remains locked unless and until the torque across the clutch exceeds the static friction limit. The static friction limit is a product of two factors:

(static friction peak factor)·(kinetic friction torque for $\omega \to 0$)

SimDriveline computes the kinetic friction torque with the kinetic friction coefficient μ interpolated to zero relative angular velocity with a cubic spline.

The static friction peak factor measures how much larger the static friction is compared to the kinetic friction at the instant of unlocking, when $\omega = 0$.

Wait State: How the Friction Clutch Locks and Unlocks

If the pressure signal $P \geq P_{th}$, the friction clutch locks the two driveline axes together when $|\omega| \leq \omega_{Tol}$. You can also lock a clutch before the simulation starts with the **Lock clutch at start of simulation** option in the dialog. In either case, SimDriveline treats a locked clutch as an extra constraint on the angular motion of the system.

If the absolute value of the friction constraint torque applied across the two driveline axes exceeds the static friction limit, the clutch enters the wait state and might unlock. If a clutch unlocks, the unlocking removes a constraint from your driveline system and restores an independent degree of freedom.

The unlocking of a friction clutch is a conditional, multistep process implemented internally by SimDriveline.

- If you turn off mode iteration for your driveline model (in the Driveline Environment block dialog), the clutch unlocks over multiple simulation time steps.

- If you leave it on, SimDriveline suspends the simulation in time and starts mode iteration to determine whether to unlock the clutch.

The wait state encompasses the steps that test the entire machine for unlocking.

1 SimDriveline first checks the relative acceleration $\alpha = d\omega/dt$ of the two driveline axes, given the torques present when the clutch enters the wait state.

The clutch returns from the wait state to the locked state if

- The whole machine requires the axes to turn in the relative forward direction, but α is negative

- The whole machine requires the axes to turn in the relative reverse direction, but α is positive

2 If the clutch remains in the wait state instead of returning to locked, SimDriveline integrates the relative acceleration in time to obtain the absolute value of the virtual angular velocity and checks this result against angular velocity tolerance ω_{Tol}. If the result is less than ω_{Tol}, the clutch returns to the start of the wait state and the relative acceleration check. If the result exceeds ω_{Tol}, the clutch unlocks.

3 In the unlocked state, the clutch begins applying kinetic friction again. SimDriveline also begins again to check for the locking condition ($|\omega| = \omega_{Tol}$ with $P > P_{th}$).

Clutch State Summary

This chart summarizes the possible states and transitions of a bidirectional clutch. The states and transitions of a unidirectional clutch consist of just the right half of this diagram.
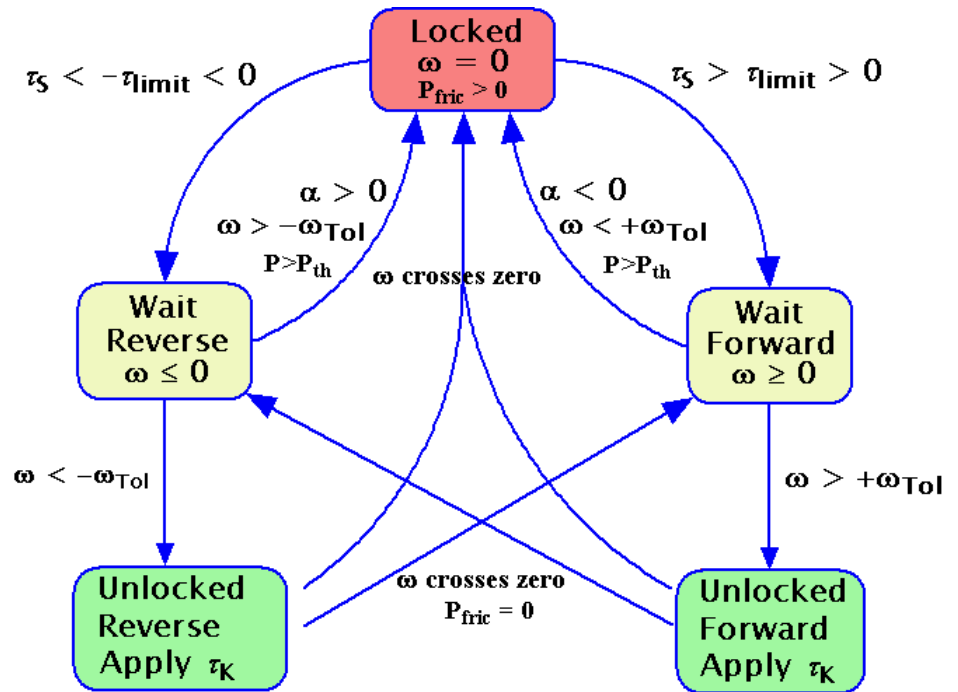
Clutch States and Transitions

Clutch State Variables

Symbol	Meaning	Definition
ω	Relative angular velocity	$\omega_F - \omega_B$
α	Relative angular acceleration	$d\omega/dt$
ω_{Tol}	Relative angular velocity tolerance for clutch locking	See "Friction Clutch Model" on page 5-11

Controllable Friction Clutch

Symbol	Meaning	Definition
τ_K	Kinetic friction torque	See "Friction Clutch Model" on page 5-11
τ_S	Static friction torque	Whatever is necessary to maintain locking, unless limit is exceeded
τ_{limit}	Static friction torque limit	(static friction peak factor)·(kinetic friction torque for $\omega \rightarrow 0$) *(see text)*
P_{fric}	Clutch friction capacity	$|\omega| \cdot \tau_K$

Examples   These SimDriveline demo models contain working examples of clutches used to change gear couplings:

- drive_sclutch

- drive_clutch_engage

- drive_strans

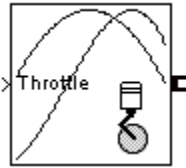- drive_simpson

- drive_full_car

- drive_vehicle

See Also   Differential, Driveline Environment, Torque Converter

Purpose   Model a diesel fuel engine with throttle control, speed governor, and driveline output

Library   Vehicle Components

Description

The Diesel Engine block models a diesel-fuel, compression-ignition engine with a speed governor. The engine runs at a variable speed that you can control with a Simulink throttle signal. The throttle signal directly controls the output torque that the engine generates and indirectly controls the speed at which the engine runs. If the engine speed exceeds the maximum speed that you specify, the engine torque drops to zero. The model does not include the air-fuel dynamics of combustion.

The block accepts the throttle signal through a Simulink inport. This signal specifies the engine torque as a fraction of the maximum torque possible in a steady state at a fixed engine speed and should lie between 0 and 1. A throttle signal below zero is interpreted as zero; above one, as one.

To prevent engine power and torque from becoming negative, the block imposes an upper limit on the maximum engine speed itself. If the maximum speed you specify exceeds this limit, SimDriveline issues an error that indicates what the upper limit is.

Using Vehicle Component Blocks

Use the blocks of the Vehicle Components library as a starting point for vehicle modeling. To see how a Vehicle Component block models a driveline component, look under the block mask. The blocks of this library serve as suggestions for developing variant or entirely new models to simulate the same components. Break the block's library link before modifying it and creating your own version.

Diesel Engine

Dialog Box and Parameters

Block Parameters: Diesel Engine

Diesel Engine (mask) (link)

Simple diesel fuel engine model with speed governor. The throttle input signal lies between zero and one and specifies the torque demanded from the engine as a fraction of the maximum possible torque. This signal also indirectly controls the engine speed. The engine model does not include air-fuel combustion dynamics.

If the throttle signal rises above one or falls below zero, the engine limits the signal to one or zero, respectively. If the engine speed rises above the maximum speed, the engine torque drops to zero.

Parameters

Maximum power [W]:

150000

Speed at maximum power [rpm]:

3500

Maximum speed [rpm]:

4500

OK   Cancel   Help   Apply

Maximum power
> Maximum power that the engine can output, in watts (W). The default is 150000.

Speed at maximum power
> Engine speed, in revolutions per minute (rpm), when the engine is running at maximum power. The default is 3500.

Maximum speed
> Maximum speed, in revolutions per minute (rpm), at which the engine can turn. The default is 4500.
>
> During simulation, if the engine speed exceeds this maximum, the engine torque drops to zero.

Engine Model

The engine model uses a programmed relationship between torque and speed, modulated by the throttle signal. An actual diesel engine does not have a throttle.

Engine Speed, Torque, and Throttle

The engine model is specified by an *engine torque demand* function $g(\Omega)$ built into the block. It provides the maximum torque available for a given engine speed Ω. The block dialog entries (maximum power, speed at maximum power, and maximum speed) normalize this function to physical maximum torque and speed values.

The throttle input signal T specifies the actual engine torque delivered as a fraction of the maximum torque possible in a steady state at a fixed engine speed. It modulates the actual torque delivered τ from the engine: $\tau = T \cdot g(\Omega)$. The actual engine drive shaft speed Ω is fed back to the engine input.

Engine Power and Torque Demand

The demand function *g(Ω)* is specified in terms of the steady-state engine power *P(Ω)*.

The engine speed is limited to a maximum: $0 \le \Omega \le \Omega_{max}$. The absolute maximum engine power P_{max} defines Ω_0 such that $P_{max} = P(\Omega_0)$. Define $w = \Omega/\Omega_0$ and $P(\Omega) = P_{max} \cdot p(w)$. Then $p(1) = 1$ and $dp(1)/dw = 0$. Power is the product of torque and angular velocity. The torque demand function is thus

$$\tau_{max} = g(w) = (P_{max}/\Omega_0) \cdot [p(w)/w]$$

You can derive forms for *p(w)* from engine data and models.
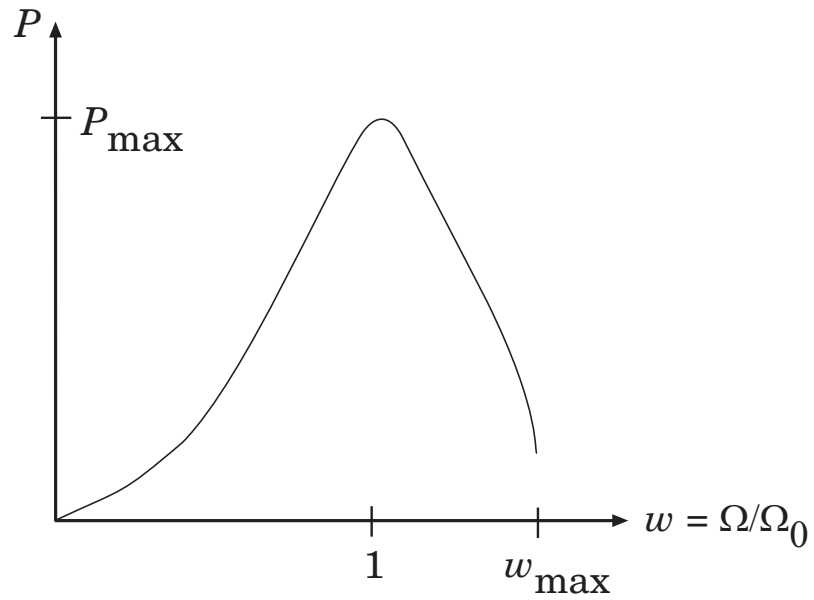
SimDriveline uses a polynomial form for *P(Ω)*:

$$p(w) = p_1 w + p_2 w^2 - p_3 w^3$$

satisfying

$$p_1 + p_2 - p_3 = 1 \text{ and } p_1 + 2p_2 - 3p_3 = 0$$

Diesel Engine
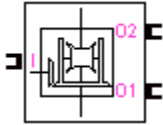

Typical Engine Power Demand Function

See Also   Controllable Friction Clutch, Gasoline Engine, Torque
Converter

Purpose   Represent a differential gear with specified gear ratio

Library   Gears

Description   The Differential block represents a differential gear that couples rotational motion about the longitudinal axis to rotational motion about two lateral axes.

Any one axis can be the input. In normal use, the longitudinal shaft is the input, and motion, torque, and power flow out through the lateral shafts. The output axes, in general, have different angular velocities. The longitudinal motion is divided by the drive gear ratio that you specify and then split between the two lateral shafts.

Differentials in drivelines often have a controllable clutch connecting the two output shafts. You can add this clutch control by appropriately connecting a Controllable Friction Clutch block to the Differential block.

Axis Motions and Constraints

The three rotational degrees of freedom, the longitudinal ω_I and the lateral ω_{O1} and ω_{O2}, are subject to one gear constraint and reduce to two independent degrees of freedom. In terms of the drive gear ratio g_D, the longitudinal motion is related to the sum of the lateral motions:

$$\omega_I = (1/2) \cdot g_D(\omega_{O1} + \omega_{O2})$$

The *sum* of the lateral motions depends on the longitudinal motion, once the longitudinal axis is connected. The *difference* of lateral motions $\omega_{O1} - \omega_{O2}$ is independent of the longitudinal motion. These two independent degrees of freedom have this physical significance:

- One degree of freedom is equivalent to the two lateral shafts rotating at the same angular velocity ($\omega_{O1} = \omega_{O2}$) and at a fixed ratio with respect to the longitudinal shaft.

- The other degree of freedom is equivalent to keeping the longitudinal shaft locked ($\omega_I = 0$) while the lateral shafts rotate with respect to each other in opposite directions ($\omega_{O1} = -\omega_{O2}$).

Differential

The general motion of the lateral shafts is a superposition of these two motions.

The torques along the lateral axes, τ_{O1} and τ_{O2}, are constrained to the longitudinal torque τ_I in such a way that the power input equals the sum of the power outputs:

$$\omega_I \tau_I = \omega_{O1}\tau_{O1} + \omega_{O2}\tau_{O2}$$

When the kinematic and power constraints are combined,

$$g_D\tau_I = 2(\omega_{O1}\tau_{O1} + \omega_{O2}\tau_{O2}) / (\omega_{O1} + \omega_{O2})$$

Caution   All gear ratios must be strictly positive. If any gear ratio equals 0 or becomes negative at any time during a simulation, SimDriveline stops with an error.

