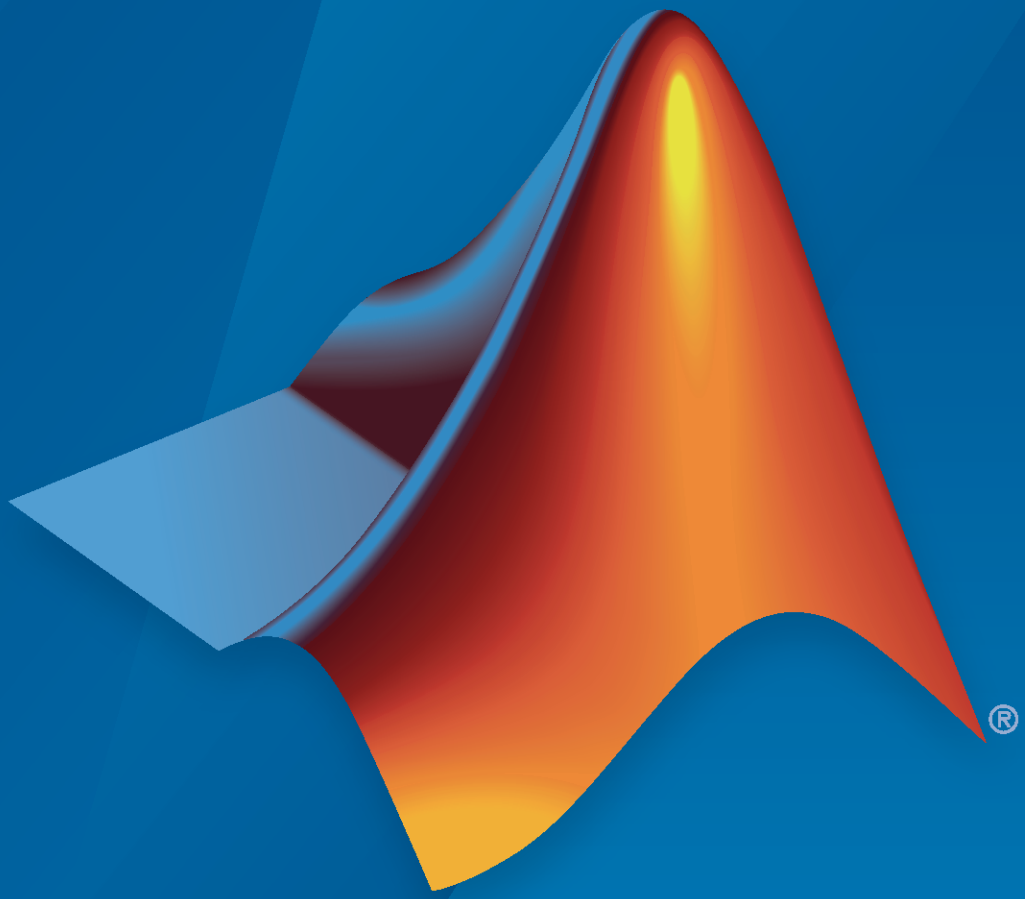
# Powertrain Blockset™

Reference



MATLAB® & SIMULINK®

R2020b

MathWorks®

# How to Contact MathWorks

Latest news: `www.mathworks.com`

Sales and services: `www.mathworks.com/sales_and_services`

User community: `www.mathworks.com/matlabcentral`

Technical support: `www.mathworks.com/support/contact_us`

Phone: 508-647-7000

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

*Powertrain Blockset™ Reference*

© COPYRIGHT 2016–2020 by The MathWorks, Inc.

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

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

**Trademarks**

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

**Patents**

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

**Revision History**

| | | |
|---|---|---|
| October 2016 | Online only | New for Version 1.0 (Release 2016b+) |
| March 2017 | Online only | Revised for Version 1.1 (Release 2017a) |
| September 2017 | Online only | Revised for Version 1.2 (Release 2017b) |
| March 2018 | Online only | Revised for Version 1.3 (Release 2018a) |
| September 2018 | Online only | Revised for Version 1.4 (Release 2018b) |
| March 2019 | Online only | Revised for Version 1.5 (Release 2019a) |
| September 2019 | Online only | Revised for Version 1.6 (Release 2019b) |
| March 2020 | Online only | Revised for Version 1.7 (Release 2020a) |
| September 2020 | Online only | Revised for Version 1.8 (Release 2020b) |

# Contents

1 Drivetrain Blocks

2 Vehicle Dynamics Blocks

3 Energy Storage Blocks

4 Propulsion Blocks

5 Electric Motor, Converters, Inverter Blocks

6 Scenario Creation Blocks

7 Transmission Blocks

8 Functions

# Drivetrain Blocks

# Rotational Inertia

Ideal mechanical rotational inertia

**Library:** Powertrain Blockset / Drivetrain / Couplings
Vehicle Dynamics Blockset / Powertrain / Drivetrain / Couplings

## Description

The Rotational Inertia block implements an ideal mechanical rotational inertia.

### Power Accounting

For the power accounting, the block implements these equations.

| Bus Signal | | | Description | Variable | Equations |
|---|---|---|---|---|---|
| `PwrInfo` | `PwrTrnsfrd` — Power transferred between blocks<br>• Positive signals indicate flow into block<br>• Negative signals indicate flow out of block | `PwrR` | Mechanical power from base shaft | $P_{TR}$ | $P_{TR} = \quad T_R\omega$ |
| | | `PwrC` | Mechanical power from follower shaft | $P_{TC}$ | $P_{TC} = \quad T_C\omega$ |
| | `PwrNotTrnsfrd` — Power crossing the block boundary, but not transferred<br>• Positive signals indicate an input<br>• Negative signals indicate a loss | `PwrDampLoss` | Power loss due to damping | $P_d$ | $P_d = \ -b\lvert\omega\rvert^2$ |
| | `PwrStored` — Stored energy rate of change<br>• Positive signals indicate an increase<br>• Negative signals indicate a decrease | `PwrStoredShft` | Rate change of stored internal torsional energy | $P_s$ | $P_s = \omega\dot{\omega}J$ |

The equations use these variables.

| | |
|---|---|
| $T_R$ | Input torque |
| $T_C$ | Output torque |
| $\omega$ | Driveshaft angular velocity |
| $J$ | Rotational inertia |
| $b$ | Rotational viscous damping |
| $P_d$ | Power loss due to damping |
| $P_s$ | Rate change of stored internal torsional energy |

## Ports

### Input

#### `RTrq` — Input torque
`scalar`

Applied input driveshaft torque, $T_R$, in N·m.

##### Dependencies

To enable this port, for **Port Configuration**, select `Simulink`.

#### `CTrq` — Output torque
`scalar`

Load driveshaft torque, $T_C$, in N·m.

##### Dependencies

To enable this port, for **Port Configuration**, select `Simulink`.

#### `R` — Angular velocity and torque
two-way connector port

Angular velocity in rad/s. Torque is in N·m.

##### Dependencies

To enable this port, for **Port Configuration**, select `Two-way connection`.

#### `Inertia` — Input
`scalar`

Rotational inertia, in kg·m^2.

##### Dependencies

To create the `Inertia` port, select **External inertia input**.

### Output

#### `Info` — Bus signal
bus

Bus signal containing these block calculations.

| Signal | | | Description | Variable | Units |
|---|---|---|---|---|---|
| Trq | R | | Applied input driveshaft torque | $T_R$ | N·m |
| | C | | Output driveshaft torque | $T_C$ | N·m |
| | Damp | | Damping torque | $T_d=b\omega$ | N·m |
| Spd | | | Angular driveshaft speed | $\omega$ | rad/s |
| PwrInfo | PwrTrnsfrd | PwrR | Mechanical power from base shaft | $P_{TR}$ | W |
| | | PwrC | Mechanical power from follower shaft | $P_{TC}$ | W |
| | PwrNotTrnsfrd | PwrDampLoss | Power loss due to damping | $P_d$ | W |
| | PwrStored | PwrStoredShft | Rate change of stored internal torsional energy | $P_s$ | W |

**Dependencies**

To enable this port, select **Output Info bus**.

### Spd — Driveshaft speed

`scalar`

Angular driveshaft speed, $\omega$, in rad/s.

**Dependencies**

To enable this port, for **Port Configuration**, select `Simulink`.

### C — Angular velocity and torque

two-way connector port

Angular velocity in rad/s. Torque is in N·m.

**Dependencies**

To enable this port, for **Port Configuration**, select `Two-way connection`.

## Parameters

**Block Options**

### Port Configuration — Specify configuration

`Simulink` (default) | `Two-way connection`

Specify the port configuration.

**Dependencies**

Specifying `Simulink` creates these ports:

- `RTrq`
- `CTrq`
- `Spd`

Specifying `Two-way connection` creates these ports:

- `R`
- `C`

**`Output Info bus` — Selection**
`off` (default) | `on`

Select to create the `Info` output port.

**`External inertia input` — Input rotational inertia**
`off` (default) | `on`

#### Dependencies

To create the `Inertia` port, select **External inertia input**.

#### Parameters

**`Rotational inertia, J` — Inertia**
`.01` (default) | `scalar`

Rotational inertia, in kg·m^2.

#### Dependencies

To enable this parameter, clear **Input rotational inertia**.

**`Torsional damping, b` — Damping**
`.001` (default) | `scalar`

Torsional damping, in N·m· s/rad.

**`Initial velocity, omega_o` — Angular**
`0` (default) | `scalar`

Initial angular velocity, in rad/s.

## Extended Capabilities

#### C/C++ Code Generation
Generate C and C++ code using Simulink® Coder™.

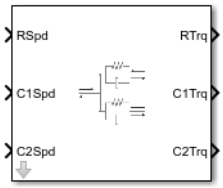
## See Also
Split Torsional Compliance | Torsional Compliance

#### Introduced in R2017a

# Split Torsional Compliance

Split torsional coupler

**Library:** Powertrain Blockset / Drivetrain / Couplings
Vehicle Dynamics Blockset / Powertrain / Drivetrain / Couplings




## Description

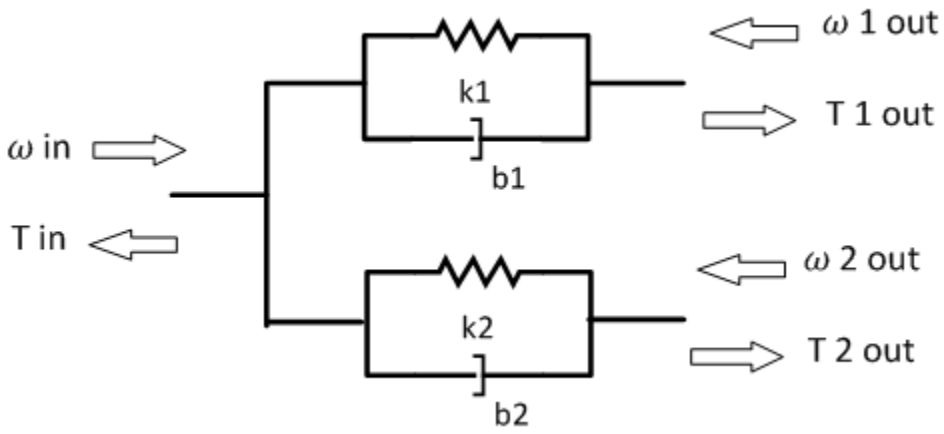
The Split Torsional Compliance block implements parallel spring-damper coupling between shafts. You can specify the type of coupling by selecting one of the **Coupling Configuration** parameters:

- `Shaft split` — Single input shaft coupled to two output shafts
- `Shaft merge` — Two input shafts coupled to a single output shaft

In fuel economy and emissions studies, you can use the Split Torsional Compliance block to model mechanical rotational compliance between common driveline elements such as motors, planetary gears, and clutches. For example, use the `Shaft split` configuration to couple a motor and two planetary gear sets. Use the `Shaft merge` configuration to couple a dual clutch transmission to an output shaft.