Dialog Box and Parameters
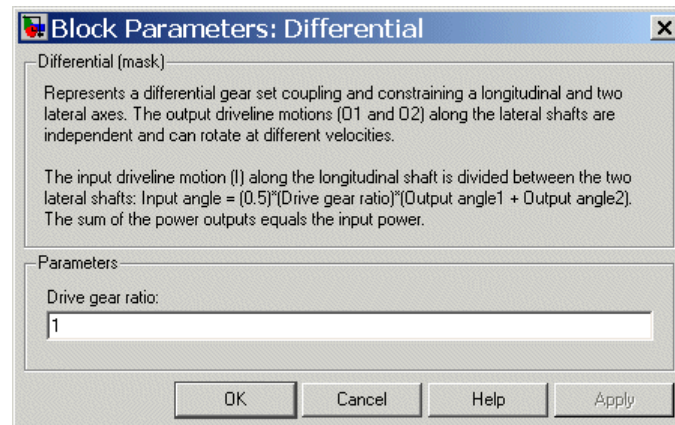
Drive gear ratio
> Ratio g_D is twice the ratio of the input angular motion to the sum of the two output angular motions. This ratio must be strictly positive. The default is 1.

Examples
The demo model `drive_4wd_dynamics` combines two differentials with four tire-wheel assemblies to model the contact of tires with the road and the longitudinal vehicle motion.

See Also
`Controllable Friction Clutch`, `Longitudinal Vehicle Dynamics`, `Tire`

Driveline Environment

Purpose   Represent the SimDriveline environment

Library   Solver & Inertias

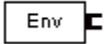**Description**   Each driveline machine represented by a connected SimDriveline block diagram requires global environment information for simulation. The Driveline Environment block specifies this global information and connects the solver that your model needs before you can begin simulation.

Each topologically distinct driveline block diagram requires exactly one Driveline Environment block to be connected to it.

Clutch Modes and Mode Iteration

The Driveline Environment block also controls how the Controllable Friction Clutches of your model lock and unlock during simulation. Once a clutch locks and switches from kinetic to static friction or unlocks and switches from static to kinetic friction, it can unlock only under certain conditions that are tested by *mode iteration*.

- With mode iteration turned on, SimDriveline suspends the simulation in time and enters an algebraic loop to check the conditions for locking or unlocking until it reaches global consistency across the model. This is the default.

- With mode iteration turned off, SimDriveline checks for locking and unlocking while it continues to simulate in time. Turning off mode iteration improves your simulation performance, but might degrade its accuracy.

Mode Iteration and Code Generation

In the default simulation mode, SimDriveline simulates clutch mode changes with mode iteration turned on. However, in the generated code versions of SimDriveline models, mode iteration is turned off automatically. Clutch locking and unlocking are determined over multiple time steps.

Specifying Clutch Velocity Tolerances in a Driveline

A Driveline Environment block can control the velocity tolerance settings for all the clutches in the driveline connected to it. It specifies these settings by default if you do not specify overriding settings in individual clutches.

Apart from overriding individual clutch settings, the velocity tolerance settings act on the entire driveline differently depending on the type of solver you choose.

- If you choose a variable-step solver, you can require non-overriding clutches either to

  - Compute the velocity tolerance value automatically from solver settings (the default configuration of the Driveline Environment block).

  - Use the velocity tolerance value you enter here.

- If you choose a fixed-step solver, you must specify a velocity tolerance value. In those clutches without overriding individual values, SimDriveline always uses the velocity tolerance value you enter here.

Driveline Environment

Block Parameters: Driveline Environment

─ Driveline Environment (mask) ───

Provides the simulation environment for a connected driveline block diagram. Each separate diagram requires exactly one connected Driveline Environment block.

You can disable clutch mode iteration in the connected driveline with the check box provided.

To specify a default velocity tolerance for the clutches in the connected driveline, use the check box and field provided here. If you use a variable-step solver and you select the automatic default tolerance check box, the default tolerance is automatically computed instead from solver settings. If you use a fixed-step solver, the default tolerance specified here is always used whether or not the automatic default tolerance check box is selected. You can override these settings in individual clutches.

─ Parameters ───

☐ Disable mode iteration for clutch locking

☑ Use automatic default clutch velocity tolerance for variable-step solvers (instead of the value below)

Default clutch velocity tolerance (rad/s)

1e-3

| OK | Cancel | Help | Apply |

Disable mode iteration for clutch locking

Controls the mode iteration for locking and unlocking of Controllable Friction Clutches in your driveline model. Select the check box to disable mode iteration. The default is unselected.

Use automatic default clutch velocity tolerance for variable-step solvers

Requires clutches in the connected driveline without individual overrides to compute a velocity tolerance based on solver settings. Valid only if you use a variable-step solver. The default is selected.

Default clutch velocity tolerance

For clutches without individual override settings, the default velocity tolerance value, in radians/second (rad/s). This value must be strictly positive. The default value is 1e-3.

This setting is valid if

- You use a variable-step solver and unselect the **Use automatic default clutch velocity tolerance for variable-step solvers** check box.

- You use a fixed-step solver, regardless of the **Use automatic default clutch velocity tolerance for variable-step solvers** check box setting.

See Also Controllable Friction Clutch, Shared Environment

Dual-Ratio Planetary

Purpose   Represent a set of carrier, sun, planet, and ring gear wheels with specified ring-planet and planet-sun gear ratios

Library   Gears

Description   The Dual-Ratio Planetary block represents a set of carrier, sun, planet, and ring gear wheels. The planet is a single gear wheel with two different radii meshing with the ring and the sun, respectively. The ring and planet corotate with one fixed gear ratio. The planet and sun corotate with another fixed gear ratio.
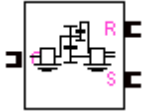
To model the planet's rotational inertia, connect an Inertia block to the optional planet connector port.

Axis Motions and Constraints

The Dual-Ratio Planetary block imposes two kinematic and two geometric constraints on the three connected axes and the fourth, internal wheel (planet):

$$r_C \omega_C = r_S \omega_S + r_{P1} \omega_P \ , \ r_C = r_S + r_{P1}$$

$$r_R \omega_R = r_C \omega_C + r_{P2} \omega_P \ , \ r_R = r_C + r_{P2}$$

In terms of the ring-to-planet gear ratio $g_{RP} = r_R/r_{P2}$ and the planet-to-sun gear ratio $g_{PS} = r_{P1}/r_S$, the key kinematic constraint is

$$(1 + g_{RP} \cdot g_{PS}) \omega_C = \omega_S + g_{RP} \cdot g_{PS} \cdot \omega_R$$

The four degrees of freedom are reduced to two independent degrees of freedom.

The gear ratios are also the ratios of the number of teeth on each gear and the ratios of the torques in each axis, $g_{RP} = N_R/N_{P2} = \tau_R/\tau_{P2}$ and $g_{PS} = N_{P1}/N_S = \tau_{P1}/\tau_S$.

Caution   All gear ratios must be strictly positive. If any gear ratio equals 0 or becomes negative at any time during a simulation, SimDriveline stops with an error.

The gear ratio g_{RP} must be strictly greater than one.
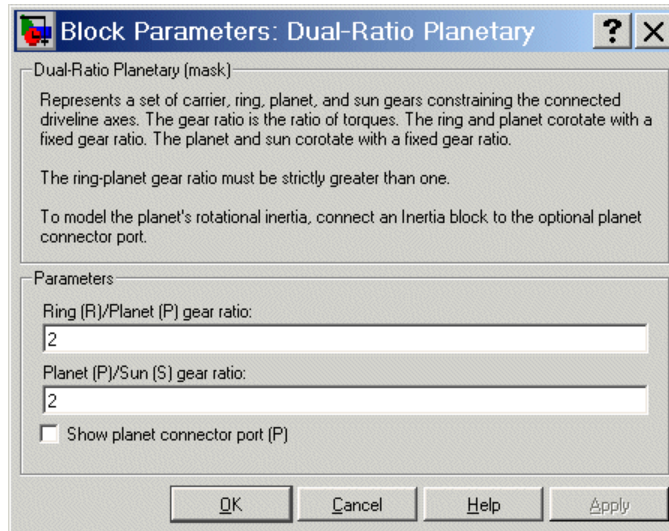
Dialog Box and Parameters

Ring (R)/Planet (P) gear ratio

> Ratio g_{RP} of the ring gear wheel radius to the planet gear wheel radius. This ratio must be strictly greater than 1. The default is 2.

Planet (P)/Sun (S) gear ratio

> Ratio g_{PS} of the planet gear wheel radius to the sun gear wheel radius. This ratio must be strictly positive. The default is 2.

Show planet connector port (P)

> Selecting this check box makes the connector port for the planet gear visible and available for connection to other driveline blocks.

Dual-Ratio Planetary

Use this connector port to connect an Inertia block if you want to model the planet gear's inertia. The default is unselected, with the planet gear's inertia neglected in the dynamics.
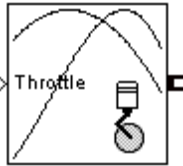
See Also   `Planet-Planet`, `Planetary Gear`, `Ring-Planet`

Purpose

Model a gasoline fuel engine with throttle control, speed governor, and driveline output

Library

Vehicle Components

Description

The Gasoline Engine block models a gasoline-fuel, spark-ignition engine with a speed governor. The engine runs at a variable speed that you can control with a Simulink throttle signal. The throttle signal directly controls the output torque that the engine generates and indirectly controls the speed at which the engine runs. If the engine speed exceeds the maximum speed that you specify, the engine torque drops to zero. The model does not include the air-fuel dynamics of combustion.

The block accepts the throttle signal through a Simulink inport. This signal specifies the engine torque as a fraction of the maximum torque possible in a steady state at a fixed engine speed and should lie between 0 and 1. A throttle signal below zero is interpreted as zero; above one, as one.

To prevent engine power and torque from becoming negative, the block imposes an upper limit on the maximum engine speed itself. If the maximum speed you specify exceeds this limit, SimDriveline issues an error that indicates what the upper limit is.

Using Vehicle Component Blocks

Use the blocks of the Vehicle Components library as a starting point for vehicle modeling. To see how a Vehicle Component block models a driveline component, look under the block mask. The blocks of this library serve as suggestions for developing variant or entirely new models to simulate the same components. Break the block's library link before modifying it and creating your own version.

Gasoline Engine

Dialog Box and Parameters
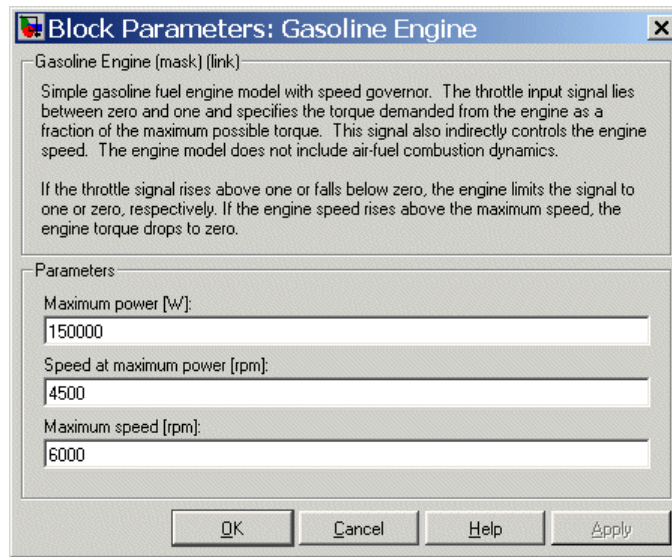
Maximum power

Maximum power that the engine can output, in watts (W). The default is 150000.

Speed at maximum power

Engine speed, in revolutions per minute (rpm), when the engine is running at maximum power. The default is 4500.

Maximum speed

Maximum speed, in revolutions per minute (rpm), at which the engine can turn. The default is 6000.

During simulation, if the engine speed exceeds this maximum, the engine torque drops to zero.

Engine Model

The engine model uses a programmed relationship between torque and speed, modulated by the throttle signal.

Engine Speed, Torque, and Throttle

The engine model is specified by an *engine torque demand* function $g(\Omega)$ built into the block. It provides the maximum torque available for a given engine speed Ω. The block dialog entries (maximum power, speed at maximum power, and maximum speed) normalize this function to physical maximum torque and speed values.

The throttle input signal T specifies the actual engine torque delivered as a fraction of the maximum torque possible in a steady state at a fixed engine speed. It modulates the actual torque delivered τ from the engine: $\tau = T \cdot g(\Omega)$. The actual engine drive shaft speed Ω is fed back to the engine input.

Engine Power and Torque Demand

The demand function $g(\Omega)$ is specified in terms of the steady-state engine power $P(\Omega)$.

The engine speed is limited to a maximum: $0 \leq \Omega \leq \Omega_{max}$. The absolute maximum engine power P_{max} defines Ω_0 such that $P_{max} = P(\Omega_0)$. Define $w = \Omega/\Omega_0$ and $P(\Omega) = P_{max} \cdot p(w)$. Then $p(1) = 1$ and $dp(1)/dw = 0$. Power is the product of torque and angular velocity. The torque demand function is thus

$$\tau_{max} = g(w) = (P_{max}/\Omega_0) \cdot [p(w)/w]$$

You can derive forms for $p(w)$ from engine data and models.
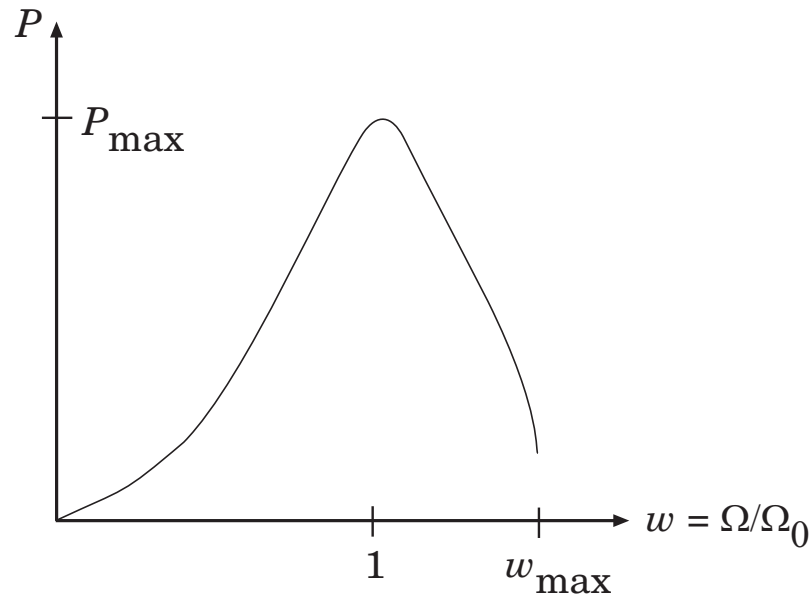
SimDriveline uses a polynomial form for $P(\Omega)$:

$$p(w) = p_1 w + p_2 w^2 - p_3 w^3$$

satisfying

$$p_1 + p_2 - p_3 = 1 \text{ and } p_1 + 2p_2 - 3p_3 = 0$$

Gasoline Engine

The plot shows a curve rising from the origin, peaking at P_{max} near $w = 1$, and falling off toward w_{max}. The vertical axis is labeled P with a tick at P_{max}. The horizontal axis is labeled $w = \Omega/\Omega_0$ with ticks at 1 and w_{max}.

Typical Engine Power Demand Function

Examples   These SimDriveline demo models make use of gasoline engines to power their drivelines:

- drive_full_car

- drive_4wd_dynamics

- drive_vehicle

See Also   Controllable Friction Clutch, Diesel Engine, Torque Converter
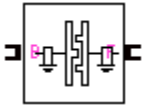
Purpose   Model the restriction on the relative angular motion of two driveline axes to a free gap with elastic upper and lower limits

Library   Dynamic Elements

Description   The Hard Stop block simulates a two-position rotational stop that restricts the relative angular displacement θ of the two connected driveline axes.

- If the relative displacement falls in the gap between the stop's upper and lower limits, the stop applies no torque.

- If the relative displacement becomes greater than the upper limit θ_+ or smaller than the lower limit θ_-, the stop applies a torque.

At each limit, the stop imposes a one-sided damped, linear torsional torque limiting the motion of θ.

The relative angular displacement is the difference of the follower and base driveline axis angles, $\theta = \theta_F - \theta_B$, and the relative angular velocity $\omega = d\theta/dt = \omega_F - \omega_B$. If the angular displacement reaches beyond one of the stop limits, the torque applied is a sum of restoring and damping terms,

$$\tau = -k \cdot (\theta - \theta_\pm) - b\omega$$

where k is the contact stiffness, b the contact damping, and $\theta_\pm$ is the upper limit θ_+ or the lower limit θ_-. Both constants k and b must be nonnegative. The restoring torque depends only on the deformation angle $\theta - \theta_\pm$, measuring how far the displacement has penetrated beyond the upper or lower limit.

Relationship to Restitution

A restitution description of impact specifies the ratio of postimpact and preimpact velocities.

The effective inertias attached to the base and follower axes are I_B and I_F, respectively. The reduced inertia for the relative motion is

Hard Stop

$$I = I_B I_F / (I_B + I_F)$$

Let the relative motion begin penetration of a one-sided stop limit at time t_i and complete its bounce at time t_f. The damped, linear springy torque reduces the final angular velocity, compared to the initial, by the ratio

$$\left| \frac{\dot{\theta}(t_f)}{\dot{\theta}(t_i)} \right| = \sqrt{1 - \frac{2b}{I} \cdot \int_i^f dt \cdot \left[\frac{\dot{\theta}(t)}{\dot{\theta}(t_i)} \right]^2}$$

Using Dynamic Element Blocks

Use the blocks of the Dynamic Elements library as a starting point for vehicle modeling. To see how a Dynamic Element block models a driveline component, look under the block mask. The blocks of this library serve as suggestions for developing variant or entirely new models to simulate the same components. Break the block's library link before modifying it and creating your own version.

Dialog Box and Parameters
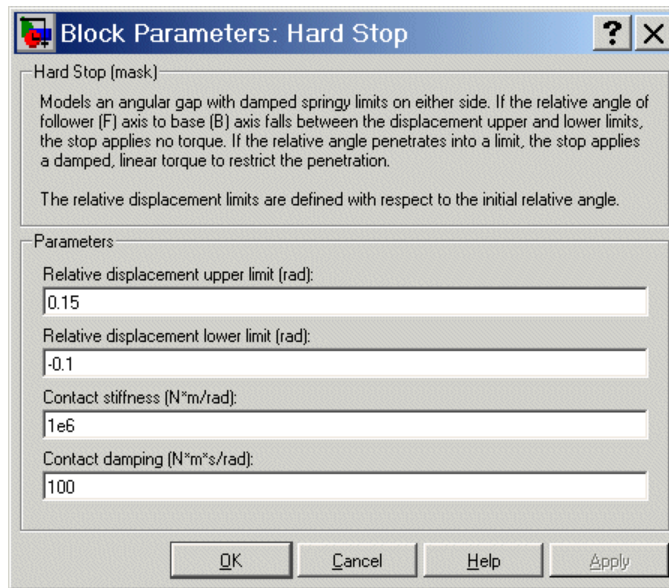
Relative displacement upper limit

> The largest relative displacement angle, in radians (rad), for which the stop does not apply a torque, measured relative to the initial relative angle. Must be larger than the relative displacement lower limit. The default is 0.15.

Relative displacement lower limit

> The smallest relative displacement angle, in radians (rad), for which the stop does not apply a torque, measured relative to the initial relative angle. Must be smaller than the relative displacement upper limit. The default is -0.1.

Contact stiffness

> The linear contact stiffness constant k, in newton-meters/radian (N·m/rad). Must be nonnegative. The default is 1e6.

Contact damping

The linear damping torque constant b, in newton-meter-seconds/radian (N·m·s/rad). Must be nonnegative. The default is 100.

**Examples**   The demo model `drive_hard_stop` simulates angular motion limited by a hard stop.

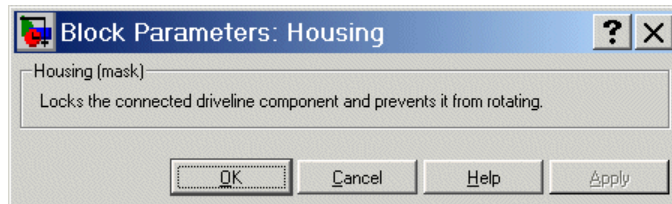**See Also**   `Torsional Spring-Damper`

Purpose   Rotationally lock the connected driveline axis and prevent it from turning

Library   Solver & Inertias

Description   The Housing block prevents any driveline component connected to it from rotating about its driveline axis by locking its motion to zero angular velocity.
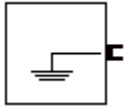
Dialog Box and Parameters

This block has no parameters.

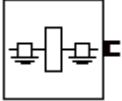**See Also**   Inertia, Shared Environment

Inertia

Purpose   Represent a body with rotational inertia

Library   Solver & Inertias

Description   The Inertia block represents a rigid rotating body. It rotates about the connected driveline axis, which carries the degree of freedom of motion. The body carries a rotational moment of inertia about that axis, which you specify in the block dialog.

The Inertia block has one port. You can connect it to a driveline axis by

- Connecting the port to the end of the axis

- Branching a connection line off the main line and connecting it to the port

---

Caution   Normally, a rigid body in SimDriveline has a positive inertia. You can also enter zero inertia for particular driveline bodies. If you do so, you must ensure that the effective inertia of your entire driveline is positive before actuating it with torques.
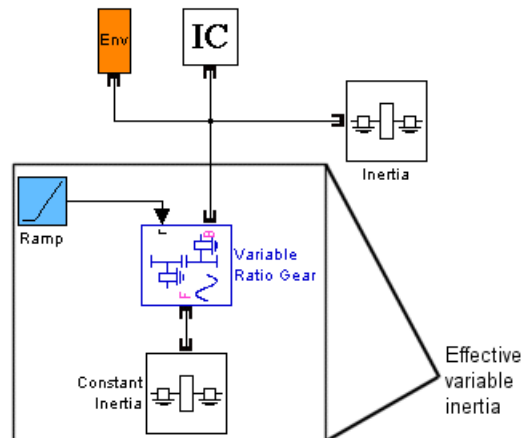

Modeling a Variable Inertia

You cannot vary the inertia value of an Inertia block during a simulation. However, you can model a time-varying inertia indirectly with a Variable Ratio Gear block. This method relies on the conservation of angular momentum.

Place a Variable Ratio Gear between a shaft and an Inertia. Connect this constant Inertia to the Gear's base (B) or follower (F) port. Then vary the gear ratio of the Variable Ratio Gear with an incoming Simulink signal. By changing the gear ratio, you change the effective inertia on the shaft from the constant Inertia.

- Effective inertia = (constant inertia)*(variable gear ratio) if the B port is connected to Inertia

- Effective inertia = (constant inertia)/(variable gear ratio) if the F port is connected to Inertia

Effective Variable Inertia with a Variable Ratio Gear

Dialog Box and Parameters
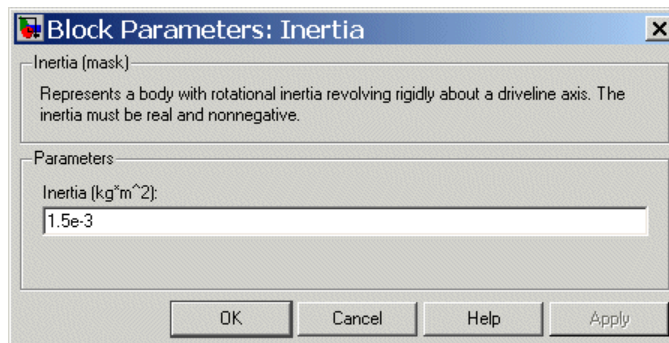
Inertia

Inertia

Rotational moment of inertia of the body represented by the block. Must be a real, nonnegative number or MATLAB expression. The units are kilogram-meters^2 (kg·m^2). The default is `1.5e-3`.

See Also   `Housing`, `Variable Ratio Gear`

Purpose   Set the initial angular velocity of a driveline axis to a nonzero value

Library   Sensors & Actuators

Description   The Initial Condition block connects to a driveline axis and specifies a value for the initial angular velocity of that axis. You specify the initial angular velocity, in radians/second, in the block dialog.
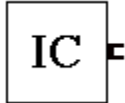
The Initial Condition block has one port. You can connect it to a driveline axis by

- Connecting the port to the end of the axis

- Branching a connection line off the main line and connecting it to the port

Note If you do not connect an Initial Condition block to a driveline axis, SimDriveline assumes by default that the axis starts the simulation with zero angular velocity. You must ensure that the initial angular velocities of your coupled driveline axes are consistent with one another. If they are not, the simulation stops with an error.

Dialog Box and Parameters
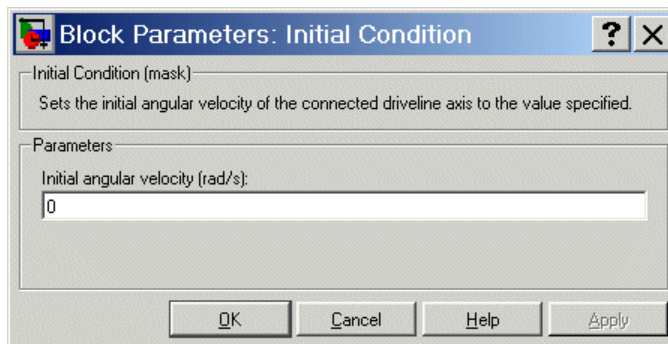
Initial Condition

Initial angular velocity
> The initial angular velocity of the driveline axis connected to this block. The units are radians/second (rad/s). The default is 0.
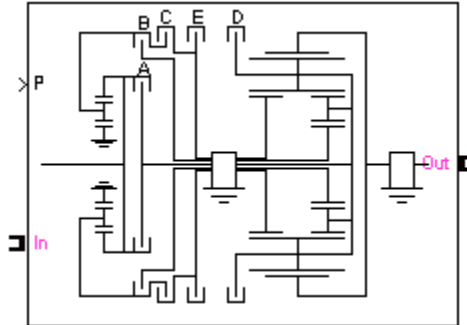
See Also   Housing, Inertia, Motion Actuator

Purpose   Model a six-speed Lepelletier transmission based on a planetary gear
                  and a Ravigneaux gear

Library   Transmission Templates

Description

The Lepelletier 6-Speed transmission block is a subsystem that models
a standard automotive transmission having six selectable forward
gear ratios and a single reverse gear ratio. The Lepelletier gearbox
is constructed by connecting a planetary gear to a Ravigneaux gear.
The sun of the planetary gear is connected to the housing and cannot
rotate. The carrier of the planetary gear is connected, by clutches (A
and C), to the large and small sun wheels of the Ravigneaux gear,
respectively. The input, or driver, shaft is always connected to the
ring of the planetary gear and can simultaneously be connected to the
carrier of the Ravigneaux gear using a separate clutch (B). The output,
or driven, shaft is connected to the ring of the Ravigneaux gear. The five
transmission clutches A, B, C, D, and E are modeled with Controllable
Friction Clutch blocks. You connect the transmission along a driveline
axis, with the In and Out connector ports representing the input and
output shafts, respectively.

This transmission subsystem has two independent internal degrees
of freedom and therefore requires that two clutches be locked at any
instant in order to achieve a unique drive ratio from the input shaft
to the output shaft. The clutch schedule and the corresponding drive

ratios are provided in the block subsystem's clutch schedule table. You disengage this transmission by unlocking all its clutches simultaneously.

Using Transmission Template Subsystems

A Transmission Template block is not library-linked. Once you make a copy in your model, you can use it as is. You can also open and customize it as a subsystem by reconfiguring the properties of the individual Gear, Controllable Friction Clutch, and Inertia blocks.

You must provide a five-component Simulink vector signal of the normalized pressures applied to each clutch. The order of the pressure signals is ABCDE.

Default Inertia, Gear, and Clutch Settings

All the Controllable Friction Clutch blocks in this Transmission subsystem have their default settings. The Gear ratios are reset to nondefault values.

To prevent dynamical singularities, some of the gear wheels have attached Inertia blocks with small default inertias in the 10^{-2} kg-m^2 (kilogram-meters^2) range.

Subsystem Parameters

The gear ratio is the ratio of gear wheel radii r, gear wheel teeth N, or torque transferred τ. The gear ratio is the reciprocal of the ratio of the angular velocities ω transferred. The *drive ratio* is the effective gear ratio, output to input, of the entire transmission.

The basic Lepelletier 6-speed transmission gear ratios are

g_{RS} = Planetary ring/sun gear ratio = $r_{pR}/r_{pS} = N_{pR}/N_{pS}$

g_{RSl} = Ravigneaux ring/large sun gear ratio = $r_R/r_{Sl} = N_R/N_{Sl}$

g_{RSs} = Ravigneaux ring/small sun gear ratio = $r_R/r_{Ss} = N_R/N_{Ss}$

This table specifies the locked (*L*) and free (*F*) clutches A, B, C, D, and E for each gear setting. A free clutch is completely disengaged.