### Shaft Split

For the `Shaft split` configuration, the block implements this schematic and equations.



$$T_{in} = -(\omega_{in} - \omega_{1out})b_1 - (\omega_{in} - \omega_{2out})b_2 - \theta_1 k_1 - \theta_2 k_2$$
$$T_{1out} = (\omega_{in} - \omega_{1out})b_1 + \theta_1 k_1$$
$$T_{2out} = (\omega_{in} - \omega_{2out})b_2 + \theta_2 k_2$$

$$\dot{\theta}_1 = (\omega_{in} - \omega_{1out})$$
$$\dot{\theta}_2 = (\omega_{in} - \omega_{2out})$$

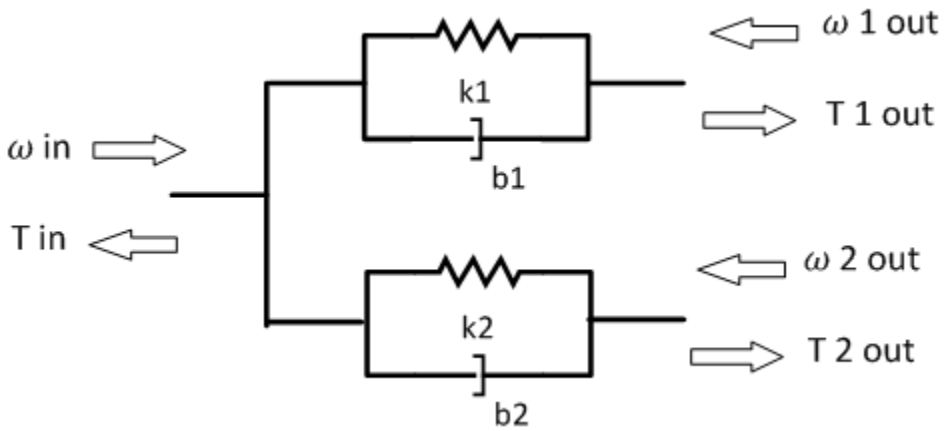
To account for frequency-dependent damping, both damping terms incorporate a low-pass filter.

The equations use these variables.

| | |
|---|---|
| $T_{in}$ | Resulting applied input reaction torque |
| $\omega_{in}$ | Input shaft rotational velocity |
| $T_{1out}$ | Resulting applied torque to first output shaft |
| $\omega_{1out}$ | First output shaft rotational velocity |
| $T_{2out}$ | Resulting applied torque to second output shaft |
| $\omega_{2out}$ | Second output shaft rotational velocity |
| $\theta_1$, $\theta_2$ | First, second shaft rotation, respectively |
| $b_1$, $b_2$ | First, second shaft viscous damping, respectively |
| $k_1$, $k_2$ | First, second shaft torsional stiffness, respectively |

**Shaft Merge**

For the `Shaft merge` configuration, the block implements this schematic and equations.




$$T_{out} = (-\omega_{out} + \omega_{1in})b_1 + (-\omega_{out} + \omega_{2in})b_2 + \theta_1 k_1 + \theta_2 k_2$$
$$T_{1out} = (\omega_{out} - \omega_{1in})b_1 - \theta_1 k_1$$
$$T_{2out} = (\omega_{out} - \omega_{2in})b_2 - \theta_2 k_2$$

$$\dot{\theta}_1 = (\omega_{1in} - \omega_{out})$$
$$\dot{\theta}_2 = (\omega_{2in} - \omega_{out})$$

To account for frequency-dependent damping, both damping terms incorporate a low-pass filter.

The equations use these variables.

| | |
|---|---|
| $T_{out}$ | Resulting applied output torque |
| $\omega_{out}$ | Output shaft rotational velocity |
| $T_{1in}$ | Resulting reaction torque to first input shaft |
| $\omega_{1in}$ | First input shaft rotational velocity |
| $T_{2in}$ | Resulting reaction torque to second input shaft |
| $\omega_{2in}$ | Second input shaft rotational velocity |
| $\theta_1$, $\theta_2$ | First, second shaft rotation, respectively |
| $b_1$, $b_2$ | First, second shaft viscous damping, respectively |
| $k_1$, $k_2$ | First, second shaft torsional stiffness, respectively |

### Power Accounting

For the power accounting, the block implements these equations.

| Bus Signal | | | Description | Variable | Equations |
|---|---|---|---|---|---|
| `PwrInfo` | `PwrTrnsfrd` — Power transferred between blocks<br>• Positive signals indicate flow into block<br>• Negative signals indicate flow out of block | `PwrR` | For the `Shaft split` configuration, mechanical power from input shaft | $P_{TR}$ | $P_{TR} = -T_R\omega_R$ |
| | | `PwrC1` | For the `Shaft split` configuration, mechanical power from first output shaft | $P_{TC1}$ | $P_{TC1} = -T_{C1}\omega_{C1}$ |
| | | `PwrC2` | For the `Shaft split` configuration, mechanical power from second output shaft | $P_{TC2}$ | $P_{TC2} = -T_{C2}\omega_{C2}$ |
| | | `PwrC` | For the `Shaft merge` configuration, mechanical power from output shaft | $P_{TC}$ | $P_{TC} = T_C\omega_C$ |

| Bus Signal | | | Description | Variable | Equations |
|---|---|---|---|---|---|
| | | `PwrR1` | For the `Shaft merge` configuration, mechanical power from first input shaft | $P_{TR1}$ | $P_{TR1} = T_{R1}\omega_{R1}$ |
| | | `PwrR2` | For the `Shaft merge` configuration, mechanical power from second input shaft | $P_{TR2}$ | $P_{TR2} = T_{R2}\omega_{R2}$ |
| | `PwrNotTrnsfrd` — Power crossing the block boundary, but not transferred<br>• Positive signals indicate an input<br>• Negative signals indicate a loss | `PwrDampLoss` | Mechanical damping loss | $P_d$ | $P_d = -\left(b_1\left\|\dot{\theta}_1\right\|^2 + b_2\left\|\dot{\theta}_2\right\|^2\right)$ |
| | `PwrStored` — Stored energy rate of change<br>• Positive signals indicate an increase<br>• Negative signals indicate a decrease | `PwrStoredShft` | Rate change in spring energy | $P_s$ | $P_s = \left(k_1\theta_1\dot{\theta}_1 + k_2\theta_2\dot{\theta}_2\right)$ |

The equations use these variables.

| | |
|---|---|
| $T_R$ | Shaft R torque |
| $T_C$ | Shaft C torque |
| $\omega_R$ | Shaft R angular velocity |
| $\omega_C$ | Shaft C angular velocity |
| $\theta$ | Coupled shaft rotation |
| $k$ | Shaft torsional stiffness |
| $b$ | Rotational viscous damping |
| $P_t$ | Total mechanical power |
| $P_d$ | Power loss due to damping |
| $P_s$ | Rate change of stored spring energy |

## Ports

### Input

#### RSpd — Input shaft speed
scalar

Input shaft rotational velocity, $\omega_{in}$, in rad/s.

**Dependencies**

To enable this port, set both of these parameters:

- **Port Configuration** to `Simulink`
- **Coupling Configuration** to `Shaft split`

#### C1Spd — First output shaft speed
scalar

First output shaft rotational velocity, $\omega_{1out}$, in rad/s.

**Dependencies**

To enable this port, set both of these parameters:

- **Port Configuration** to `Simulink`
- **Coupling Configuration** to `Shaft split`

#### C2Spd — Second output shaft speed
scalar

Second output shaft rotational velocity, $\omega_{2out}$, in rad/s.

**Dependencies**

To enable this port, set both of these parameters:

- **Port Configuration** to `Simulink`
- **Coupling Configuration** to `Shaft split`

#### CSpd — Input speed
scalar

Output shaft rotational velocity, $\omega_{out}$, in rad/s.

**Dependencies**

To enable this port, set both of these parameters:

- **Port Configuration** to `Simulink`
- **Coupling Configuration** to `Shaft merge`

#### R1Spd — First input shaft speed
scalar

First input shaft rotational velocity, $\omega_{1in}$, in rad/s.

#### Dependencies

To enable this port, set both of these parameters:

- **Port Configuration** to `Simulink`
- **Coupling Configuration** to `Shaft merge`

### R2Spd — Second input shaft speed

`scalar`

Second input shaft rotational velocity, $\omega_{2in}$, in rad/s.

#### Dependencies

To enable this port, set both of these parameters:

- **Port Configuration** to `Simulink`
- **Coupling Configuration** to `Shaft merge`

### R — Input shaft angular velocity and torque

two-way connector port

Input shaft angular velocity, $\omega_{in}$, in rad/s and torque, $T_{in}$, in N·m.

#### Dependencies

To enable this port, select:

- **Port Configuration**>`Two-way connection`
- **Coupling Configuration**>`Shaft split`

### R1 — First input shaft angular velocity and torque

two-way connector port

First input shaft angular velocity, $\omega_{1in}$, in rad/s and torque, $T_{1in}$, in N·m.

#### Dependencies

To enable this port, select:

- **Port Configuration**>`Two-way connection`
- **Coupling Configuration**>`Shaft merge`

### R2 — Second input shaft angular velocity and torque

two-way connector port

Second input shaft angular velocity, $\omega_{2in}$, in rad/s and torque, $T_{2in}$, in N·m.

#### Dependencies

To enable this port, select:

- **Port Configuration**>`Two-way connection`
- **Coupling Configuration**>`Shaft merge`

### Output

#### `Info` — Bus signal

bus

If you set **Coupling Configuration** to `Shaft split`, the Info bus contains these signals.

| Signal | | | Description | Variable | Units |
|---|---|---|---|---|---|
| Trq | `R` | | Input shaft torque | $T_{in}$ | N·m |
| | `C1` | | First output shaft torque | $T_{1out}$ | N·m |
| | `C2` | | Second output shaft torque | $T_{2out}$ | N·m |
| | `Damp` | `C1` | First output shaft damping torque | $b_1\omega_{1out}$ | N·m |
| | | `C2` | Second output shaft damping torque | $b_2\omega_{2out}$ | N·m |
| | `Spring` | `C1` | First output shaft spring torque | $k_1\theta_1$ | N·m |
| | | `C2` | Second output shaft spring torque | $k_2\theta_2$ | N·m |
| Spd | `R` | | Input shaft angular velocity | $\omega_{in}$ | rad/s |
| | `C1` | | First output shaft angular velocity | $\omega_{1out}$ | rad/s |
| | `C2` | | Second output shaft angular velocity | $\omega_{2out}$ | rad/s |
| | `deltadot1` | | Difference in input and first output shaft angular velocity | $\dot{\theta}_1$ | rad/s |
| | `deltadot2` | | Difference in input and second output shaft angular velocity | $\dot{\theta}_2$ | rad/s |
| PwrInfo | `PwrTrnsfrd` | `PwrR` | Mechanical power from input shaft | $P_{TR}$ | W |
| | | `PwrC1` | Mechanical power from first output shaft | $P_{TC1}$ | W |
| | | `PwrC2` | Mechanical power from second output shaft | $P_{TC2}$ | W |
| | `PwrNotTrnsfrd` | `PwrDampLoss` | Mechanical damping loss | $P_d$ | W |
| | `PwrStored` | `PwrStoredShft` | Rate change of stored internal torsional energy | $P_s$ | W |