Lepelletier 6-Speed Clutch Schedule

| Gear Setting	Drive Ratio	A	B	C	D	E
Reverse	$-g_{RSl} \cdot (1+g_{RS})/g_{RS}$	L	F	F	L	F
1	$g_{RSs} \cdot (1+g_{RS})/g_{RS}$	F	F	L	L	F
2	$(1+g_{RS})(g_{RSs}+g_{RSl})/[g_{RS} \cdot (1+g_{RSl})]$	F	F	L	F	L
3	$(1 + g_{RS})/g_{RS}$	L	F	L	F	F
4	$g_{RSs} \cdot (1 + g_{RS})/[g_{RSs} \cdot (1 + g_{RS}) - 1]$	F	L	L	F	F
5	$g_{RSl}/[g_{RSl} + 1/(1 + g_{RS})]$	L	L	F	F	F
6	$g_{RSl}/(1 + g_{RSl})$	F	L	F	F	L

See Also   Controllable Friction Clutch, Inertia, Lepelletier 7-Speed, Planetary Gear, Ravigneaux, Ravigneaux 4-Speed, Simpson 4-Speed

Lepelletier 7-Speed

Purpose   Model a seven-speed Lepelletier transmission based on a planetary gear and a Ravigneaux gear

Library   Transmission Templates

Description
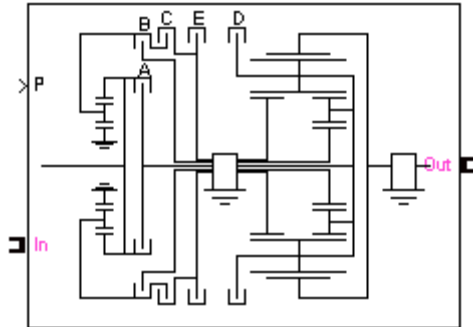
The Lepelletier 7-Speed transmission block is a subsystem that models a standard automotive transmission having seven selectable forward gear ratios and a single reverse gear ratio. The Lepelletier gearbox is constructed by connecting a planetary gear to a Ravigneaux gear. The sun of the planetary gear is connected by a clutch (F) to the housing and can be braked. The carrier of the planetary gear is connected, by another clutch (C), to the small sun wheel of the Ravigneaux gear. The input, or driver, shaft is always connected to the ring of the planetary gear and can simultaneously be connected to the carrier and large sun of the Ravigneaux gear using separate clutches (B and A). The output, or driven, shaft is connected to the ring of the Ravigneaux gear. The six transmission clutches A, B, C, D, E, and F are modeled with Controllable Friction Clutch blocks. You connect the transmission along a driveline axis, with the In and Out connector ports representing the input and output shafts, respectively.

This transmission subsystem has three independent internal degrees of freedom and therefore requires that three clutches be locked at any instant in order to achieve a unique drive ratio from the input shaft to the output shaft. The clutch schedule and the corresponding drive ratios are provided in the block subsystem's clutch schedule

table. You disengage this transmission by unlocking all its clutches simultaneously.

Using Transmission Template Subsystems

A Transmission Template block is not library-linked. Once you make a copy in your model, you can use it as is. You can also open and customize it as a subsystem by reconfiguring the properties of the individual Gear, Controllable Friction Clutch, and Inertia blocks.

You must provide a six-component Simulink vector signal of the normalized pressures applied to each clutch. The order of the pressure signals is ABCDEF.

Default Inertia, Gear, and Clutch Settings

All the Controllable Friction Clutch blocks in this Transmission subsystem have their default settings. The Gear ratios are reset to nondefault values.

To prevent dynamical singularities, some of the gear wheels have attached Inertia blocks with small default inertias in the 10^{-2} kg-m^2 (kilogram-meters^2) range.

Subsystem Parameters

The gear ratio is the ratio of gear wheel radii r, gear wheel teeth N, or torque transferred τ. The gear ratio is the reciprocal of the ratio of the angular velocities ω transferred. The *drive ratio* is the effective gear ratio, output to input, of the entire transmission.

The basic Lepelletier 7-speed transmission gear ratios are

g_{RS} = Planetary ring/sun gear ratio = r_{pR}/r_{pS} = N_{pR}/N_{pS}

g_{RSl} = Ravigneaux ring/large sun gear ratio = r_R/r_{Sl} = N_R/N_{Sl}

g_{RSs} = Ravigneaux ring/small sun gear ratio = r_R/r_{Ss} = N_R/N_{Ss}

This table specifies the locked (*L*) and free (F) clutches A, B, C, D, E, and F for each gear setting. A free clutch is completely disengaged.

Lepelletier 7-Speed

Lepelletier 7-Speed Clutch Schedule

Gear Setting	Drive Ratio	A	B	C	D	E	F
Reverse	$-g_{RSl} \cdot (1 + g_{RS})/g_{RS}$	L	F	F	L	F	L
1	$g_{RSs} \cdot (1 + g_{RS})/g_{RS}$	F	F	L	L	F	L
2	$(g_{RSl} + g_{RSs}) \cdot (1 + g_{RS})/[g_{RS} \cdot (1 + g_{RSl})]$	F	F	L	F	L	L
3	$(1 + g_{RS})/g_{RS}$	L	F	L	F	F	L
4	$g_{RSs} \cdot (1 + g_{RS})/[g_{RSs} \cdot (1 + g_{RS}) - 1]$	F	L	L	F	F	L
5	1	L	L	L	F	F	F
6	$g_{RSl} \cdot (1 + g_{RS})/[g_{RSl} \cdot (1 + g_{RS}) + 1]$	L	L	F	F	F	L
7	$g_{RSl}/(1 + g_{RSl})$	F	L	F	F	L	L

See Also   Controllable Friction Clutch, Inertia, Lepelletier 6-Speed, Planetary Gear, Ravigneaux, Ravigneaux 4-Speed, Simpson 4-Speed
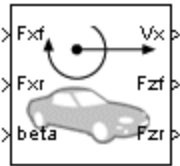
Purpose   Model longitudinal dynamics and motion of a two-axle, four-wheel vehicle

Library   Vehicle Components

Description   The Longitudinal Vehicle Dynamics block models a two-axle vehicle, with four equally sized wheels, moving forward or backward along its longitudinal axis. You specify front and rear longitudinal forces F_{xf}, F_{xr} applied at the front and rear wheel contact points, as well as the incline angle β, as a set of Simulink input signals. The block computes the vehicle velocity V_x and the front and rear vertical load forces F_{zf}, F_{zr} on the vehicle as a set of Simulink output signals. All signals have MKS units.

You must specify the vehicle mass and certain geometric and kinematic details:

- Position of the vehicle's center of gravity (CG) relative to the front and rear axles and to the ground

- Effective frontal cross-sectional area

- Aerodynamic drag coefficient

- Initial longitudinal velocity

See "Vehicle Model" on page 5-53 for details of the vehicle dynamics.

Limitations

The Longitudinal Vehicle Dynamics block lets you model only longitudinal (horizontal) dynamics. Depending on the initial configuration, the block might implement inconsistent initial conditions for the vertical load forces, causing spurious transient dynamics just after the simulation starts.

Using Vehicle Component Blocks

Use the blocks of the Vehicle Components library as a starting point for vehicle modeling. To see how a Vehicle Component block models a driveline component, look under the block mask. The blocks of this

library serve as suggestions for developing variant or entirely new
models to simulate the same components. Break the block's library link
before modifying it and creating your own version.

**Dialog
Box and
Parameters**
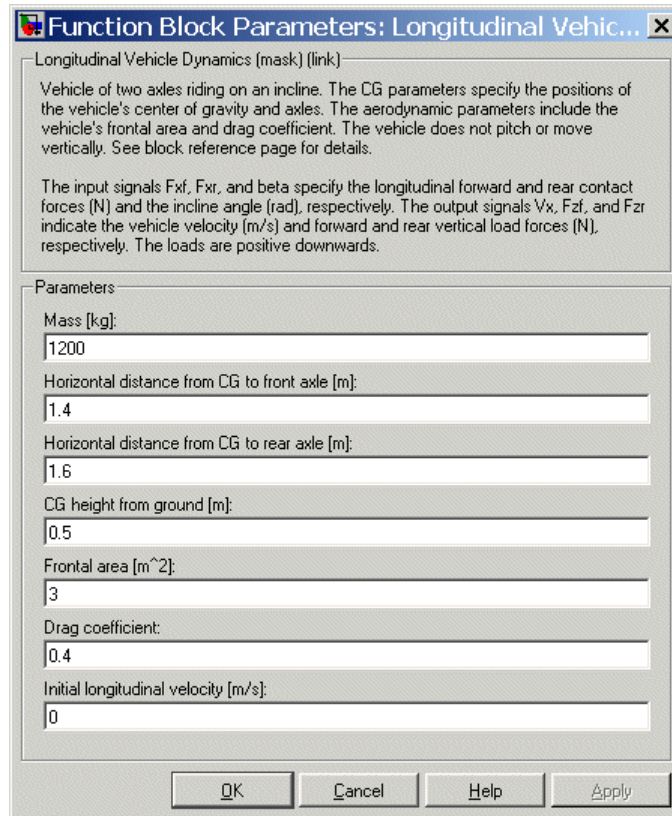
Mass
> Mass m of the vehicle in kilograms (kg). The default is 1200.

Horizontal distance from CG to front axle
> Horizontal distance a, in meters (m), from the vehicle's center of
> gravity to the vehicle's front wheel axle. The default is 1.4.

Horizontal distance from CG to rear axle

Horizontal distance b, in meters (m), from the vehicle's center of gravity to the vehicle's rear wheel axle. The default is `1.6`.

CG height from ground

Height h, in meters (m), of the vehicle's center of gravity from the ground. The default is `0.5`.

Frontal area

Effective cross-sectional area A, in meters squared (m^2), presented by the vehicle in longitudinal motion, for the purpose of computing the aerodynamic drag force on the vehicle. The default is `3`.

Drag coefficient

The dimensionless aerodynamic drag coefficient C_d, for the purpose of computing the aerodynamic drag force on the vehicle. The default is `0.4`.

Initial longitudinal velocity

The initial value of the vehicle's horizontal velocity, in meters/second (m/s). The default is `0`.

Vehicle Model

The vehicle axles are parallel and lie in a plane parallel to the ground. The longitudinal x direction lies in this plane and perpendicular to the axles. If the vehicle is traveling on an incline slope β, the vertical z direction is not parallel to gravity but is always perpendicular to the axle-ground plane.

This figure and table define the vehicle motion model variables.

Longitudinal Vehicle Dynamics
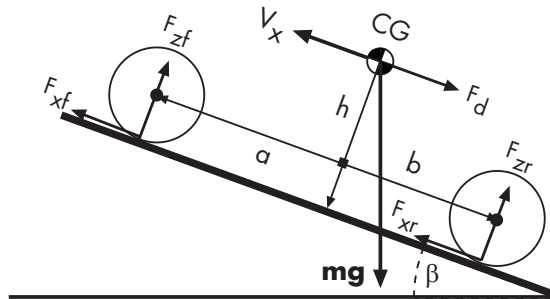
Vehicle Dynamics and Motion

Vehicle Model Variables and Constants

Symbol	Meaning and Unit
$g = -9.81$ m/s^2	Gravitational acceleration (m/s^2)
β	Incline angle (rad)
m	Vehicle mass (kg)
A	Effective frontal vehicle cross-sectional area (m^2)
h	Height of vehicle CG above the ground (m)
a, b	Distance of front and rear axles, respectively, from the vertical projection point of vehicle CG onto the axle-ground plane (m)
V_x	Longitudinal vehicle velocity (m/s)
F_{xf}, F_{xr}	Longitudinal forces on the vehicle at the front and rear wheel ground contact points, respectively (N)
F_{zf}, F_{zr}	Vertical load forces on the vehicle at the front and rear ground contact points, respectively (N)
C_d	Aerodynamic drag coefficient (N·s^2/kg·m)

Symbol	Meaning and Unit
$\rho = 1.2$ kg/m^3	Mass density of air (kg/m^3)
$\lvert F_d \rvert = \frac{1}{2} C_d \rho A V_x{}^2$	Aerodynamic drag force (N)

Vehicle Dynamics and Motion

The vehicle motion is determined by the net effect of all the forces and torques acting on it. The longitudinal tire forces push the vehicle forward or backward. The weight *mg* of the vehicle acts through its center of gravity (CG). Depending on the incline angle, the weight pulls the vehicle to the ground and either pulls it backward or forward. Whether the vehicle travels forward or backward, aerodynamic drag slows it down. For simplicity, the drag is assumed to act through the CG.

$$m\dot{V}_x = F_x + F_d - mg \cdot \sin\beta,$$
$$F_x = F_{xf} + F_{xr},$$
$$F_d = -\frac{1}{2} C_d \rho A V_x{}^2 \cdot \mathrm{sgn}(V_x)$$

Zero vertical acceleration and zero pitch torque require

$$F_{zf} = \frac{+h(F_d - mg\sin\beta - m\dot{V}_x) + b \cdot mg\cos\beta}{a+b}$$
$$F_{zr} = \frac{-h(F_d - mg\sin\beta - m\dot{V}_x) + a \cdot mg\cos\beta}{a+b}$$

Note that $F_{zf} + F_{zr} = mg \cdot \cos\beta$.

Note The Longitudinal Vehicle Dynamics block is implemented with a transfer function that imposes a small delay on the vertical force reaction to changes in the horizontal forces. The vertical and pitch equilibrium holds only on average.

Longitudinal Vehicle Dynamics

Examples   The demo model `drive_4wd_dynamics` combines two differentials with four tire-wheel assemblies to model the contact of tires with the road and the longitudinal vehicle motion.

The demo model `drive_vehicle` models an entire one-wheel vehicle, including Tire and Longitudinal Vehicle Dynamics blocks.

References   Centa, G., *Motor Vehicle Dynamics: Modeling and Simulation,* Singapore, World Scientific, 1997.

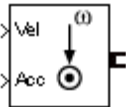Pacejka, H. B., *Tire and Vehicle Dynamics,* Society of Automotive Engineers/Butterworth-Heinemann, Oxford, 2002.

See Also   `Differential`, `Tire`

Purpose   Actuate a driveline axis with specified motions

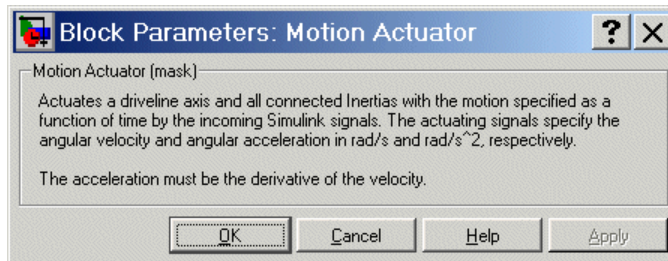**Library**   Sensors & Actuators

Description The Motion Actuator block actuates a driveline axis with specified motions. You specify the motion as velocity and acceleration with a set of two Simulink input signals in radians/second and radians/second^2, respectively. The motion specified is absolute.