If you set **Coupling Configuration** to `Shaft merge`, the Info bus contains these signals.

| Signal | | Description | Variable | Units |
|---|---|---|---|---|
| Trq | `C` | Output shaft torque | $T_{out}$ | N·m |
| | `R1` | First input shaft torque | $T_{1in}$ | N·m |
| | `R2` | Second input shaft torque | $T_{2in}$ | N·m |

| Signal | | | Description | Variable | Units |
|---|---|---|---|---|---|
| | Damp | R1 | First input shaft damping torque | $b_1\omega_{1in}$ | N·m |
| | | R2 | Second in shaft damping torque | $b_2\omega_{2in}$ | N·m |
| | Spring | R1 | First input shaft spring torque | $k_1\theta_1$ | N·m |
| | | R2 | Second in shaft spring torque | $k_2\theta_2$ | N·m |
| Spd | C | | Output shaft angular velocity | $\omega_{out}$ | rad/s |
| | R1 | | First input shaft angular velocity | $\omega_{1in}$ | rad/s |
| | R2 | | Second input shaft angular velocity | $\omega_{2in}$ | rad/s |
| | deltadot1 | | Difference in first input and output shaft angular velocity | $\dot{\theta}_1$ | rad/s |
| | deltadot2 | | Difference in second input and output shaft angular velocity | $\dot{\theta}_2$ | rad/s |
| PwrInfo | PwrTrnsfrd | PwrC | Mechanical power from output shaft | $P_{TC}$ | W |
| | | PwrR1 | Mechanical power from first input shaft | $P_{TR1}$ | W |
| | | PwrR2 | Mechanical power from second input shaft | $P_{TR2}$ | W |
| | PwrNotTrnsfrd | PwrDampLoss | Mechanical damping loss | $P_d$ | W |
| | PwrStored | PwrStoredShft | Rate change of stored internal torsional energy | $P_s$ | W |

#### Dependencies

To enable this port, select **Output Info bus**.

### RTrq — Input shaft torque

scalar

Input shaft torque, $T_{in}$, in N·m.

#### Dependencies

To enable this port, set both of these parameters:

- **Port Configuration** to `Simulink`
- **Coupling Configuration** to `Shaft split`

### C1Trq — First output shaft torque

scalar

First output shaft torque, $T_{1out}$, in N·m.

#### Dependencies

To enable this port, set both of these parameters:

- **Port Configuration** to `Simulink`
- **Coupling Configuration** to `Shaft split`

### `C2Trq` — Second output shaft torque

`scalar`

Second output shaft torque, $T_{2out}$, in N·m.

#### Dependencies

To enable this port, set both of these parameters:

- **Port Configuration** to `Simulink`
- **Coupling Configuration** to `Shaft split`

### `CTrq` — Output shaft torque

`scalar`

Output shaft torque, $T_{out}$, in N·m.

#### Dependencies

To enable this port, set both of these parameters:

- **Port Configuration** to `Simulink`
- **Coupling Configuration** to `Shaft merge`

### `R1Trq` — First input shaft torque

`scalar`

First input shaft torque, $T_{1in}$, in N·m.

#### Dependencies

To enable this port, set both of these parameters:

- **Port Configuration** to `Simulink`
- **Coupling Configuration** to `Shaft merge`

### `R2Trq` — Second input shaft torque

`scalar`

Second input shaft torque, $T_{2in}$, in N·m.

#### Dependencies

To enable this port, set both of these parameters:

- **Port Configuration** to `Simulink`
- **Coupling Configuration** to `Shaft merge`

### `C1` — First output shaft angular velocity and torque

two-way connector port

First output shaft angular velocity, $\omega_{1out}$, in rad/s and torque, $T_{1out}$, in N·m.

#### Dependencies

To enable this port, select:

- **Port Configuration**>`Two-way connection`
- **Coupling Configuration**>`Shaft split`

### `C2` — Second output shaft angular velocity and torque

two-way connector port

Second output shaft angular velocity, $\omega_{2out}$, in rad/s and torque, $T_{2out}$, in N·m.

#### Dependencies

To enable this port, select:

- **Port Configuration**>`Two-way connection`
- **Coupling Configuration**>`Shaft split`

### `C` — Output shaft angular velocity and torque

two-way connector port

Output shaft angular velocity, $\omega_{out}$, in rad/s and torque, $T_{out}$, in N·m.

#### Dependencies

To enable this port, select:

- **Port Configuration**>`Two-way connection`
- **Coupling Configuration**>`Shaft merge`

## Parameters

#### Block Options

### `Port Configuration` — Specify configuration

`Simulink` (default) | `Two-way connection`

Specify the port configuration.

### `Coupling Configuration` — Specify configuration

`Shaft split` (default) | `Shaft merge`

Specify the coupling type.

### `Output Info bus` — Selection

`off` (default) | `on`

Select to create the `Info` output port.

#### Coupling 1

### `Torsional stiffness, k1` — Stiffness

`5e4` (default) | `scalar`

Rotational inertia, $k_1$, in N·m/rad.

**`Torsional damping, b1` — Damping**
`1e2` (default) | `scalar`

Torsional damping, $b_1$, in N·m· s/rad.

**`Damping cutoff frequency, omega1_c` — Frequency**
`3000` (default) | `scalar`

Damping cutoff frequency, in rad/s.

#### Coupling 2

**`Torsional stiffness, k2` — Stiffness**
`5e4` (default) | `scalar`

Rotational inertia, $k_2$, in N·m/rad.

**`Torsional damping, b2` — Damping**
`1e2` (default) | `scalar`

Torsional damping, $b_2$, in N·m· s/rad.

**`Damping cutoff frequency, omega2_c` — Frequency**
`3000` (default) | `scalar`

Damping cutoff frequency, in rad/s.

## Extended Capabilities

#### C/C++ Code Generation
Generate C and C++ code using Simulink® Coder™.

## See Also
Rotational Inertia | Torsional Compliance

#### Introduced in R2017b

# Torsional Compliance

Parallel spring-damper

**Library:** Powertrain Blockset / Drivetrain / Couplings
Vehicle Dynamics Blockset / Powertrain / Drivetrain / Couplings

## Description

The Torsional Compliance block implements a parallel spring-damper to couple two rotating driveshafts. The block uses the driveshaft angular velocities, torsional stiffness, and torsional damping to determine the torques.

$T_R = -(\omega_R - \omega_C)b - \theta k$

$T_C = (\omega_R - \omega_C)b + \theta k$

$\dot{\theta} = (\omega_R - \omega_C)$

### Power Accounting

For the power accounting, the block implements these equations.

| Bus Signal | | | Description | Variable | Equations |
|---|---|---|---|---|---|
| `PwrInfo` | `PwrTrnsfrd` — Power transferred between blocks<br>• Positive signals indicate flow into block<br>• Negative signals indicate flow out of block | `PwrR` | Mechanical power from driveshaft R | $P_{TR}$ | $P_{TR} = T_R\omega_R$ |
| | | `PwrC` | Mechanical power from driveshaft C | $P_{TC}$ | $P_{TC} = T_c\omega_c$ |
| | `PwrNotTrnsfrd` — Power crossing the block boundary, but not transferred<br>• Positive signals indicate an input<br>• Negative signals indicate a loss | `PwrDampLoss` | Mechanical damping loss | $P_d$ | $P_d = -b\lvert\dot{\theta}\rvert^2$ |

| Bus Signal | | | Description | Variable | Equations |
|---|---|---|---|---|---|
| | `PwrStored` — Stored energy rate of change<br>• Positive signals indicate an increase<br>• Negative signals indicate a decrease | `PwrStoredShft` | Rate change in spring energy | $P_S$ | $P_s = -\theta k \dot{\theta}$ |

The equations use these variables.

| | |
|---|---|
| $T_R$ | Driveshaft R torque |
| $T_C$ | Driveshaft C torque |
| $\omega_R$ | Driveshaft R angular velocity |
| $\omega_C$ | Driveshaft C angular velocity |
| $\theta$ | Coupled driveshaft rotation |
| $k$ | Driveshaft torsional stiffness |
| $b$ | Rotational viscous damping |
| $P_d$ | Power loss due to damping |
| $P_s$ | Rate change of stored spring energy |

## Ports

### Input

#### RSpd — Driveshaft R angular velocity

`scalar`

Input driveshaft angular velocity, in rad/s.

##### Dependencies

To enable this port, for **Port Configuration**, select `Simulink`.

#### CSpd — Driveshaft C angular velocity

`scalar`

Output driveshaft angular velocity, in rad/s.

##### Dependencies

To enable this port, for **Port Configuration**, select `Simulink`.

#### R — Angular velocity and torque

two-way connector port

Angular velocity in rad/s. Torque is in N·m.

##### Dependencies

To enable this port, for **Port Configuration**, select `Two-way connection`.

### Output

#### `Info` — Bus signal
bus

Bus signal containing these block calculations.

| Signal | | | Description | Variable | Units |
|---|---|---|---|---|---|
| Trq | `R` | | Input driveshaft torque | $T_R$ | N·m |
| | `C` | | Output driveshaft torque | $T_C$ | N·m |
| | `Damp` | | Damping torque | $T_s = b\dot{\theta}$ | N·m |
| | `Spring` | | Spring torque | $T_d = k\theta$ | N·m |
| Spd | `R` | | Input driveshaft angular velocity | $\omega_R$ | rad/s |
| | `C` | | Output driveshaft angular velocity | $\omega_C$ | rad/s |
| | `deltadot` | | Difference in input and output driveshaft angular velocity | $\dot{\theta}$ | rad/s |
| PwrInfo | `PwrTrnsfrd` | `PwrR` | Mechanical power from driveshaft R | $P_{TR}$ | W |
| | | `PwrC` | Mechanical power from driveshaft C | $P_{TC}$ | W |
| | `PwrNotTrnsfrd` | `PwrDampLoss` | Power loss due to damping | $P_d$ | W |
| | `PwrStored` | `PwrStoredShft` | Rate change of stored internal kinetic energy | $P_s$ | W |

##### Dependencies

To enable this port, select **Output Info bus**.

#### `RTrq` — Driveshaft R torque
scalar

Input drive shaft torque, in N·m.

##### Dependencies

To enable this port, for **Port Configuration**, select `Simulink`.

#### `CTrq` — Driveshaft C torque
scalar

Applied output driveshaft torque, in N·m.

##### Dependencies

To enable this port, for **Port Configuration**, select `Simulink`.

**C — Angular velocity and torque**
two-way connector port

Angular velocity in rad/s. Torque is in N·m.

**Dependencies**

To enable this port, for **Port Configuration**, select `Two-way connection`.

## Parameters

**Block Options**

**Port Configuration — Specify configuration**
`Simulink` (default) | `Two-way connection`

Specify the port configuration.

**Dependencies**

Specifying `Simulink` creates these ports:

- `RSpd`
- `CSpd`
- `RTrq`
- `CTrq`

Specifying `Two-way connection` creates these ports:

- `R`
- `C`

**Output Info bus — Selection**
`off` (default) | `on`

Select to create the `Info` output port.

**Torsional stiffness, k — Inertia**
`1e4` (default) | `scalar`

Torsional stiffness, in N·m/rad.