The Motion Actuator block has one driveline port. You can connect it to a driveline axis

- By connecting the port to the end of the axis

- By branching a connection line off the main line and connecting it to the port

Dialog Box and Parameters

Block Parameters: Motion Actuator

Motion Actuator (mask)

Actuates a driveline axis and all connected Inertias with the motion specified as a function of time by the incoming Simulink signals. The actuating signals specify the angular velocity and angular acceleration in rad/s and rad/s^2, respectively.

The acceleration must be the derivative of the velocity.

| OK | Cancel | Help | Apply |

This block has no active parameters.

**See Also**   Initial Condition, Motion Sensor, Torque Actuator, Torque Sensor

Motion Sensor

Purpose   Measure the motion of a driveline axis

Library   Sensors & Actuators

Description   The Motion Sensor block senses the motion of a driveline axis. The block can output the motions as a set of three Simulink signals for the angle, angular velocity, and angular acceleration, in radians, radians/second, and radians/second^2, respectively. You can select any combination or all of these output signals. The motion measured is absolute.
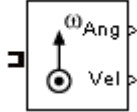
The Motion Sensor block has one driveline port. You can connect it to a driveline axis

- By connecting the port to the end of the axis
- By branching a connection line off the main line and connecting it to the port

Dialog Box and Parameters
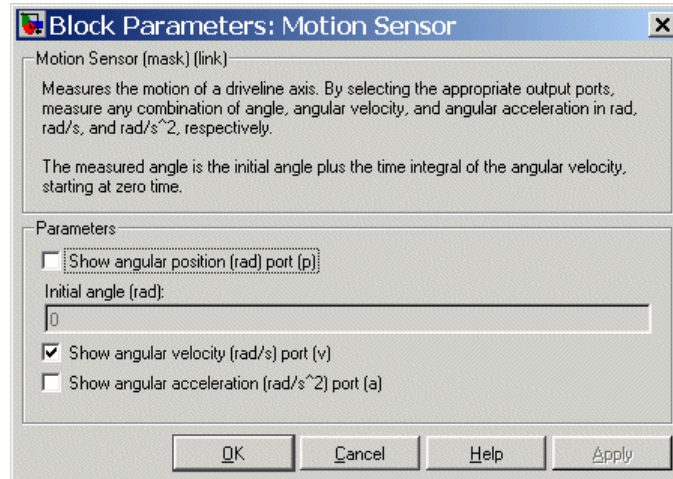
Show angular position port (p)
>Select this check box to create a Simulink outport on the block for the angular motion signal. This signal has units of radians (rad/s). The default is unselected.

Initial angle
>This field is active only if the **Show angular position port** check box is selected. Enter a value for the initial angle of the axis motion, in radians (rad). The default is 0.
>
>The measured angle is this value plus the time integral of the angular velocity, starting at zero time.

Show angular velocity port (v)
>Select this check box to create a Simulink outport on the block for the angular velocity signal. This signal has units of radians/second (rad/s). The default is selected.

Show angular acceleration port (a)
>Select this check box to create a Simulink outport on the block for the angular acceleration signal. This signal has units of radians/second^2 (rad/s^2). The default is unselected.
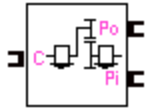
See Also Motion Actuator, Torque Actuator, Torque Sensor

Planet-Planet

Purpose　　　Represent a set of carrier, inner planet, and outer planet gear wheels with specified planet-planet gear ratio

Library　　　Gears

Description　　The Planet-Planet gear block represents a set of carrier, inner planet, and outer planet gear wheels. The outer planet is connected to and rotates with respect to the carrier. The inner planet rotates independently. The planets corotate with a fixed gear ratio that you specify and in opposite directions with respect to the carrier. A planet-planet gear is, along with a ring-planet gear, a basic element of a planetary gear set.

Axis Motions and Constraints

The Planet-Planet block imposes one kinematic and one geometric constraint on the three connected axes:

$$r_C \omega_C = r_{Po} \omega_{Po} + r_{Pi} \omega_{Pi} \;,\; r_C = r_{Po} + r_{Pi}$$

In terms of the outer planet-to-inner planet gear ratio $g_{oi} = r_{Po}/r_{Pi}$, the effective kinematic constraint is

$$(1 + g_{oi})\omega_C = \omega_{Pi} + g_{oi} \cdot \omega_{Po} \;,$$

reducing the three axes to two independent degrees of freedom.

The gear ratio is also the ratio of the number of teeth on each gear and the ratio of the torques in each axis, $g_{oi} = N_{Po}/N_{Pi} = \tau_{Po}/\tau_{Pi}$.

Caution　　All gear ratios must be strictly positive. If any gear ratio equals 0 or becomes negative at any time during a simulation, SimDriveline stops with an error.

**Dialog
Box and
Parameters**
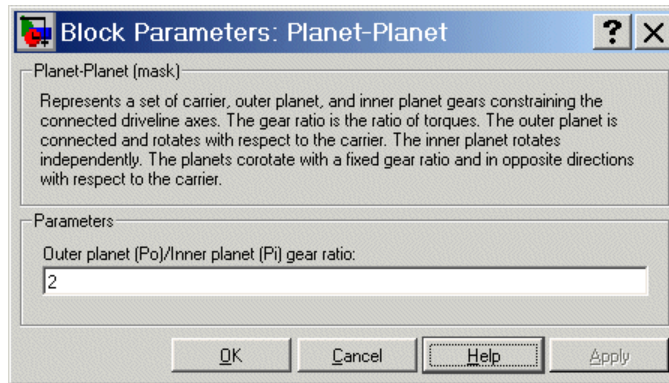
Outer planet (Po)/Inner planet (Pi) gear ratio

Ratio g_{oi} of the outer planet gear radius wheel to the inner planet gear wheel radius. This gear ratio must be strictly positive. The default is 2.
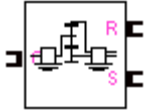
See Also   Planetary Gear, Ring-Planet

Planetary Gear

Purpose   Represent a set of carrier, sun, planet, and ring gear wheels with specified ring-sun gear ratio

Library   Gears

Description   The Planetary Gear block represents a set of carrier, ring, planet, and sun gear wheels. A planetary gear set can be constructed from planet-planet and ring-planet gears. The ring and sun corotate with a fixed gear ratio and in opposite directions with respect to the carrier.

To model the planet's rotational inertia, connect an Inertia block to the optional planet connector port.

Axis Motions and Constraints

The Planetary Gear block imposes two kinematic and two geometric constraints on the three connected axes and the fourth, internal wheel (planet):

$$r_C\omega_C = r_S\omega_S + r_P\omega_P \ , \ r_C = r_S + r_P$$

$$r_R\omega_R = r_C\omega_C + r_P\omega_P \ , \ r_R = r_C + r_P$$

In terms of the ring-to-sun gear ratio $g_{RS} = r_R/r_S$, the key effective kinematic constraint is

$$(1 + g_{RS})\omega_C = \omega_S + g_{RS} \cdot \omega_R$$

The four degrees of freedom are reduced to two independent degrees of freedom.

The gear ratio is also the ratio of the number of teeth on each gear and the ratio of the torques in each axis, $g_{RS} = N_R/N_S = \tau_R/\tau_S$.
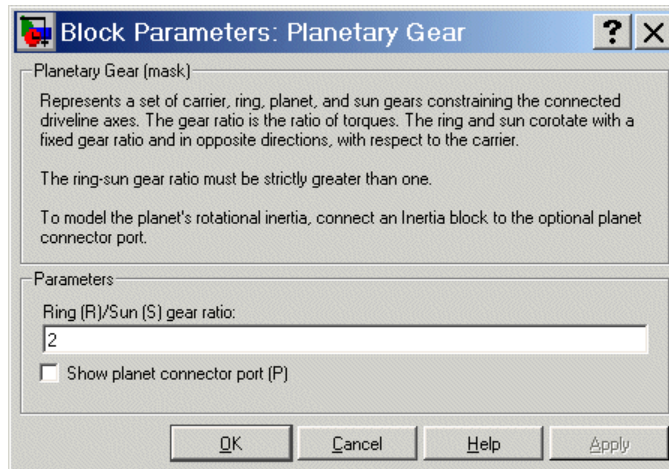
Caution   All gear ratios must be strictly positive. If any gear ratio equals 0 or becomes negative at any time during a simulation, SimDriveline stops with an error.

The gear ratio g_{RS} must be strictly greater than one.

Dialog Box and Parameters

Ring (R)/Sun (S) gear ratio

Ratio g_{RS} of the ring gear wheel radius to the sun gear wheel radius. This gear ratio must be strictly greater than 1. The default is 2.

Show planet connector port (P)

Selecting this check box makes the connector port for the planet gear visible and available for connection to other driveline blocks.

Use this connector port to connect an Inertia block if you want to model the planet gear's inertia. The default is unselected, with the planet gear's inertia neglected in the dynamics.
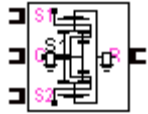
See Also

Dual-Ratio Planetary, Planet-Planet, Ring-Planet

Ravigneaux

Purpose   Represent a Ravigneaux planetary set of carrier, sun, planet, and ring gear wheels with specified ring-sun gear ratios

Library   Gears

Description   The Ravigneaux block represents a double planetary gear set commonly used in automatic transmissions. This planetary gear set is constructed from two gear pairs, ring-planet and planet-planet. The Ravigneaux set has two sun gear wheels, a large sun and a small sun, and a single carrier gear with two independent planetary gear wheels connected to it, an inner planet and an outer planet. The carrier is one wheel but has two radii to couple with the inner and outer planets, respectively. The two planet gears rotate independently of the carrier but corotate with a fixed gear ratio with respect to each other. The inner planet couples with the small sun gear and corotates at a fixed gear ratio with respect to it. The outer planet couples with the large sun gear and corotates with a fixed gear ratio with respect to it. Finally, the ring gear also couples and corotates with the outer planet in a fixed gear ratio with respect to it.

To model the planets' rotational inertia, connect an Inertia block to the optional planet connector port.

Axis Motions and Constraints

The Ravigneaux block imposes four kinematic and four geometric constraints on the four connected axes and the two internal wheels (inner and outer planets):

$$r_{Ci}\omega_C = r_{Ss}\omega_{Ss} + r_{Pi}\omega_{Pi} \, , \; r_{Ci} = r_{Ss} + r_{Pi}$$

$$r_{Co}\omega_C = r_{Sl}\omega_{Sl} + r_{Po}\omega_{Po} \, , \; r_{Co} = r_{Sl} + r_{Po}$$

$$(r_{Co} - r_{Ci})\omega_C = r_{Pi}\omega_{Pi} + r_{Po}\omega_{Po} \, , \; r_{Co} - r_{Ci} = r_{Po} + r_{Pi}$$

$$r_R\omega_R = r_{Co}\omega_C + r_{Po}\omega_{Po} \, , \; r_R = r_{Co} + r_{Po}$$

In terms of the ring-to-small sun gear ratio $g_{RSs} = r_R/r_{Ss}$ and the ring-to-large sun gear ratio $g_{RSl} = r_R/r_{Sl}$, the key kinematic constraints are

$$(g_{RSs} - 1)\omega_C = g_{RSs} \cdot \omega_R - \omega_{Ss}$$

$$(g_{RSl} + 1)\omega_C = g_{RSl} \cdot \omega_R + \omega_{Sl}$$

The six degrees of freedom are reduced to two independent degrees of freedom.

The gear ratios are also the ratios of the number of teeth on each gear and the ratios of torques in each axis, $g_{RSl} = N_R/N_{Sl} = \tau_R/\tau_{Sl}$ and $g_{RSs} = N_R/N_{Ss} = \tau_R/\tau_{Ss}$.
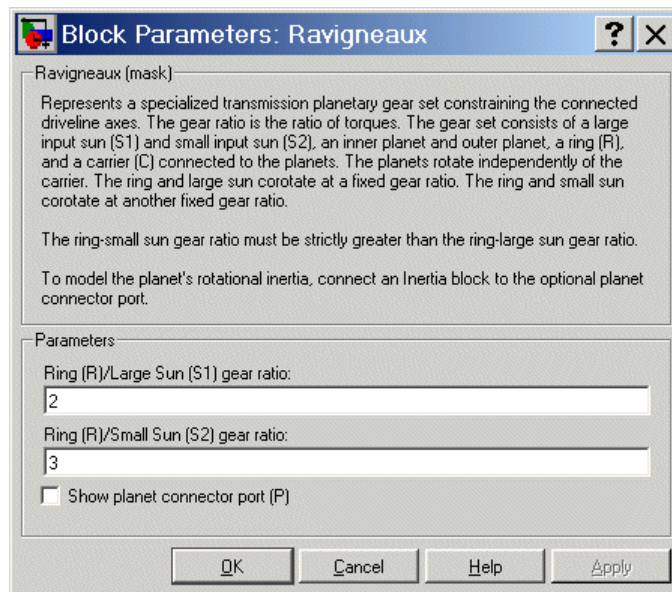

Caution   All gear ratios must be strictly positive. If any gear ratio equals 0 or becomes negative at any time during a simulation, SimDriveline stops with an error.

The gear ratio g_{RSs} must be strictly greater than the gear ratio g_{RSl}.

Ravigneaux

Dialog Box and Parameters

Block Parameters: Ravigneaux

Ravigneaux (mask)

Represents a specialized transmission planetary gear set constraining the connected driveline axes. The gear ratio is the ratio of torques. The gear set consists of a large input sun (S1) and small input sun (S2), an inner planet and outer planet, a ring (R), and a carrier (C) connected to the planets. The planets rotate independently of the carrier. The ring and large sun corotate at a fixed gear ratio. The ring and small sun corotate at another fixed gear ratio.

The ring-small sun gear ratio must be strictly greater than the ring-large sun gear ratio.

To model the planet's rotational inertia, connect an Inertia block to the optional planet connector port.

Parameters

Ring (R)/Large Sun (S1) gear ratio:

2

Ring (R)/Small Sun (S2) gear ratio:

3

☐ Show planet connector port (P)

OK   Cancel   Help   Apply

Ring (R)/Large Sun (S1) gear ratio

Ratio g_{RSl} of the ring gear wheel radius to the large sun gear wheel radius. This gear ratio must be strictly smaller than the ring-small sun gear ratio. The default is 2.