**Torsional damping, b — Damping**
`1e2` (default) | `scalar`

Torsional damping, in N·m· s/rad.

**Initial deflection, theta_o — Angular**
`0` (default) | `scalar`

Initial deflection, in rad.

**Initial velocity difference, domega_o — Angular**
`0` (default) | `scalar`

Initial velocity difference, in rad/s.

**`Damping cut-off frequency, omega_c` — Frequency**
`3000` (default) | `scalar`

Damping cut-off frequency, in rad/s.

## Extended Capabilities

### C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

## See Also

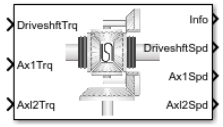
Rotational Inertia | Split Torsional Compliance

**Introduced in R2017a**

# Limited Slip Differential

Limited differential as a planetary bevel gear

**Library:** Powertrain Blockset / Drivetrain / Final Drive Unit
Vehicle Dynamics Blockset / Powertrain / Drivetrain / Final Drive Unit




## Description

The Limited Slip Differential block implements a differential as a planetary bevel gear train. The block matches the driveshaft bevel gear to the crown (ring) bevel gear. You can specify:

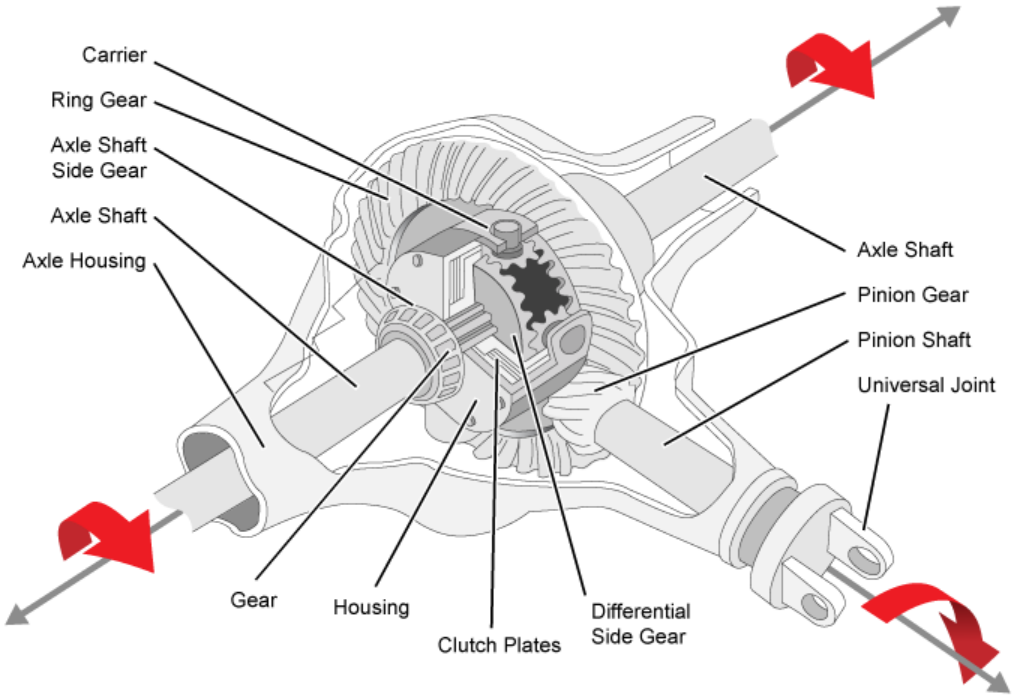
- Carrier-to-driveshaft ratio
- Crown wheel location
- Viscous and damping coefficients for the axles and carrier
- Type of slip coupling

Use the block in system-level driveline analysis to account for the power transfer from the transmission to the wheels. The block is suitable for use in hardware-in-the-loop (HIL) and optimization workflows. All the parameters are tunable.

In a limited slip differential, to prevent one of the wheels from slipping, the differential splits the torque applied to the left and right axles. With different torque applied to the axles, the wheels can move at different angular velocities, preventing slip. The block implements three methods for coupling the different torques applied to the axes:

- Pre-loaded ideal clutch
- Slip speed-dependent torque data
- Input torque dependent torque data

The block uses a coordinate system that produces positive tire and vehicle motion for standard engine, transmission, and differential configurations. The arrows indicate positive motion.




### Efficiency

To account for the block efficiency, use the **Efficiency factors** parameter. This table summarizes the block implementation for each setting.

| Setting | Implementation |
|---|---|
| `Constant` | Constant efficiency that you can set with the **Constant efficiency factor, eta** parameter. |
| `Driveshaft torque, temperature and speed` | Efficiency as a function of base gear input torque, air temperature, and driveshaft speed. Use these parameters to specify the lookup table and breakpoints:<br>• **Efficiency lookup table, eta_tbl**<br>• **Efficiency torque breakpoints, Trq_bpts**<br>• **Efficiency speed breakpoints, omega_bpts**<br>• **Efficiency temperature breakpoints, Temp_bpts**<br>For the air temperature, you can either:<br>• Select **Input temperature** to create an input port.<br>• Set a **Ambient temperature, Tamb** parameter value.<br>To select the interpolation method, use the **Interpolation method** parameter. For more information, see “Interpolation Methods”. |

### Power Accounting

For the power accounting, the block implements these equations.

| Bus Signal | | | Description | Equations |
|---|---|---|---|---|
| `PwrInfo` | `PwrTrnsfrd` — Power transferred between blocks<br>• Positive signals indicate flow into block<br>• Negative signals indicate flow out of block | `PwrDriveshft` | Mechanical power from driveshaft | $\eta T_d \omega_d$ |
| | | `PwrAxl1` | Mechanical power from axle 1 | $\eta T_1 \omega_1$ |
| | | `PwrAxl2` | Mechanical power from axle 2 | $\eta T_2 \omega_2$ |
| | `PwrNotTrnsfrd` — Power crossing the block boundary, but not transferred<br>• Positive signals indicate an input<br>• Negative signals indicate a loss | `PwrMechLoss` | Total power loss | $\dot{W}_{loss} = -(P_t + P_d + P_c) + P_s$<br>$P_t = \eta(T_d\omega_d + T_1\omega_1 + T_2\omega_2)$ |
| | | `PwrDampLoss` | Power loss due to damping | $P_d = -(b_1\lvert\omega_1\rvert + b_2\lvert\omega_2\rvert + b_d\lvert\omega_d\rvert)$ |
| | | `PwrCplngLoss` | Power loss due to clutch | $P_c = T_c\lvert\overline{\omega}\rvert$ |
| | `PwrStored` — Stored energy rate of change<br>• Positive signals indicate an increase<br>• Negative signals indicate a decrease | `PwrStoredShft` | Rate change of stored internal energy | $P_s = -(\omega_1\dot{\omega}_1 J_1 + \omega_2\dot{\omega}_2 J_2 + \omega_d\dot{\omega}_d J_d)$ |

### Dynamics

The Limited Slip Differential block implements these differential equations to represent the mechanical dynamic response for the crown gear, left axle, and right axle.

| Mechanical Dynamic Response | Differential Equation |
|---|---|
| Crown Gear | $\dot{\omega}_d J_d = \eta T_d - \omega_d b_d - T_i$ |
| Left Axle | $\dot{\omega}_1 J_1 = \eta T_1 - \omega_1 b_1 - T_{i1}$ |
| Right Axle | $\dot{\omega}_2 J_2 = \eta T_2 - \omega_2 b_2 - T_{i2}$ |

The block assumes rigid coupling between the crown gear and axles. These constraint equations apply.

$$\eta T_1 = \frac{N}{2} T_i - \frac{1}{2} T_c$$

$$\eta T_2 = \frac{N}{2} T_i + \frac{1}{2} T_c$$

$$\omega_d = \frac{N}{2}(\omega_1 + \omega_2)$$

The equations use these variables.

| | |
|---|---|
| $N$ | Carrier-to-driveshaft gear ratio |
| $J_d$ | Rotational inertia of the crown gear assembly |
| $b_d$ | Crown gear linear viscous damping |
| $\omega_d$ | Driveshaft angular speed |
| $\varpi$ | Slip speed |
| $J_1$ | Axle 1 rotational inertia |
| $b_1$ | Axle 1 linear viscous damping |
| $\omega_1$ | Axle 1 speed |
| $J_2$ | Axle 2 rotational inertia |
| $b_2$ | Axle 2 linear viscous damping |
| $\omega_2$ | Axle 2 angular speed |
| $\eta$ | Efficiency |
| $T_d$ | Driveshaft torque |
| $T_1$ | Axle 1 torque |
| $T_2$ | Axle 2 torque |
| $T_i$ | Axle internal resistance torque |
| $T_{i1}$ | Axle 1 internal resistance torque |
| $T_{i2}$ | Axle 2 internal resistance torque |
| $\mu$ | Coefficient of friction |
| $R_{eff}$ | Effective clutch radius |
| $R_o$ | Annular disk outer radius |
| $R_i$ | Annular disk inner radius |
| $F_c$ | Clutch force |
| $T_c$ | Clutch torque |
| $\mu$ | Coefficient of friction |

Table blocks in the Limited Slip Differential have these parameter settings:

- **Interpolation method** — `Linear`
- **Extrapolation method** — `Clip`

#### Ideal Clutch Coupling

The ideal clutch coupling model uses the axle slip speed and friction to calculate the clutch torque. The friction coefficient is a function of the slip speed.

$$T_c = F_c N \mu(|\varpi|) R_{eff} \tanh(4|\varpi|)$$

The disc radii determine the effective clutch radius over which the clutch force acts.

$$R_{eff} = \frac{2(R_o3 - R_i3)}{3(R_o2 - R_i2)}$$

The angular velocities of the axles determine the slip speed.

$$\varpi = \omega_1 - \omega_2$$

#### Slip Speed Coupling

To calculate the clutch torque, the slip speed coupling model uses torque data that is a function of slip speed. The angular velocities of the axles determine the slip speed.

$$\varpi = \omega_1 - \omega_2$$

#### Input Torque Coupling

To calculate the clutch torque, the input torque coupling model uses torque data that is a function of input torque.

The Open Differential block assumes rigid coupling between the crown gear and axles. These constraint equations apply.

$$\eta T_1 = \quad \eta T_2 = \frac{N}{2} T_i$$

$$\omega_{d\,=} \; = \frac{N}{2}(\omega_1 + \omega_2)$$

## Ports

### Inputs

#### `DriveshftTrq` — Torque

`scalar`

Applied input torque, typically from the engine crankshaft, in N·m.

#### `Axl1Trq` — Torque

`scalar`

Axle 1 torque, $T_1$, in N·m.

#### `Axl2Trq` — Torque

`scalar`

Axle 2 torque, $T_2$, in N·m.

#### `Temp` — Temperature

`scalar`

Temperature, in K.

#### Dependencies

To enable this port:

- Set **Efficiency factors** to `Driveshaft torque, speed and temperature`.
- Select **Input temperature**.

### Output

#### `Info` — Bus signal

bus

Bus signal containing these block calculations.

| Signal | | | Description | Units |
|---|---|---|---|---|
| `Driveshft` | `DriveshftTrq` | | Driveshaft torque | N·m |
| | `DriveshftSpd` | | Driveshaft speed | rad/s |
| `Axl1` | `Axl1Trq` | | Axle 1 torque | N·m |
| | `Axl1Spd` | | Axle 1 speed | rad/s |
| `Axl2` | `Axl2Trq` | | Axle 2 torque | N·m |
| | `Axl2Spd` | | Axle 2 speed | rad/s |
| `Cplng` | `CplngTrq` | | Torque coupling | N·m |
| | `CplngSlipSpd` | | Slip speed | rad/s |
| `PwrInfo` | `PwrTrnsfrd` | `PwrDriveshft` | Mechanical power from driveshaft | W |
| | | `PwrAxl1` | Mechanical power from axle 1 | W |
| | | `PwrAxl2` | Mechanical power from axle 2 | W |
| | `PwrNotTrnsfrd` | `PwrMechLoss` | Total power loss | W |
| | | `PwrDampLoss` | Power loss due to damping | W |
| | | `PwrCplngLoss` | Power loss due to clutch | W |
| | `PwrStoredShft` | `PwrStoredShft` | Rate change of stored internal energy | W |

#### `DriveshftSpd` — Angular speed

scalar

Driveshaft angular speed, $\omega_d$, in rad/s.

#### `Axl1Spd` — Angular speed

scalar

Axle 1 angular speed, $\omega_1$, in rad/s.

#### `Axl2Spd` — Angular speed

scalar

Axle 2 angular speed, $\omega_2$, in rad/s.

## Parameters

#### Block Options

**`Efficiency factors` — Specify configuration**
`Constant` (default) | `Driveshaft torque, speed and temperature`

To account for the block efficiency, use the **Efficiency factors** parameter. This table summarizes the block implementation for each setting.

| Setting | Implementation |
|---|---|
| `Constant` | Constant efficiency that you can set with the **Constant efficiency factor, eta** parameter. |
| `Driveshaft torque, temperature and speed` | Efficiency as a function of base gear input torque, air temperature, and driveshaft speed. Use these parameters to specify the lookup table and breakpoints:<br>• **Efficiency lookup table, eta_tbl**<br>• **Efficiency torque breakpoints, Trq_bpts**<br>• **Efficiency speed breakpoints, omega_bpts**<br>• **Efficiency temperature breakpoints, Temp_bpts**<br>For the air temperature, you can either:<br>• Select **Input temperature** to create an input port.<br>• Set a **Ambient temperature, Tamb** parameter value.<br>To select the interpolation method, use the **Interpolation method** parameter. For more information, see “Interpolation Methods”. |

**`Interpolation method` — Method**
`Flat` | `Nearest` | `Linear point-slope` | `Linear Lagrange` | `Cubic spline`

For more information, see “Interpolation Methods”.

#### Dependencies

To enable this parameter, set **Efficiency factors** to `Driveshaft torque, speed and temperature`.

**`Input temperature` — Create input port**
`off` (default) | `on`

Select to create input port `Temp` for the temperature.

#### Dependencies

To enable this parameter, set **Efficiency factors** to `Driveshaft torque, speed and temperature`.

#### Open Differential

**`Crown wheel (ring gear) located` — Specify crown wheel connection**
`To the left of center-line` (default) | `To the right of center-line`

Specify the crown wheel connection to the driveshaft.

**`Carrier to drive shaft ratio, NC/ND` — Ratio**
`4` (default) | `scalar`

Carrier-to-driveshaft gear ratio, $N$.

**`Carrier inertia, Jd` — Inertia**
`.1` (default) | `scalar`

Rotational inertia of the crown gear assembly, $J_d$, in kg·m^2. You can include the driveshaft inertia.

**`Carrier damping, bd` — Damping**
`1e-3` (default) | `scalar`

Crown gear linear viscous damping, $b_d$, in N·m·s/rad.

**`Axle 1 inertia, Jw1` — Inertia**
`.1` (default) | `scalar`

Axle 1 rotational inertia, $J_1$, in kg·m^2.

**`Axle 1 damping, bw1` — Damping**
`1e-3` (default) | `scalar`

Axle 1 linear viscous damping, $b_1$, in N·m·s/rad.

**`Axle 2 inertia, Jw2` — Inertia**
`.1` (default) | `scalar`

Axle 2 rotational inertia, $J_2$, in kg·m^2.

**`Axle 2 damping, bw2` — Damping**
`1e-3` (default) | `scalar`

Axle 2 linear viscous damping, $b_2$, in N·m·s/rad.

**`Axle 1 initial velocity, omegaw1o` — Angular velocity**
`0` (default) | `scalar`

Axle 1 initial velocity, $\omega_{o1}$, in rad/s.

**`Axle 2 initial velocity, omegaw2o` — Angular velocity**
`0` (default) | `scalar`

Axle 2 initial velocity, $\omega_{o2}$, in rad/s.

**`Constant efficiency factor, eta` — Efficiency**
`1` (default) | `scalar`

Constant efficiency, $\eta$.

#### Dependencies

To enable this parameter, set **Efficiency factors** to `Constant`.

**`Efficiency lookup table, eta_tbl` — Lookup table**
`M`-by-`N`-by-`L` array

Dimensionless array of values for efficiency as a function of:

- `M` input torques
- `N` input speed
- `L` air temperatures

Each value specifies the efficiency for a specific combination of torque, speed, and temperature. The array size must match the dimensions defined by the torque, speed, and temperature breakpoint vectors.

**Dependencies**

To enable this parameter, set **Efficiency factors** to `Driveshaft torque, speed and temperature`.

**`Efficiency torque breakpoints, Trq_bpts` — Torque breakpoints**
`[25, 50, 75, 100, 150, 200, 250]` (default) | 1-by-`M` vector

Vector of input torque, breakpoints for efficiency, in N·m.

**Dependencies**

To enable this parameter, set **Efficiency factors** to `Driveshaft torque, speed and temperature`.

**`Efficiency speed breakpoints, omega_bpts` — Speed breakpoints**
`[52.4 78.5 105 131 157 183 209 262 314 419 524]` (default) | 1-by-`N` vector

Vector of speed, breakpoints for efficiency, in rad/s.

**Dependencies**

To enable this parameter, set **Efficiency factors** to `Driveshaft torque, speed and temperature`.

**`Efficiency temperature breakpoints, Temp_bpts` — Temperature breakpoints**
`[290 358]` (default) | 1-by-`L` vector

Vector of ambient temperature breakpoints for efficiency, in K.

**Dependencies**

To enable this parameter, set **Efficiency factors** to `Driveshaft torque, speed and temperature`.

**`Ambient temperature, Tamb` — Ambient temperature**
`297.15` (default) | `scalar`

Ambient air temperature, $T_{air}$, in K.

**Dependencies**

To enable this parameter:

- Set **Efficiency factors** to `Driveshaft torque, speed and temperature`.
- Clear **Input temperature**.

#### Slip Coupling

**`Coupling type` — Torque coupling**
`Pre-loaded ideal clutch` (default) | `Slip speed dependent torque data` | `Input torque dependent torque data`

Specify the type of torque coupling.

**`Number of disks, Ndisks` — Torque coupling**
`4` (default) | `scalar`

Number of disks.

#### Dependencies

To enable the ideal clutch parameters, select `Pre-loaded ideal clutch` for the **Coupling type** parameter.

**`Effective radius, Reff` — Radius**
`.20` (default) | `scalar`

The effective radius, $R_{eff}$, used with the applied clutch friction force to determine the friction force. The effective radius is defined as:

$$R_{eff} = \frac{2(R_o{}^3 - R_i{}^3)}{3(R_o{}^2 - R_i{}^2)}$$

The equation uses these variables.

| | |
|---|---|
| $R_o$ | Annular disk outer radius |
| $R_i$ | Annular disk inner radius |

#### Dependencies

To enable the clutch parameters, select `Pre-loaded ideal clutch` for the **Coupling type** parameter.

**`Nominal preload force, Fc` — Force**
`500` (default) | `scalar`

Nominal preload force, in N.

#### Dependencies

To enable the clutch parameters, select `Pre-loaded ideal clutch` for the **Coupling type** parameter.

**`Friction coefficient vector, muc` — Friction**
`[.16 0.13 0.115 0.11 0.105 0.1025 0.10125]` (default) | `vector`

Friction coefficient vector.

**Dependencies**

To enable the clutch parameters, select `Pre-loaded ideal clutch` for the **Coupling type** parameter.

**`Slip speed vector, dw` — Angular velocity**
`[0 10 20 40 60 80 100]` (default) | `vector`

Slip speed vector, in rad/s.

**Dependencies**

To enable the clutch parameters, select `Pre-loaded ideal clutch` for the **Coupling type** parameter.

**`Torque - slip speed vector, Tdw` — Torque**
`[-100, -90, -50, -5, 0, 5, 50, 90, 100]` (default) | `vector`

Torque vector, in N·m.

**Dependencies**

To enable the slip speed parameters, select `Slip speed dependent torque data` for the **Coupling type** parameter.

**`Slip speed vector, dwT` — Angular velocity**
`[-200, -175, -100, - 50, 0, 50, 100, 175, 200]` (default) | `vector`

Slip speed vector, in rad/s.

**Dependencies**

To enable the slip speed parameters, select `Slip speed dependent torque data` for the **Coupling type** parameter.

**`Torque - input torque vector, TTin` — Torque**
`[-200 -175 -100 - 50 0 50 100 175 200]` (default) | `vector`

Torque vector, in N·m.

**Dependencies**

To enable the input torque parameters, select `Input torque dependent torque data` for the **Coupling type** parameter.

**`Input torque vector, Tin` — Torque**
`[-200 -175 -100 - 50 0 50 100 175 200]` (default) | `vector`

Torque vector, in N·m.

**Dependencies**

To enable the input torque parameters, select `Input torque dependent torque data` for the **Coupling type** parameter.

**`Coupling time constant, tauC` — Constant**
`.01` (default) | `scalar`

Coupling time constant, in s.

## References

[1] Deur, J., Ivanović, V., Hancock, M., and Assadian, F. "Modeling of Active Differential Dynamics." In ASME proceedings. *Transportation Systems*. Vol. 17, pp: 427-436.