Ring (R)/Small Sun (S2) gear ratio

Ratio g_{RSs} of the ring gear wheel radius to the small sun gear wheel radius. This gear ratio must be strictly greater than the ring-large sun gear ratio. The default is 3.

Show planet connector port (P)

Selecting this check box makes the connector port for the planet gears visible and available for connection to other driveline blocks.

Use this connector port to connect an Inertia block if you want to model the planet gears' inertia as a single body. The default is unselected, with the planet gears' inertia neglected in the dynamics.
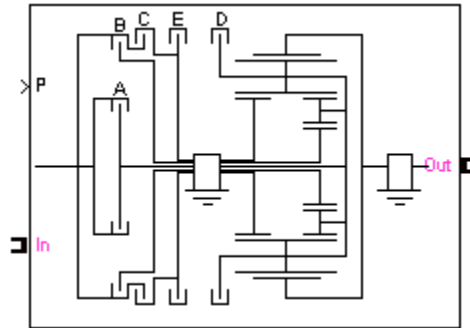
See Also   Dual-Ratio Planetary, Planet-Planet, Planetary Gear,
Ring-Planet, Ravigneaux 4-Speed

Ravigneaux 4-Speed

Purpose　　　Model a Ravigneaux four-speed transmission based on a Ravigneaux gear

Library　　　Transmission Templates

Description

The Ravigneaux 4-Speed transmission block is a subsystem that models a standard automotive transmission having four selectable forward gear ratios and a single reverse gear ratio. The main component of the transmission is a Ravigneaux gear. The input, or driver, shaft is connected, through a combination of the five clutches, to the small sun, the large sun, and the carrier of the Ravigneaux gear. The output, or driven, shaft is connected to the ring of the Ravigneaux gear. The five transmission clutches A, B, C, D, and E are modeled with Controllable Friction Clutch blocks. You connect the transmission along a driveline axis, with the In and Out connector ports representing the input and output shafts, respectively.

This transmission subsystem has two independent degrees of freedom and therefore requires two clutches be locked at any instant in order to achieve a unique drive ratio from the input shaft to the output shaft. The clutch schedule and the corresponding drive ratios are provided in the block subsystem's clutch schedule table. You disengage this transmission by unlocking all its clutches simultaneously.

Using Transmission Template Subsystems

A Transmission Template block is not library-linked. Once you make a copy in your model, you can use it as is. You can also open and customize it as a subsystem by reconfiguring the properties of the individual Gear, Controllable Friction Clutch, and Inertia blocks.

You must provide a five-component Simulink vector signal of the normalized pressures applied to each clutch. The order of the pressure signals is ABCDE.

Default Inertia, Gear, and Clutch Settings

All the Controllable Friction Clutch blocks in this Transmission subsystem have their default settings. The Gear ratios are reset to nondefault values.

To prevent dynamical singularities, some of the gear wheels have attached Inertia blocks with small default inertias in the 10^{-2} kg-m^2 (kilogram-meters^2) range.

Subsystem Parameters

The gear ratio is the ratio of gear wheel radii *r*, gear wheel teeth *N*, or torque transferred τ. The gear ratio is the reciprocal of the ratio of the angular velocities ω transferred. The *drive ratio* is the effective gear ratio, output to input, of the entire transmission.

The basic Ravigneaux 4-speed transmission gear ratios are

$$g_{RSl} = \text{Ring/large sun gear ratio} = r_R/r_{Sl} = N_R/N_{Sl}$$

$$g_{RSs} = \text{Ring/small sun gear ratio} = r_R/r_{Ss} = N_R/N_{Ss}$$

This table specifies the locked (*L*) and free (F) clutches A, B, C, D, and E for each gear setting. A free clutch is completely disengaged.

Ravigneaux 4-Speed

Ravigneaux 4-Speed Clutch Schedule

Gear Setting	Drive Ratio	A	B	C	D	E
Reverse	$-g_{RSl}$	F	F	L	L	F
1	g_{RSs}	F	L	F	L	F
2	$(g_{RSl} + g_{RSs})/(1 + g_{RSl})$	F	L	F	F	L
3	1	L	L	F	F	F
4	$g_{RSl}/(1 + g_{RSl})$	L	F	F	F	L

See Also   Controllable Friction Clutch, Inertia, Lepelletier 6-Speed, Lepelletier 7-Speed, Ravigneaux, Simpson 4-Speed

Purpose   Represent a set of carrier, planet, and ring gear wheels with specified ring-planet gear ratio

Library   Gears
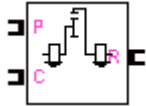
Description   The Ring-Planet gear block represents a set of carrier, planet, and ring gear wheels. The planet is connected to and rotates with respect to the carrier. The planet and ring corotate with a fixed gear ratio that you specify and in the same direction with respect to the carrier. A ring-planet gear is, along with a planet-planet gear, a basic element of a planetary gear set.

Axis Motions and Constraints

The Ring-Planet block imposes one kinematic and one geometric constraint on the three connected axes:

$$r_R\omega_R = r_C\omega_C + r_P\omega_P \ , \ r_R = r_C + r_P$$

In terms of the ring-to-planet gear ratio $g_{RP} = r_R/r_P$, the effective kinematic constraint is

$$g_{RP}{\cdot}\omega_R = \omega_P + (g_{RP} - 1)\omega_C \ ,$$

reducing the three axes to two independent degrees of freedom.

The gear ratio is also the ratio of the number of teeth on each gear and the ratio of the torques in each axis, $g_{RP} = N_R/N_P = \tau_R/\tau_P$.
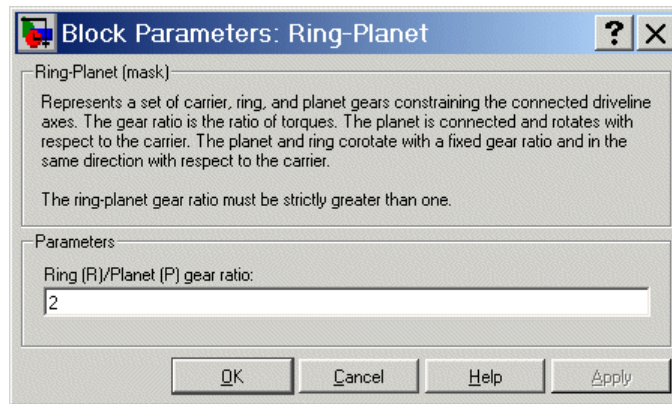

Caution   All gear ratios must be strictly positive. If any gear ratio equals 0 or becomes negative at any time during a simulation, SimDriveline stops with an error.

The ring-planet gear ratio g_{RP} must be strictly greater than one.

Ring-Planet

Dialog Box and Parameters

Block Parameters: Ring-Planet

Ring-Planet (mask)

Represents a set of carrier, ring, and planet gears constraining the connected driveline axes. The gear ratio is the ratio of torques. The planet is connected and rotates with respect to the carrier. The planet and ring corotate with a fixed gear ratio and in the same direction with respect to the carrier.

The ring-planet gear ratio must be strictly greater than one.

Parameters

Ring (R)/Planet (P) gear ratio:

2

OK   Cancel   Help   Apply

Ring (R)/Planet (P) gear ratio
> Ratio g_{RP} of the ring gear wheel radius to the planet gear wheel radius. This gear ratio must be strictly greater than 1. The default value is 2.

See Also   Planet-Planet, Planetary Gear

Purpose   Connect two driveline components so that they share the same driveline environment

Library   Solver & Inertias
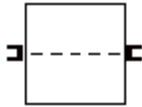
Description   The Shared Environment block provides a nonphysical connection between two independent driveline block diagrams, or subdrivelines. The block carries no inertia, adds no degrees of freedom, imposes no constraints, and transfers no motion or torque between the SimDriveline blocks to which it is connected.

You can use this block to connect two independent drivelines into one driveline, so that the two subdrivelines then share the same driveline environment. Making this connection does not change the structure or dynamics of either subdriveline.

Note The two connected subdrivelines have to be independently valid. The resulting composite driveline needs exactly one Driveline Environment block, not two.

Dialog Box and Parameters

Block Parameters: Shared Environment   ⊠

┌─ Shared Environment (mask) ──────────────────────────────────┐
│ Connects two subdrivelines without adding any inertias, degrees of freedom, or │
│ constraints. The connected subdrivelines rotate independently of each other and must │
│ be independently valid. The resulting composite driveline requires exactly one Driveline │
│ Environment block. │
│   │
│   [OK]   Cancel   Help   Apply   │
└──┘
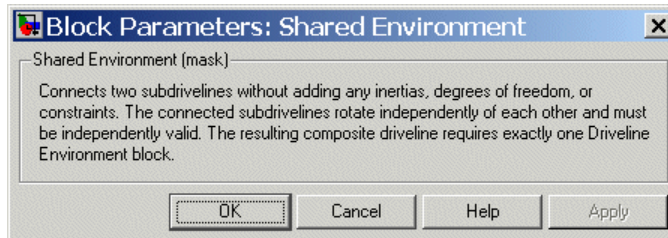
This block has no parameters.

See Also   Driveline Environment, Housing

Simple Gear

Purpose　　Represent a gear with fixed gear ratio

Library　　Gears

Description
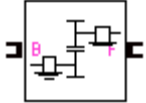
The Simple Gear block represents a gearbox that constrains the two connected driveline axes, base (B) and follower (F), to corotate with a fixed ratio that you specify. You can choose whether the follower axis rotates in the same or opposite direction as the base axis. If they rotate in the same direction, ω_F and ω_B have the same sign. If they rotate in opposite directions, ω_F and ω_B have opposite signs.

Axis Motion and Constraint

The Simple Gear imposes a single constraint, specified by the fixed gear ratio g_{FB}, on the motions and torques of the two axes:

$$\pm g_{FB} = \omega_B/\omega_F = \tau_F/\tau_B$$

The plus and minus signs refer to the axes corotating in the same or opposite directions.

If the Simple Gear represents two coupled gear wheels, the gear ratio is related to the ratio of the radii r of the gear wheels and the ratio of the number N of teeth on each gear wheel:

$$g_{FB} = r_F/r_B = N_F/N_B$$

Caution　　The gear ratio g_{FB} must be strictly positive. If any gear ratio equals 0 or becomes negative at any time during a simulation, SimDriveline stops with an error.

Dialog Box and Parameters
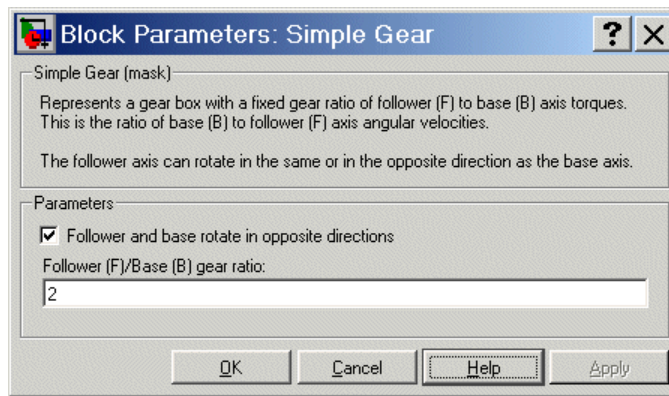
Follower and base rotate in opposite directions

> Select to make the follower and base axes corotate in opposite directions. The default is selected.

Follower (F)/Base (B) gear ratio

> Fixed ratio g_{FB} of the follower axis to the base axis. The gear ratio must be strictly positive. The default is 2.

See Also   Variable Ratio Gear

Simpson 4-Speed

Purpose　　　Model a Simpson four-speed transmission based on two planetary gear sets

Library　　　Transmission Templates

Description
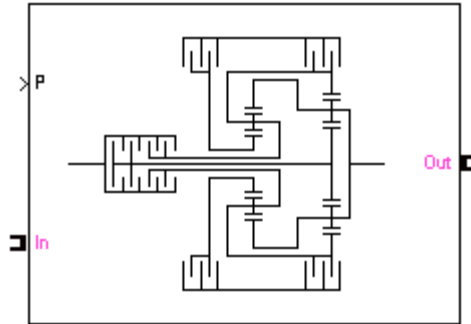
The Simpson 4-Speed transmission block is a subsystem model based on a simple planetary gearbox consisting of two standard planetary gear sets, an input set and an output set, connected together. The ring of the input planetary gear set is connected to the carrier of the output planetary set. Similarly, the ring of the output planetary gear set is connected to the carrier of the input planetary gear set. The input, or driver, shaft is connected either to the carrier of the input planetary gear set or to the sun of the output planetary gear set. The output, or driven, shaft is connected to the ring of the output planetary gear set and to the carrier of the input planetary gear set. The four transmission clutches A, B, C, and D are modeled with Controllable Friction Clutch blocks. You connect the transmission along a driveline axis, with the In and Out connector ports representing the input and output shafts, respectively.

This transmission subsystem has two independent internal degrees of freedom and therefore requires that two clutches be locked at any instant in order to achieve a unique drive ratio from the input shaft to the output shaft. The clutch schedule and the corresponding drive ratios are provided in the block subsystem's clutch schedule table. You disengage this transmission by unlocking all its clutches simultaneously.

Using Transmission Template Subsystems

A Transmission Template block is not library-linked. Once you make a copy in your model, you can use it as is. You can also open and customize it as a subsystem by reconfiguring the properties of the individual Gear, Controllable Friction Clutch, and Inertia blocks.

You must provide a four-component Simulink vector signal of the normalized pressures applied to each clutch. The order of the pressure signals is ABCD.

Default Inertia, Gear, and Clutch Settings

All the Controllable Friction Clutch blocks in this Transmission subsystem have their default settings. The Gear ratios are reset to nondefault values.

To prevent dynamical singularities, some of the gear wheels have attached Inertia blocks with small default inertias in the 10^{-2} kg-m^2 (kilogram-meters^2) range.

Subsystem Parameters

The gear ratio is the ratio of gear wheel radii r, gear wheel teeth N, or torque transferred τ. The gear ratio is the reciprocal of the ratio of the angular velocities ω transferred. The *drive ratio* is the effective gear ratio, output to input, of the entire transmission.

The basic Simpson 4-speed transmission gear ratios are

g_i = Input planetary ring/input planetary sun gear ratio = r_{Ri}/r_{Si} = N_{Ri}/N_{Si}

g_o = Output planetary ring/output planetary sun gear ratio = r_{Ro}/r_{So} = N_{Ro}/N_{So}

This table specifies the locked (L) and free (F) clutches A, B, C, and D for each gear setting. A free clutch is completely disengaged.