## Extended Capabilities

### C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

## See Also

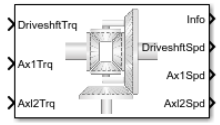
Open Differential

### Introduced in R2017a

# Open Differential

Differential as a planetary bevel gear

**Library:** Powertrain Blockset / Drivetrain / Final Drive Unit
Vehicle Dynamics Blockset / Powertrain / Drivetrain / Final Drive Unit




## Description

The Open Differential block implements a differential as a planetary bevel gear train. The block matches the driveshaft bevel gear to the crown (ring) bevel gear. You can specify:

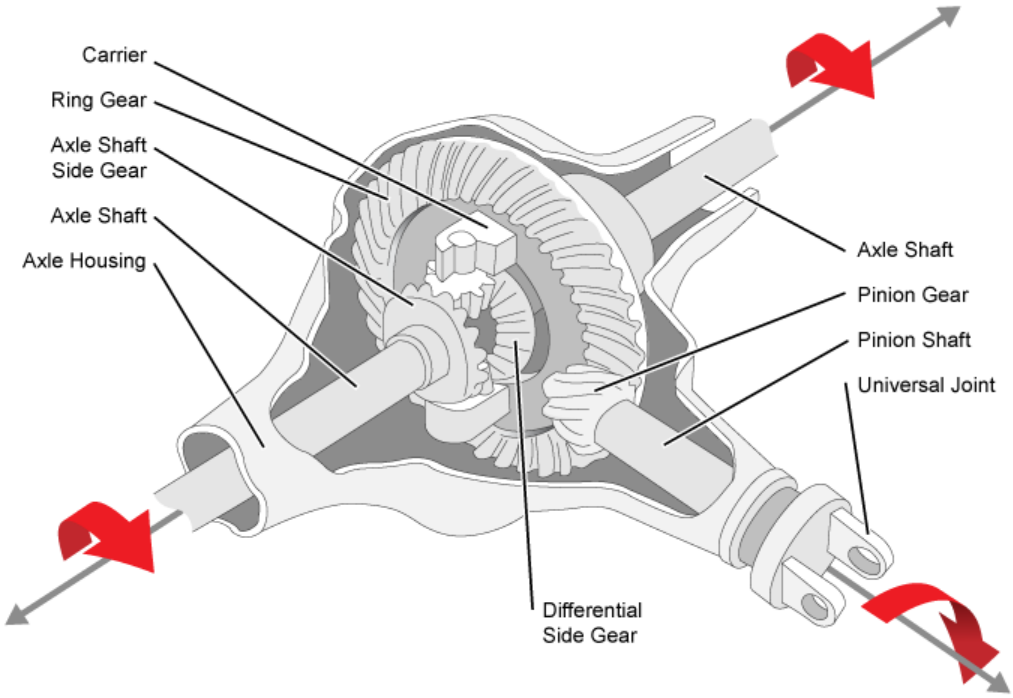
- Carrier-to-driveshaft ratio
- Crown wheel location
- Viscous and damping coefficients for the axles and carrier

Use the Open Differential block to:

- Dynamically couple the post-transmission driveshaft to the wheel axles or universal joints
- Model simplified or older drivetrains when optimal traction control does not require passive or active torque vectoring
- Model mechanical power splitting in generic gearbox and drive line scenarios

The block is suitable for use in hardware-in-the-loop (HIL) and optimization workflows. All the parameters are tunable.

The block uses a coordinate system that produces positive tire and vehicle motion for standard engine, transmission, and differential configurations. The arrows indicate positive motion.




### Efficiency

To account for the block efficiency, use the **Efficiency factors** parameter. This table summarizes the block implementation for each setting.

| Setting | Implementation |
|---|---|
| `Constant` | Constant efficiency that you can set with the **Constant efficiency factor, eta** parameter. |
| `Driveshaft torque, temperature and speed` | Efficiency as a function of base gear input torque, air temperature, and driveshaft speed. Use these parameters to specify the lookup table and breakpoints:<br>• **Efficiency lookup table, eta_tbl**<br>• **Efficiency torque breakpoints, Trq_bpts**<br>• **Efficiency speed breakpoints, omega_bpts**<br>• **Efficiency temperature breakpoints, Temp_bpts**<br>For the air temperature, you can either:<br>• Select **Input temperature** to create an input port.<br>• Set a **Ambient temperature, Tamb** parameter value.<br>To select the interpolation method, use the **Interpolation method** parameter. For more information, see “Interpolation Methods”. |

### Power Accounting

For the power accounting, the block implements these equations.

| Bus Signal | | | Description | Equations |
|---|---|---|---|---|
| `PwrInfo` | `PwrTrnsfrd` — Power transferred between blocks<br>• Positive signals indicate flow into block<br>• Negative signals indicate flow out of block | `PwrDriveshft` | Mechanical power from driveshaft | $\eta T_d \omega_d$ |
| | | `PwrAxl1` | Mechanical power from axle 1 | $\eta T_1 \omega_1$ |
| | | `PwrAxl2` | Mechanical power from axle 2 | $\eta T_2 \omega_2$ |
| | `PwrNotTrnsfrd` — Power crossing the block boundary, but not transferred<br>• Positive signals indicate an input<br>• Negative signals indicate a loss | `PwrMechLoss` | Total power loss | $\dot{W}_{loss} = -(P_t + P_d) + P_s$<br>$P_t = \eta T_d \omega_d + \eta T_1 \omega_1 + \eta T_2 \omega_2$ |
| | | `PwrDampLoss` | Power loss due to damping | $P_d = -(b_1\lvert\omega_1\rvert + b_2\lvert\omega_2\rvert + b_d\lvert\omega_d\rvert)$ |
| | `PwrStored` — Stored energy rate of change<br>• Positive signals indicate an increase<br>• Negative signals indicate a decrease | `PwrStoredShft` | Rate change of stored internal energy | $P_s = -(\omega_1\dot{\omega}_1 J_1 + \omega_2\dot{\omega}_2 J_2 + \omega_d\dot{\omega}_d J_d)$ |

### Dynamics

The Open Differential block implements these differential equations to represent the mechanical dynamic response for the crown gear, left axle, and right axle.

| Mechanical Dynamic Response | Differential Equation |
|---|---|
| Crown Gear | $\dot{\omega}_d J_d = \eta T_d - \omega_d b_d - T_i$ |
| Left Axle | $\dot{\omega}_1 J_1 = \eta T_1 - \omega_1 b_1 - T_{i1}$ |
| Right Axle | $\dot{\omega}_2 J_2 = \eta T_2 - \omega_2 b_2 - T_{i2}$ |

The Open Differential block assumes rigid coupling between the crown gear and axles. These constraint equations apply.

$$\eta T_1 = \eta T_2 = \frac{N}{2} T_i$$

$$\omega_d = \frac{N}{2}(\omega_1 + \omega_2)$$

The equations use these variables.

| | |
|---|---|
| $N$ | Carrier-to-driveshaft gear ratio |
| $J_d$ | Rotational inertia of the crown gear assembly |
| $b_d$ | Crown gear linear viscous damping |
| $\omega_d$ | Driveshaft angular speed |
| $\eta$ | Differential efficiency |
| $J_1$ | Axle 1 rotational inertia |
| $b_1$ | Axle 1 linear viscous damping |
| $\omega_1$ | Axle 1 speed |
| $J_2$ | Axle 2 rotational inertia |
| $b_2$ | Axle 2 linear viscous damping |
| $\omega_2$ | Axle 2 angular speed |
| $T_d$ | Driveshaft torque |
| $T_1$ | Axle 1 torque |
| $T_2$ | Axle 2 torque |
| $T_i$ | Driveshaft internal resistance torque |
| $T_{i1}$ | Axle 1 internal resistance torque |
| $T_{i2}$ | Axle 2 internal resistance torque |

## Ports

### Inputs

#### `DriveshftTrq` — Torque

scalar

Applied input torque, typically from the engine crankshaft, in N·m.

#### `Axl1Trq` — Torque

scalar

Axle 1 torque, $T_1$, in N·m.

#### `Axl2Trq` — Torque

scalar

Axle 2 torque, $T_2$, in N·m.

#### `Temp` — Temperature

scalar

Temperature, in K.

##### Dependencies

To enable this port:

- Set **Efficiency factors** to `Driveshaft torque, speed and temperature`.
- Select **Input temperature**.

### Output

#### `Info` — Bus signal

bus

Bus signal containing these block calculations.

| Signal | | | Description | Units |
|---|---|---|---|---|
| `Driveshft` | `DriveshftTrq` | | Driveshaft torque | N·m |
| | `DriveshftSpd` | | Driveshaft speed | rad/s |
| `Axl1` | `Axl1Trq` | | Axle 1 torque | N·m |
| | `Axl1Spd` | | Axle 1 speed | rad/s |
| `Axl2` | `Axl2Trq` | | Axle 2 torque | N·m |
| | `Axl2Spd` | | Axle 2 speed | rad/s |
| `PwrInfo` | `PwrTrnsfrd` | `PwrDriveshft` | Mechanical power from driveshaft | W |
| | | `PwrAxl1` | Mechanical power from axle 1 | W |
| | | `PwrAxl2` | Mechanical power from axle 2 | W |
| | `PwrTrnsfrd` | `PwrMechLoss` | Total power loss | W |
| | | `PwrDampLoss` | Power loss due to damping | W |
| | `PwrStored` | `PwrStoredShft` | Rate change of stored internal energy | W |

#### `DriveshftSpd` — Angular speed

scalar

Driveshaft angular speed, $\omega_d$, in rad/s.

#### `Axl1Spd` — Angular speed

scalar

Axle 1 angular speed, $\omega_1$, in rad/s.

#### `Axl2Spd` — Angular speed

scalar

Axle 2 angular speed, $\omega_2$, in rad/s.

## Parameters

#### Block Options

**`Efficiency factors` — Specify configuration**
`Constant` (default) | `Driveshaft torque, speed and temperature`

To account for the block efficiency, use the **Efficiency factors** parameter. This table summarizes the block implementation for each setting.

| Setting | Implementation |
|---|---|
| `Constant` | Constant efficiency that you can set with the **Constant efficiency factor, eta** parameter. |
| `Driveshaft torque, temperature and speed` | Efficiency as a function of base gear input torque, air temperature, and driveshaft speed. Use these parameters to specify the lookup table and breakpoints:<br>• **Efficiency lookup table, eta_tbl**<br>• **Efficiency torque breakpoints, Trq_bpts**<br>• **Efficiency speed breakpoints, omega_bpts**<br>• **Efficiency temperature breakpoints, Temp_bpts**<br>For the air temperature, you can either:<br>• Select **Input temperature** to create an input port.<br>• Set a **Ambient temperature, Tamb** parameter value.<br>To select the interpolation method, use the **Interpolation method** parameter. For more information, see “Interpolation Methods”. |

**`Interpolation method` — Method**
`Flat` | `Nearest` | `Linear point-slope` | `Linear Lagrange` | `Cubic spline`

For more information, see “Interpolation Methods”.

#### Dependencies

To enable this parameter, set **Efficiency factors** to `Driveshaft torque, speed and temperature`.

**`Input temperature` — Create input port**
`off` (default) | `on`

Select to create input port `Temp` for the temperature.

#### Dependencies

To enable this parameter, set **Efficiency factors** to `Driveshaft torque, speed and temperature`.

**`Crown wheel (ring gear) located` — Specify crown wheel connection**
`To the left of center-line` (default) | `To the right of center-line`

Specify the crown wheel connection to the driveshaft.

**Carrier to drive shaft ratio, Ndiff — Ratio**
4 (default) | scalar

Carrier-to-driveshaft gear ratio, $N$, dimensionless.

**Carrier inertia, Jd — Inertia**
.1 (default) | scalar

Rotational inertia of the crown gear assembly, $J_d$, in kg·m^2. You can include the driveshaft inertia.

**Carrier damping, bd — Damping**
1e-3 (default) | scalar

Crown gear linear viscous damping, $b_d$, in N·m·s/rad.

**Axle 1 inertia, Jw1 — Inertia**
.1 (default) | scalar

Axle 1 rotational inertia, $J_1$, in kg·m^2.

**Axle 1 damping, bw1 — Damping**
1e-3 (default) | scalar

Axle 1 linear viscous damping, $b_1$, in N·m·s/rad.

**Axle 2 inertia, Jw2 — Inertia**
.1 (default) | scalar

Axle 2 rotational inertia, $J_2$, in kg·m^2.

**Axle 2 damping, bw2 — Damping**
1e-3 (default) | scalar

Axle 2 linear viscous damping, $b_2$, in N·m·s/rad.

**Axle 1 initial velocity, omegaw1o — Angular velocity**
0 (default) | scalar

Axle 1 initial velocity, $\omega_{o1}$, in rad/s.

**Axle 2 initial velocity, omegaw2o — Angular velocity**
0 (default) | scalar

Axle 2 initial velocity, $\omega_{o2}$, in rad/s.

#### Efficiency

**Constant efficiency factor, eta — Efficiency**
1 (default) | scalar

Constant efficiency, $\eta$.

#### Dependencies

To enable this parameter, set **Efficiency factors** to `Constant`.

### `Efficiency lookup table, eta_tbl` — Lookup table

`M`-by-`N`-by-`L` array

Dimensionless array of values for efficiency as a function of:

- `M` input torques
- `N` input speed
- `L` air temperatures

Each value specifies the efficiency for a specific combination of torque, speed, and temperature. The array size must match the dimensions defined by the torque, speed, and temperature breakpoint vectors.

#### Dependencies

To enable this parameter, set **Efficiency factors** to `Driveshaft torque, speed and temperature`.

### `Efficiency torque breakpoints, Trq_bpts` — Torque breakpoints

`[25, 50, 75, 100, 150, 200, 250]` (default) | 1-by-`M` vector

Vector of input torque, breakpoints for efficiency, in N·m.

#### Dependencies

To enable this parameter, set **Efficiency factors** to `Driveshaft torque, speed and temperature`.

### `Efficiency speed breakpoints, omega_bpts` — Speed breakpoints

`[52.4 78.5 105 131 157 183 209 262 314 419 524]` (default) | 1-by-`N` vector

Vector of speed, breakpoints for efficiency, in rad/s.

#### Dependencies

To enable this parameter, set **Efficiency factors** to `Driveshaft torque, speed and temperature`.

### `Efficiency temperature breakpoints, Temp_bpts` — Temperature breakpoints

`[290 358]` (default) | 1-by-`L` vector

Vector of ambient temperature breakpoints for efficiency, in K.

#### Dependencies

To enable this parameter, set **Efficiency factors** to `Driveshaft torque, speed and temperature`.

### `Ambient temperature, Tamb` — Ambient temperature

`297.15` (default) | `scalar`

Ambient air temperature, $T_{air}$, in K.

#### Dependencies

To enable this parameter:

- Set **Efficiency factors** to `Driveshaft torque, speed and temperature`.
- Clear **Input temperature**.

## Extended Capabilities

### C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

## See Also

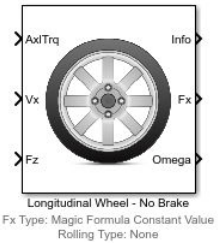
Limited Slip Differential

**Introduced in R2017a**

# Longitudinal Wheel

Longitudinal wheel with disc, drum, or mapped brake

**Library:** Powertrain Blockset / Drivetrain / Wheels
Vehicle Dynamics Blockset / Wheels and Tires




## Description

The Longitudinal Wheel block implements the longitudinal behavior of an ideal wheel. You can specify the longitudinal force and rolling resistance calculation method, and brake type. Use the block in driveline and longitudinal vehicle simulations where low frequency tire-road and braking forces are required to determine vehicle acceleration, braking, and wheel-rolling resistance. For example, you can use the block to determine the torque and power requirements for a specified drive cycle or braking event. The block is not suitable for applications that require combined lateral slip.

There are four types of Longitudinal Wheel blocks. Each block implements a different brake type.

| Block Name | Brake Type Setting | Brake Implementation |
|---|---|---|
| Longitudinal Wheel - No Brake | `None` | None |
| Longitudinal Wheel - Disc Brake | `Disc` | Brake that converts the brake cylinder pressure into a braking force. |
| Longitudinal Wheel - Drum Brake | `Drum` | Simplex drum brake that converts the applied force and brake geometry into a net braking torque. |
| Longitudinal Wheel - Mapped Brake | `Mapped` | Lookup table that is a function of the wheel speed and applied brake pressure. |

The block models longitudinal force as a function of wheel slip relative to the road surface. To calculate the longitudinal force, specify one of these **Longitudinal Force** parameters.

| Setting | Block Implementation |
|---|---|
| `Magic Formula constant value` | Magic Formula with constant coefficient for stiffness, shape, peak, and curvature. |
| `Magic Formula pure longitudinal slip` | Magic Formula with load-dependent coefficients that implement equations 4.E9 through 4.E18 in *Tire and Vehicle Dynamics*. |
| `Mapped force` | Lookup table that is a function of the normal force and wheel slip ratio. |

To calculate the rolling resistance torque, specify one of these **Rolling Resistance** parameters.

| Setting | Block Implementation |
|---|---|
| `None` | None |
| `Pressure and velocity` | Method in *Stepwise Coastdown Methodology for Measuring Tire Rolling Resistance*. The rolling resistance is a function of tire pressure, normal force, and velocity. |
| `ISO 28580` | Method specified in ISO 28580:2018, *Passenger car, truck and bus tyre rolling resistance measurement method — Single point test and correlation of measurement results*. |
| `Magic Formula` | Magic formula equations from 4.E70 in *Tire and Vehicle Dynamics*. The magic formula is an empirical equation based on fitting coefficients. |
| `Mapped torque` | Lookup table that is a function of the normal force and spin axis longitudinal velocity. |

To calculate vertical motion, specify one of these **Vertical Motion** parameters.

| Setting | Block Implementation |
|---|---|
| `None` | Block passes the applied chassis forces directly through to the rolling resistance and longitudinal force calculations. |
| `Mapped stiffness and damping` | Vertical motion depends on wheel stiffness and damping. Stiffness is a function of tire sidewall displacement and pressure. Damping is a function of tire sidewall velocity and pressure. |

### Rotational Wheel Dynamics

The block calculates the inertial response of the wheel subject to:

- Axle losses
- Brake and drive torque
- Tire rolling resistance
- Ground contact through the tire-road interface

The input torque is the summation of the applied axle torque, braking torque, and moment arising from the combined tire torque.

$$T_i = T_a - T_b + T_d$$

For the moment arising from the combined tire torque, the block implements tractive wheel forces and rolling resistance with first order dynamics. The rolling resistance has a time constant parameterized in terms of a relaxation length.

$$T_d(s) = \frac{1}{\frac{|\omega| R_e}{L_e} s + 1} (F_x R_e + M_y)$$

To calculate the rolling resistance torque, you can specify one of these **Rolling Resistance** parameters.

| Setting | Block Implementation |
|---|---|
| `None` | Block sets rolling resistance, $M_y$, to zero. |
| `Pressure and velocity` | Block uses the method in SAE *Stepwise Coastdown Methodology for Measuring Tire Rolling Resistance*. The rolling resistance is a function of tire pressure, normal force, and velocity. Specifically,<br>$M_y = R_e\{a + b\|V_x\| + cV_x^2\}\{F_z^{\beta} p_i^{\alpha}\}\tanh(4V_x)$ |
| `ISO 28580` | Block uses the method specified in ISO 28580:2018, *Passenger car, truck and bus tyre rolling resistance measurement method — Single point test and correlation of measurement results*. The method accounts for normal load, parasitic loss, and thermal corrections from test conditions. Specifically,<br>$M_y = R_e(\frac{F_z C_r}{1 + K_t(T_{amb} - T_{meas})} - F_{pl})\tanh(\omega)$ |
| `Magic Formula` | Block calculates the rolling resistance, $M_y$, using the Magic Formula equations from 4.E70 in *Tire and Vehicle Dynamics*. The magic formula is an empirical equation based on fitting coefficients. |
| `Mapped torque` | For the rolling resistance, $M_y$, the block uses a lookup table that is a function of the normal force and spin axis longitudinal velocity. |

If the brakes are enabled, the block determines the braking locked or unlocked condition based on an idealized dry clutch friction model. Based on the lock-up condition, the block implements these friction and dynamic models.

| If | Lock-Up Condition | Friction Model | Dynamic Model |
|---|---|---|---|
| $\omega \neq 0$<br>or<br>$T_S < \left\|T_i + T_f - \omega b\right\|$ | Unlocked | $T_f = T_k$<br>where,<br>$T_k = F_c R_{eff} \mu_k \tanh[4(-\omega_d)]$<br>$T_s = F_c R_{eff} \mu_s$<br>$R_{eff} = \frac{2(R_o^3 - R_i^3)}{3(R_o^2 - R_i^2)}$ | $\dot{\omega} J = -\omega b + T_i + T_o$ |
| $\omega = 0$<br>and<br>$T_S \geq \left\|T_i + T_f - \omega b\right\|$ | Locked | $T_f = T_s$ | $\omega = 0$ |

The equations use these variables.

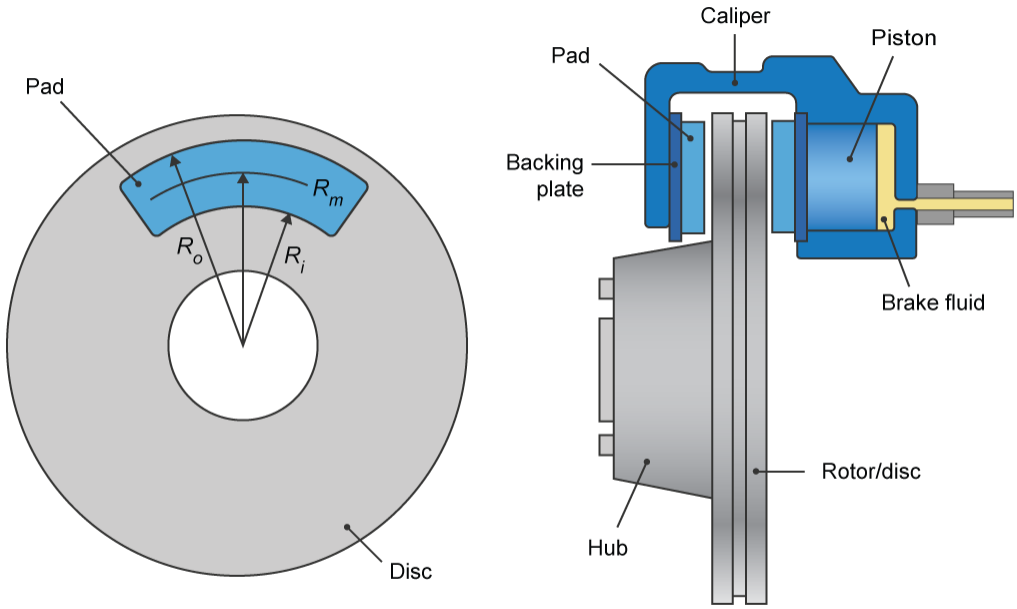
| | |
|---|---|
| $\omega$ | Wheel angular velocity |
| $a$ | Velocity-independent force component |
| $b$ | Linear velocity force component |
| $c$ | Quadratic velocity force component |
| $L_e$ | Tire relaxation length |
| $J$ | Moment of inertia |

| | |
|---|---|
| $M_y$ | Rolling resistance torque |
| $T_a$ | Applied axle torque |
| $T_b$ | Braking torque |
| $T_d$ | Combined tire torque |
| $T_f$ | Frictional torque |
| $T_i$ | Net input torque |
| $T_k$ | Kinetic frictional torque |
| $T_o$ | Net output torque |
| $T_s$ | Static frictional torque |
| $F_c$ | Applied clutch force |
| $F_x$ | Longitudinal force developed by the tire road interface due to slip |
| $R_{eff}$ | Effective clutch radius |
| $R_o$ | Annular disk outer radius |
| $R_i$ | Annular disk inner radius |
| $R_e$ | Effective tire radius while under load and for a given pressure |
| $V_x$ | Longitudinal axle velocity |
| $F_z$ | Vehicle normal force |
| $C_r$ | Rolling resistance constant |
| $T_{amb}$ | Ambient temperature |
| $T_{meas}$ | Measured temperature for rolling resistance constant |
| $F_{pl}$ | Parasitic force loss |
| $K_t$ | Thermal correction factor |
| $\alpha$ | Tire pressure exponent |
| $\beta$ | Normal force exponent |
| $p_i$ | Tire pressure |
| $\mu_s$ | Coefficient of static friction |
| $\mu_k$ | Coefficient of kinetic friction |

## Brakes

### Disc

If you specify the **Brake Type** parameter `Disc`, the block implements a disc brake. This figure shows the side and front views of a disc brake.