Simpson 4-Speed

Simpson 4–Speed Clutch Schedule

Gear Setting	Drive Ratio	A	B	C	D
1	$1 + g_o$	L	F	F	L
2	$1 + g_o/(1 + g_i)$	L	F	L	F
3	1	L	L	F	F
4	$g_i/(1 + g_i)$	F	L	L	F

Examples

These SimDriveline demo models make use of Simpson transmission blocks to transfer drivetrain torque and motion:
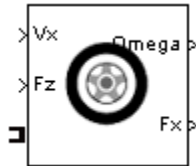
- drive_simpson
- drive_full_car
- drive_vehicle

See Also

Controllable Friction Clutch, Inertia, Lepelletier 6-Speed, Lepelletier 7-Speed, Planetary Gear, Ravigneaux 4-Speed

Purpose   Model tire dynamics and motion at the end of a driveline axis

Library   Vehicle Components

Description   The Tire block models a vehicle tire in contact with the road. The driveline port ◱ transfers the torque from the wheel axis to the tire. You must specify the vertical load F_z and vehicle longitudinal velocity V_x as Simulink input signals. The model provides the tire angular velocity Ω and the longitudinal force F_x on the vehicle as Simulink output signals. All signals have MKS units.

The convention for the F_z signal is positive downward. If the vertical load F_z is zero or negative, the horizontal tire force F_x vanishes. In that case, the tire is just touching or has left the ground.

The longitudinal direction lies along the forward-backward axis of the vehicle. See "Tire Model" on page 5-81 for model details.

Using Vehicle Component Blocks

Use the blocks of the Vehicle Components library as a starting point for vehicle modeling. To see how a Vehicle Component block models a driveline component, look under the block mask. The blocks of this library serve as suggestions for developing variant or entirely new models to simulate the same components. Break the block's library link before modifying it and creating your own version.

Tire

Dialog Box and Parameters
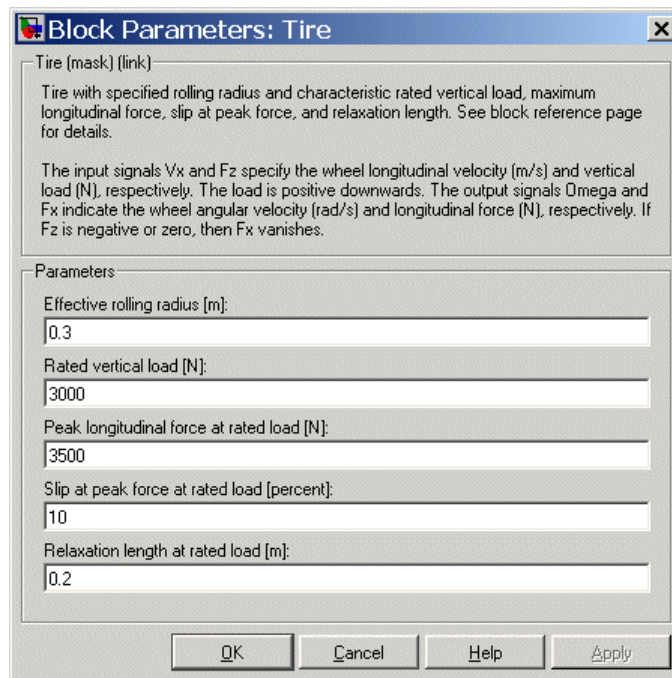
Effective rolling radius
Effective radius r_e, in meters (m), at which the longitudinal force is transferred to the wheel as a torque. The default is 0.3.

Rated vertical load
Rated vertical load force F_z, in newtons (N). The default is 3000.

Peak longitudinal force at rated load
Maximum longitudinal force F_x, in newtons (N), the tire exerts on the wheel when the vertical load equals its rated value. The default is 3500.

Slip at peak force at rated load
The value of the contact slip, in percent (%), when F_x equals its maximum value and F_z equals its rated value. The default is 10.

Relaxation length at rated load

Tire relaxation length σ_κ, in meters (m), that determines the transient response of the tire force F_x at the tire's rated load. The default is `0.2`.

Note For more about the tire parameters, see "Relationship to Block Parameters" on page 5-85.

Tire Model

The tire is a flexible body in contact with the road surface and subject to slip. When a torque is applied to the wheel axle, the tire deforms, pushes on the ground (while subject to contact friction), and transfers the resulting reaction as a force back on the wheel, pushing the wheel forward or backward.

The Tire block models the tire as a rigid-wheel, flexible-body combination in contact with the road. The model includes only longitudinal motion and no camber, turning, or lateral motion. At full speed, the tire acts like a damper, and the longitudinal force F_x is determined mainly by the slip. At low speeds, when the tire is starting up from or slowing down to a stop, the tire behaves more like a deformable, circular spring. The effective rolling radius r_e is normally slightly less than the nominal tire radius because the tire deforms under its vertical load. The tire relaxation length σ_κ is the ratio of the slip stiffness to longitudinal force stiffness. It determines the transient response of F_x to slip.

This figure and table define the tire model variables. The figure displays the forces from the ground on the tire. The normal convention for F_z is positive *downward,* representing the vertical load on the tire and the force from the tire on the ground.

Tire

Tire Dynamics and Motion

Tire Model Variables and Constants

Symbol	Meaning and Unit
r_e	Effective rolling radius (m)
I_w	Wheel-tire assembly inertia (kg·m^2)
τ_{drive}	Torque applied by the axle to the wheel (N·m)
V_x	Wheel center longitudinal velocity (m/s)
Ω	Wheel angular velocity (rad/s)
Ω'	Contact point angular velocity (rad/s)
$V_{sx} = V_x - r_e\Omega$	Wheel slip velocity (m/s)
$V'_{sx} = V_x - r_e\Omega'$	Contact point slip velocity (m/s)
$\kappa = -V_{sx}/|V_x|$	Wheel slip

Symbol	Meaning and Unit
$\kappa' = -V'_{sx}/\|V_x\|$	Contact patch slip
u	Tire longitudinal compliance or deformation (m)
F_z	Vertical load on tire (N)
$F_x = f(\kappa', F_z)$	Longitudinal force exerted by the tire on the wheel at the contact point (N). Also a characteristic function f of the tire.
$C_{Fx} = (\partial F_x/\partial u)_0$	Tire longitudinal stiffness (N/m)
$\sigma_\kappa = (\partial F_x/\partial \kappa')_0/(C_{Fx})$	Tire relaxation length (m)

Tire Deformation and Response

If the tire were rigid and did not slip, it would roll and translate as $V_x = r_e\Omega$. In reality, even a rigid tire slips, and a tire develops a longitudinal force F_x only in response to slip. The wheel slip velocity $V_{sx} = V_x - r_e\Omega \neq 0$. The *wheel slip* $\kappa = -V_{sx}/|V_x|$ is more convenient. For a locked, sliding tire, $\kappa = -1$. For perfect rolling, $\kappa = 0$.

The tire is also flexible. Because it deforms, the contact point turns at a slightly different angular velocity Ω' from the wheel. The *contact point slip* $\kappa' = -V'_{sx}/|V_x|$, where $V'_{sx} = V_x - r_e\Omega'$.

The *tire deformation u* directly measures the difference of wheel and contact point slip and satisfies

$$\frac{du}{dt} = V'_{sx} - V_{sx}$$

A tire model must provide the longitudinal force F_x the tire exerts on the wheel once given

Tire

- Vertical load F_z

- Contact slip κ'

The *tire characteristic function* specifies this relationship in the steady state: $F_x = f(\kappa', F_z)$.

The contact slip κ' in turn depends on the deformation u. The longitudinal force F_x is approximately proportional to the vertical load because F_x is generated by contact friction and the normal force F_z. (The relationship is somewhat nonlinear because of tire deformation and slip.) The dependence of F_x on κ' is more complex.

Tire Dynamics

The tire model incorporates transient as well as steady-state behavior and is thus appropriate for starting from, and coming to, a stop.

Because the rolling, stressed tire is not in a steady state, the contact slip κ' and deformation u are not constant. Before they can be used in the characteristic function, their time evolution must be accounted for. In this model, u and κ' are moderate to small. The relationships of F_x to u and u to κ' are then linear:

$$F_x = C_{Fx} \cdot u \ , \ C_{Fx} = \left(\frac{\partial F_x}{\partial u} \right)_{u=0}$$

$$u = \sigma_\kappa \cdot \kappa' \ , \ \sigma_\kappa = \left(\frac{\partial F_x}{\partial \kappa'} \right)_{\kappa'=0} \Big/ \left(\frac{\partial F_x}{\partial u} \right)_{u=0} = \left(\frac{\partial F_x}{\partial \kappa'} \right)_{\kappa'=0} \Big/ C_{Fx}$$

These properties are taken from empirical tire data.

The deformation u evolves according to

$$\frac{du}{dt} + \left(\frac{1}{\sigma\kappa} \right) \cdot |V_x| \, u = -V_{sx}$$

The slip κ' follows from σ_κ and u. The tire behaves like a driven damper of damping rate $|V_x|/\sigma_\kappa$.

At low speeds, the slip remains finite, and the tire behaves more like a circular spring of stiffness C_{Fx}. Manifestly nonsingular forms of the tire evolution are

$$\left(\frac{1}{C_{Fx}}\right) \cdot \frac{dF_x}{dt} + |V_x| \, \kappa' = -V_{sx} \, ,$$

$$\left(\frac{1}{C_{Fx}}\right) \cdot \frac{\partial F_x}{\partial \kappa'} \frac{d\kappa'}{dt} + |V_x| \, \kappa' = -V_{sx} - \left(\frac{1}{C_{Fx}}\right) \cdot \frac{\partial F_x}{\partial F_z} \frac{dF_z}{dt}$$

The second form explicitly shows the dependence on a varying vertical load F_z.

Finding the Wheel and Vehicle Motion

With the tire characteristic function $f(\kappa', F_z)$, the vertical load F_z, and the evolved u and κ', you can find the longitudinal force F_x and wheel velocity Ω. From these, the equations of motion determine the wheel angular motion (the angular velocity Ω) and longitudinal motion (the wheel center velocity V_x):

$$I_w \frac{d\Omega}{dt} = \tau_{\text{drive}} - r_e F_x$$

$$m \frac{dV_x}{dt} = F_x - mg \cdot \sin\beta$$

where β is the slope of the incline upon which the vehicle is traveling (positive for uphill), and m and g are the wheel load mass and the gravitational acceleration, respectively. τ_{drive} is the driveshaft torque applied to the wheel axis.

Relationship to Block Parameters

The effective rolling radius is r_e. The rated load normalizes the tire characteristic function $f(\kappa', F_z)$, and the peak force, slip at peak force, and relaxation length fields determine the peak and slope of $f(\kappa', F_z)$ and thus C_{Fx} and σ_κ.

Tire

Examples   The demo model `drive_4wd_dynamics` combines two differentials with
four tire-wheel assemblies to model the contact of tires with the road
and the longitudinal vehicle motion.

The demo model `drive_vehicle` models an entire one-wheel vehicle,
including Tire and Longitudinal Vehicle Dynamics blocks.

References   Centa, G., *Motor Vehicle Dynamics: Modeling and Simulation,*
Singapore, World Scientific, 1997.

Pacejka, H. B., *Tire and Vehicle Dynamics,* Society of Automotive
Engineers/Butterworth-Heinemann, Oxford, 2002.

See Also   `Differential, Longitudinal Vehicle Dynamics`

Purpose   Actuate a driveline axis with a specified torque

Library   Sensors & Actuators
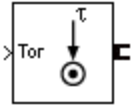
Description   The Torque Actuator block actuates the connected driveline axis with a torque. You specify this torque as a Simulink input signal in newton-meters.

The Torque Actuator block has one driveline port. You can connect it to a driveline axis

- By connecting the port to the end of the axis
- By branching a connection line off the main line and connecting it to the port

Dialog Box and Parameters

Block Parameters: Torque Actuator

Torque Actuator (mask)

Actuates a driveline axis and all connected Inertias with the torque specified by the incoming Simulink signal in N*m.

OK   Cancel   Help   Apply
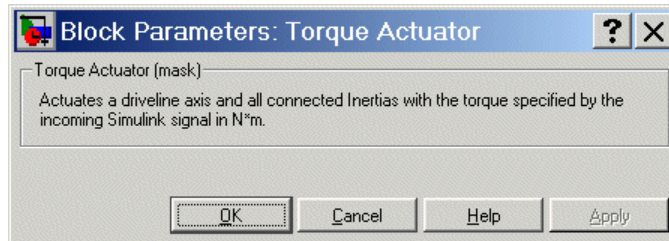
This block has no active parameters.

See Also   Motion Actuator, Motion Sensor, Torque Sensor

Torque Converter

Purpose

Transfer torque between two driveline axes as a function of their relative angular velocity

Library

Dynamic Elements

Description
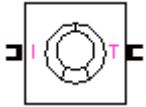
A torque converter couples two driveline axes, transferring torque and angular motion by the hydrodynamic action of a viscous fluid. Unlike a friction clutch, it cannot lock the axes together. The Torque Converter block models a torque converter acting between the two connector ports I and T as a function of the relative angular velocity of the two connected driveline axes. The input is the connector port into which power flows into the block. The output is the connector port from which power flows out of the block. The I port represents the impeller or pump. The T port represents the turbine. Forward power flow means power flowing from I to T. Reverse power flow means power flowing from T to I. Forward motion means the relative angular velocity $\omega = \omega_T - \omega_I > 0$.

Because the coupling of the axes occurs by viscous action, the torque transfer depends on this difference ω. In normal operation, the two axes have different speeds, and the output T axis speed never exactly reaches the input I axis speed ($\omega < 0$). The torque transfer is largest when $|\omega|$ is large and shrinks as $|\omega|$ shrinks. Because $|\omega|$ can never reach zero exactly, a torque converter always transfers some torque.

Speed Ratio, Torque Ratio, and Capacity Factor

You specify the torque ratio and the capacity factor of the torque converter as discrete functions of the speed ratio with tabular vector entries. The three vectors of the independent and two dependent variable values must have the same length.

- The speed ratio R_ω is the output angular speed divided by the input angular speed. You specify a range of speed ratio values from 0 up to, but not including, 1.

$$R_\omega = \min[\omega_I/\omega_T, \, \omega_T/\omega_I]$$

- The torque ratio R_τ is the output torque divided by the input torque.

  $$R_\tau = \tau_{output} / \tau_{input}$$

- The capacity factor K is the input speed divided by the square root of the input torque.

  $$K = \max[\omega_I, \omega_T] / \sqrt{\tau_{input}}$$

τ_{input} is the torque flowing into the shaft with the larger speed, and τ_{output} is the torque flowing into the shaft with the smaller speed.

Using Dynamic Element Blocks

Use the blocks of the Dynamic Elements library as a starting point for vehicle modeling. To see how a Dynamic Element block models a driveline component, look under the block mask. The blocks of this library serve as suggestions for developing variant or entirely new models to simulate the same components. Break the block's library link before modifying it and creating your own version.

Torque Converter

Dialog Box and Parameters
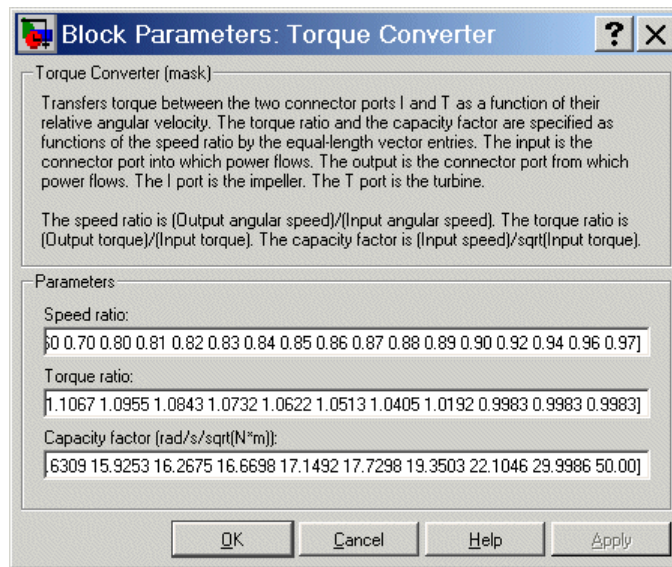
Speed ratio

Vector of values of the block function's independent variable, the dimensionless speed ratio. These values must be greater than or equal to 0 and strictly less than 1.

Torque ratio

Vector of values of the block function's first dependent variable, the dimensionless torque ratio. Each torque ratio value corresponds to a speed ratio value.

Capacity factor

Vector of values of the block function's second dependent variable, the torque conversion capacity factor. Each capacity factor value corresponds to a speed ratio value. The units are radians/second/√(newton-meters).

Torque Converter Model

Two functions specify the characteristics of the torque converter: the torque ratio R_τ and the capacity factor K, both as functions of the speed ratio R_ω. You specify these as discrete tabulated functions in the dialog.

$$R_\tau = R_\tau(R_\omega)$$

$$K = K(R_\omega)$$

In normal operation (forward power flow), the input impeller (I) and output turbine (T) torques are

$$\tau_I = \text{sgn}(1 - \omega_T / \omega_I) \cdot [\omega_I / K]^2$$

$$\tau_T = \tau_I \cdot R_\tau$$

Examples

The demo model `drive_torque_convert` simulates a torque converter.

These SimDriveline demo models include torque converters as part of complete drivetrains:

- `drive_full_car`
- `drive_vehicle`

References

Society of Automotive Engineers, *Hydrodynamic Drive Test Code (Surface Vehicle Recommended Practice),* SAE J643, May 2000.

See Also

`Controllable Friction Clutch`, `Diesel Engine`, `Gasoline Engine`

Torque Sensor

Purpose   Measure the torque transferred along a driveline axis

Library   Sensors & Actuators

Description   The Torque Sensor block measures the torque transferred along a driveline axis at the point where the Torque Sensor is inserted. A positive torque is transferred from the base (B) axis to the follower (F) axis at that point if the follower axis accelerates positively with respect to the base and if no other torques are applied to the follower-connected inertias. The torque is output as a Simulink signal in newton-meters.
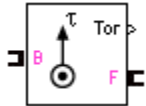
Dialog Box and Parameters
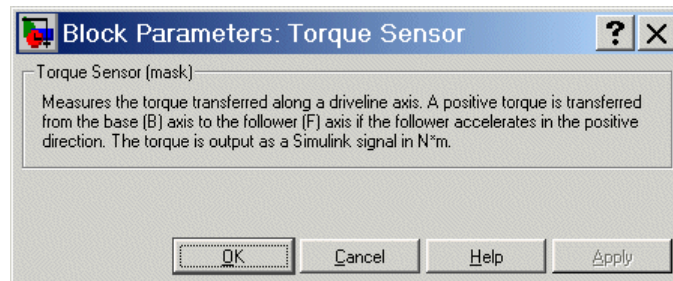
This block has no active parameters.

See Also   Motion Actuator, Motion Sensor, Torque Actuator

Purpose   Represent a damped torsional spring torque, with a free play gap, acting between two rotating axes

Library   Dynamic Elements
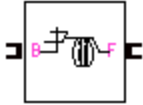
Description The Torsional Spring-Damper block models a damped torsional spring-like torque acting between two rotating axes, the base (B) and the follower (F). This torque is a function of the relative displacement angle $\theta = \theta_F - \theta_B$ and relative angular velocity $\omega = d\theta/dt = \omega_F - \omega_B$.

$$\tau = -k(\,\theta - \theta_{back}) - b\omega, \text{ if } \theta > +\theta_{back}$$

$$\tau = -k(\,\theta + \theta_{back}) - b\omega, \text{ if } \theta < -\theta_{back}$$

$$\tau = 0, \text{ if } -\theta_{back} < \theta < +\theta_{back}$$

You specify the restoring torque spring constant or spring rate k as the stiffness and the kinetic friction torque constant b as the damping. Both constants must be nonnegative.

Backlash and Initial Offset

The backlash interval is a free play gap of size $2\theta_{back}$ across. If θ lies in this range, the spring and damping torques are not applied. Above and below the backlash range, both the spring and damping torques are active.

You specify the initial relative displacement θ_0 as the initial offset, relative to the midway point between the backlash interval endpoints.

Torsional Spring-Damper
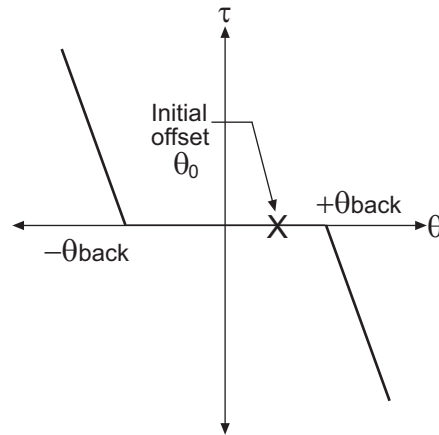
Torsional Spring-Damper Torque Law (Spring Only)

Using Dynamic Element Blocks

Use the blocks of the Dynamic Elements library as a starting point
for vehicle modeling. To see how a Dynamic Element block models a
driveline component, look under the block mask. The blocks of this
library serve as suggestions for developing variant or entirely new
models to simulate the same components. Break the block's library link
before modifying it and creating your own version.

Dialog Box and Parameters

Block Parameters: Torsional Spring-Dam... ? ✕

Torsional Spring-Damper (mask)

Models a damped torsional spring with backlash. The relative motion is the follower (F) axis minus the base (B) axis. The torque is positive if it acts from base to follower.

The stiffness is the spring rate constant. The damping is the kinetic friction constant. If the absolute value of the relative angle is less than the backlash, only the damping torque is applied. If it is greater than the backlash, the full damped spring torque is applied. Only the angular penetration beyond the backlash is used to compute the restoring spring torque.

The backlash range is -backlash to +backlash. The initial offset is defined with respect to the midpoint of this range.

Parameters

Stiffness (N*m/rad):

1e3

Damping (N*m*s/rad):

0.1

Initial Offset (rad):

0

Backlash (rad):

0

[OK] [Cancel] [Help] [Apply]

Stiffness

The spring constant or spring rate k for the restoring torque imposed by the spring. Must be nonnegative. The units are newton-meters/radian (N·m/rad). The default is 1e3.

Damping

The damping constant b for the kinetic frictional torque imposed by the damper. Must be nonnegative. The units are newton-meters-seconds/radian (N·m·s/rad). The default is 0.1.

Initial Offset

The initial angular offset θ_0 of the relative displacement θ, in radians (rad). The default is 0.

Backlash

The angular free play θ_{back} allowed in the torsional spring. Must be nonnegative. The units are radians (rad). The default is 0.

Torsional Spring-Damper

Examples   The demo model `drive_spring` illustrates a simple torsional
                spring-damper system.

See Also   Hard Stop

Purpose   Represent a gear with controllable, variable gear ratio

Library   Gears
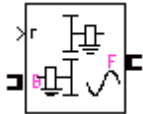
Description The Variable Ratio Gear block represents a gearbox that constrains the two connected driveline axes, base (B) and follower (F), to corotate with a variable ratio that you can control. You can choose whether the follower axis rotates in the same or opposite direction as the base axis. If they rotate in the same direction, ω_F and ω_B have the same sign. If they rotate in opposite directions, ω_F and ω_B have opposite signs.

You specify the variable gear ratio as a function of time with the Simulink input signal r.

Axis Motion and Constraint

The Variable Ratio Gear imposes a single constraint, specified by the variable gear ratio $g_{FB}(t)$, on the motions and torques of the two axes:

$$\pm g_{FB}(t) = \omega_B/\omega_F = \tau_F/\tau_B$$

Caution   The gear ratio g_{FB} must be strictly positive. If any gear ratio equals 0 or becomes negative at any time during a simulation, SimDriveline stops with an error.

Effect of Coriolis Acceleration

With the Variable Ratio Gear block, you can choose to include or not include the effect of Coriolis acceleration on the gear motion. If you choose to include it, you must supply the first derivative, dg_{FB}/dt, of the gear ratio as another Simulink input signal r.

The Coriolis acceleration is a small, nonlinear effect proportional to the angular velocity and the first derivative of the gear ratio:

$$d\omega_B/dt = g_{FB} \cdot d\omega_F/dt + \omega_F \cdot dg_{FB}/dt$$

Variable Ratio Gear

Effect of Linearization

If you simulate your model in linearization mode, SimDriveline holds the variable gear ratio $g_{FB}(t)$ fixed at its initial value, $g_{FB}(0)$. You choose the **Simulation mode** in the Driveline Environment dialog.
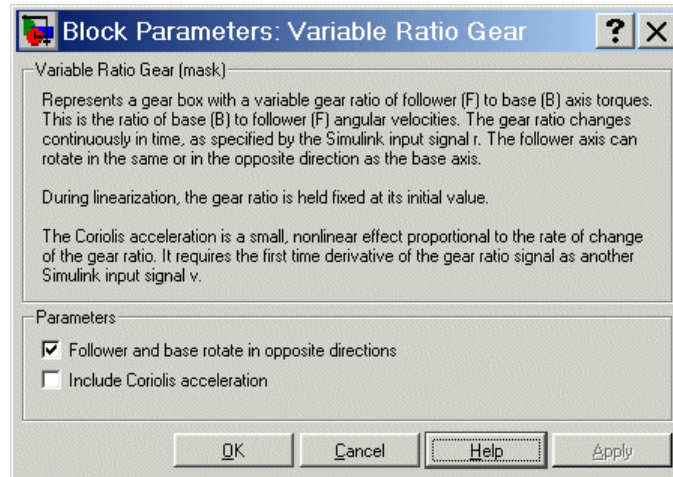
Dialog Box and Parameters

Follower and base rotate in opposite directions
Select to make the follower and base axes corotate in opposite directions. The default is selected.

Include Coriolis acceleration
Select to include the small nonlinear effect of the nonzero first derivative of the variable gear ratio in the driveline dynamics. The default is unselected.

See Also   Driveline Environment, Simple Gear

A

Technical Conventions

Driveline Abbreviations and Conventions

An important abbreviation is DoF, which means *degree of freedom* and refers to one coordinate of angular motion.

Standard symbols for angular motion analysis include the following:

Symbol	Meaning (Units)
r	Gear radius (meters)
N	Number of gear teeth
g	Gear ratio
θ	Angle (radians)
ω	Angular velocity (radians/second)
τ	Torque (newton-meters)

Gear Ratios

For a pair of coupled, coplanar gear wheels, the gear ratio g_{21} of gear 2 to gear 1 is defined as the ratio of the second gear wheel radius to the first. This definition is equivalent to the ratio of the number of teeth on the second gear wheel to the number of teeth on the first.

$$g_{21} \equiv r_2/r_1 = N_2/N_1$$

The gear ratio is the ratio of torques and the reciprocal of the angular velocity ratio.

$$g_{21} = \tau_2/\tau_1 = \omega_1/\omega_2$$

For gear boxes made of more than two gear wheels, the gear ratio is defined to be the ratio of torques or the reciprocal of the ratio of angular velocities, between the output and input shafts.

If the gear is reversing, the output ω and τ have opposite signs from the input ω and τ.

Driveline Units

SimDriveline accepts meters-kilograms-seconds or MKS (SI) units only.

Quantity	Unit
Length	meter (m)
Angle	radian (rad)
Time	second (s)
Angular Velocity	radians/second (rad/s)
Angular Acceleration	radians/second^2 (rad/s^2)
Mass	kilogram (kg)
Force	newton (N)
Inertia	kilogram-meter^2 (kg-m^2)
Torque	newton-meter (N-m)

B

Bibliography

Centa, G., *Motor Vehicle Dynamics: Modeling and Simulation,* Singapore, World Scientific, 1997.

Goodman, L. E., and W. H. Warner, *Statics* (1964) and *Dynamics* (1965), New York, Dover Publications, 2001.

Jurgen, R. K., *Electronic Transmission Controls,* Troy, Michigan, Society of Automotive Engineers, 2000.

Juvinall, R. C., *Fundamentals of Machine Component Design,* New York, John Wiley & Sons, 1983.

The MathWorks, Inc., `www.mathworks.com/products/industry/auto/`, Products: Industry: Automotive.

Meriam, J. L., and L. G. Kraige, *Dynamics*, Volume 2 of *Engineering Mechanics,* New York, John Wiley & Sons, 1987.

Nwagboso, C. O., *Automotive Sensory Systems,* London, Chapman & Hall, 1993.

Pacejka, H. B. *Tire and Vehicle Dynamics,* Society of Automotive Engineers and Butterworth-Heinemann, Oxford, 2002.

Society of Automotive Engineers, `www.sae.org,` see "SAE Store".

Wong, J. Y., *Theory of Ground Vehicles,* 3rd Ed., New York, Interscience, 2001.

Index