A disc brake converts brake cylinder pressure from the brake cylinder into force. The disc brake applies the force at the brake pad mean radius.

The block uses these equations to calculate brake torque for the disc brake.

$$T = \begin{cases} \frac{\mu P \pi B_a{}^2 R_m N_{pads}}{4} & \text{when } N \neq 0 \\ \frac{\mu_{static} P \pi B_a{}^2 R_m N_{pads}}{4} & \text{when } N = 0 \end{cases}$$

$$Rm = \frac{Ro + Ri}{2}$$

The equations use these variables.

| | |
|---|---|
| $T$ | Brake torque |
| $P$ | Applied brake pressure |
| $N$ | Wheel speed |
| $N_{pads}$ | Number of brake pads in disc brake assembly |
| $\mu_{static}$ | Disc pad-rotor coefficient of static friction |
| $\mu$ | Disc pad-rotor coefficient of kinetic friction |
| $B_a$ | Brake actuator bore diameter |
| $R_m$ | Mean radius of brake pad force application on brake rotor |
| $R_o$ | Outer radius of brake pad |

| | |
|---|---|
| $R_i$ | Inner radius of brake pad |

**Drum**

If you specify the **Brake Type** parameter `Drum`, the block implements a static (steady-state) simplex drum brake. A simplex drum brake consists of a single two-sided hydraulic actuator and two brake shoes. The brake shoes do not share a common hinge pin.

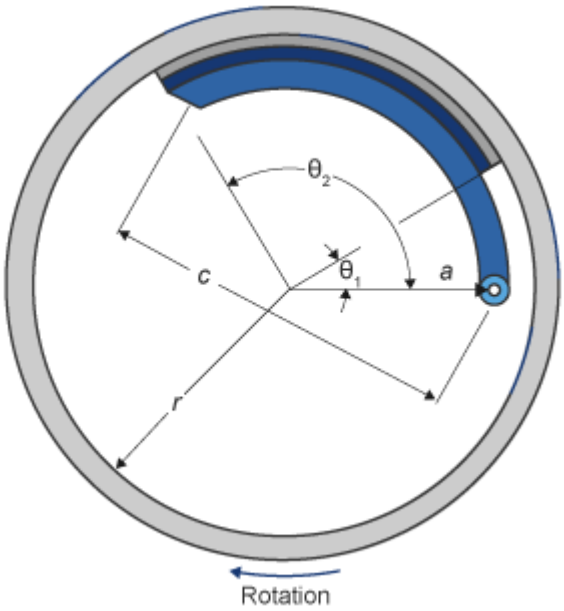
The simplex drum brake model uses the applied force and brake geometry to calculate a net torque for each brake shoe. The drum model assumes that the actuators and shoe geometry are symmetrical for both sides, allowing a single set of geometry and friction parameters to be used for both shoes.

The block implements equations that are derived from these equations in *Fundamentals of Machine Elements*.

$$T_{rshoe} = \left( \frac{\pi \mu c r (\cos\theta_2 - \cos\theta_1) B_a 2}{2\mu(2r(\cos\theta_2 - \cos\theta_1) + a(\cos^2\theta_2 - \cos^2\theta_1)) + ar(2\theta_1 - 2\theta_2 + \sin 2\theta_2 - \sin 2\theta_1)} \right) P$$

$$T_{lshoe} = \left( \frac{\pi \mu c r (\cos\theta_2 - \cos\theta_1) B_a 2}{-2\mu(2r(\cos\theta_2 - \cos\theta_1) + a(\cos^2\theta_2 - \cos^2\theta_1)) + ar(2\theta_1 - 2\theta_2 + \sin 2\theta_2 - \sin 2\theta_1)} \right) P$$

$$T = \begin{cases} T_{rshoe} + T_{lshoe} & \text{when } N \neq 0 \\ (T_{rshoe} + T_{lshoe}) \frac{\mu_{static}}{\mu} & \text{when } N = 0 \end{cases}$$




The equations use these variables.

| | |
|---|---|
| $T$ | Brake torque |
| $P$ | Applied brake pressure |
| $N$ | Wheel speed |

| | |
|---|---|
| $\mu_{static}$ | Disc pad-rotor coefficient of static friction |
| $\mu$ | Disc pad-rotor coefficient of kinetic friction |
| $T_{rshoe}$ | Right shoe brake torque |
| $T_{lshoe}$ | Left shoe brake torque |
| $a$ | Distance from drum center to shoe hinge pin center |
| $c$ | Distance from shoe hinge pin center to brake actuator connection on brake shoe |
| $r$ | Drum internal radius |
| $B_a$ | Brake actuator bore diameter |
| $\Theta_1$ | Angle from shoe hinge pin center to start of brake pad material on shoe |
| $\Theta_2$ | Angle from shoe hinge pin center to end of brake pad material on shoe |

**Mapped**

If you specify the **Brake Type** parameter `Mapped`, the block uses a lookup table to determine the brake torque.

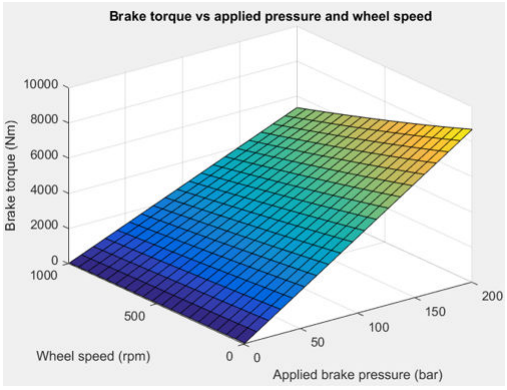
$$T = \begin{cases} f_{brake}(P, N) & \text{when } N \neq 0 \\ \left(\frac{\mu_{static}}{\mu}\right) f_{brake}(P, N) & \text{when } N = 0 \end{cases}$$

The equations use these variables.

| | |
|---|---|
| $T$ | Brake torque |
| $f_{brake}(P, N)$ | Brake torque lookup table |
| $P$ | Applied brake pressure |
| $N$ | Wheel speed |
| $\mu_{static}$ | Friction coefficient of drum pad-face interface under static conditions |
| $\mu$ | Friction coefficient of disc pad-rotor interface |

The lookup table for the brake torque, $f_{brake}(P, N)$, is a function of applied brake pressure and wheel speed, where:

- $T$ is brake torque, in N·m.
- $P$ is applied brake pressure, in bar.
- $N$ is wheel speed, in rpm.



### Longitudinal Force

To model the Longitudinal Wheel block longitudinal forces, you can use the Magic Formula. The model provides a steady-state *tire characteristic function* $F_x = f(\kappa, F_z)$, the longitudinal force $F_x$ on the tire, based on:

- Vertical load $F_z$
- Wheel slip $\kappa$

The Magic Formula model uses these variables.

| | |
|---|---|
| $\Omega$ | Wheel angular velocity |
| $r_w$ | Wheel radius |
| $V_x$ | Wheel hub longitudinal velocity |
| $r_w\Omega$ | Tire tread longitudinal velocity |
| $V_{sx} = r_w\Omega - V_x$ | Wheel slip velocity |
| $\kappa = V_{sx}/\lvert V_x\rvert$ | Wheel slip |
| $F_z$, $F_{z0}$ | Vertical load and nominal vertical load on tire |
| $F_x = f(\kappa, F_z)$ | Longitudinal force exerted on the tire at the contact point. Also a characteristic function $f$ of the tire. |

#### Magic Formula Constant Value

If you set **Longitudinal Force** to `Magic Formula constant value`, the block implements the Magic Formula as a specific form of the tire characteristic function, characterized by four dimensionless coefficients ($B$, $C$, $D$, $E$), or stiffness, shape, peak, and curvature:

$$F_x = f(\kappa, F_z) = F_z D \sin\left(C\tan^{-1}[\ \{B\kappa - E[B\kappa - \tan^{-1}(B\kappa)\ \rfloor\ \}\ ]\ )\right.$$

The slope of $f$ at $\kappa = 0$ is $BCD{\cdot}F_z$.

The coefficients are based on empirical tire data. These values are typical sets of constant Magic Formula coefficients for common road conditions.

| Surface | B | C | D | E |
|---|---|---|---|---|
| Dry tarmac | 10 | 1.9 | 1 | 0.97 |
| Wet tarmac | 12 | 2.3 | 0.82 | 1 |
| Snow | 5 | 2 | 0.3 | 1 |
| Ice | 4 | 2 | 0.1 | 1 |

#### Magic Formula Pure Longitudinal Slip

If you set **Longitudinal Force** to `Magic Formula pure longitudinal slip`, the block implements a more general Magic Formula using dimensionless coefficients that are functions of the tire load. The block implements the longitudinal force equations in Chapter 4 of *Tire and Vehicle Dynamics*, including 4.E9 through 4.E18:

$$F_{x0} = D_x \sin\left(C_x \tan^{-1}\left[ \{B_x\kappa_x - E_x[B_x\kappa_x - \tan^{-1}(B_x\kappa_x)\right]\right\}\right] ) + S_{Vx}$$

where:

$\kappa_x = \kappa + S_{Hx}$

$C_x = p_{Cx1}\lambda_{Cx}$

$D_x = \mu_x F_z \zeta_1$

$\mu_x = (p_{Dx1} + p_{Dx2}df_z)(1 + p_{px3}dp_i + p_{px4}dp_i^2)(1 - p_{Dx3}\gamma^2)\lambda^*_{\mu x}$

$E_x = (p_{Ex1} + p_{Ex2}df_z + p_{Ex3}df_z^2)[1 - p_{Ex4}\mathrm{sgn}(\kappa_x)] \lambda_{Ex}$

$K_{x\kappa} = F_z(p_{Kx1} + p_{Kx2}df_z)\exp(p_{Kx3}df_z)(1 + p_{px1}dp_i + p_{px2}dp_i^2)$

$B_x = K_{x\kappa}/(C_x D_x + \varepsilon_x)$

$S_{Hx} = p_{Hx1} + p_{Hx2}df_z$

$S_{Vx} = F_z \cdot (p_{Vx1} + p_{Vx2}df_z)\lambda_{Vx}\lambda'_{\mu x}\zeta_1$

$S_{Hx}$ and $S_{Vx}$ represent offsets to the slip and longitudinal force in the force-slip function, or horizontal and vertical offsets if the function is plotted as a curve. $\mu_x$ is the longitudinal load-dependent friction coefficient. $\varepsilon_x$ is a small number inserted to prevent division by zero as $F_z$ approaches zero.

### Vertical Dynamics

If you select no vertical degrees-of-freedom by setting **Vertical Motion** to `None`, the block passes the applied chassis forces directly through to the rolling resistance and longitudinal force calculations.

If you set **Vertical Motion** to `Mapped stiffness and damping`, the vertical motion depends on wheel stiffness and damping. Stiffness is a function of tire sidewall displacement and pressure. Damping is a function of tire sidewall velocity and pressure.

$$Fztire(z, \dot{z}, P_{tire}) = F_{zk}(z, P_{tire}) + F_{zb}(\dot{z}, P_{tire})$$

The block determines the vertical response using this differential equation.

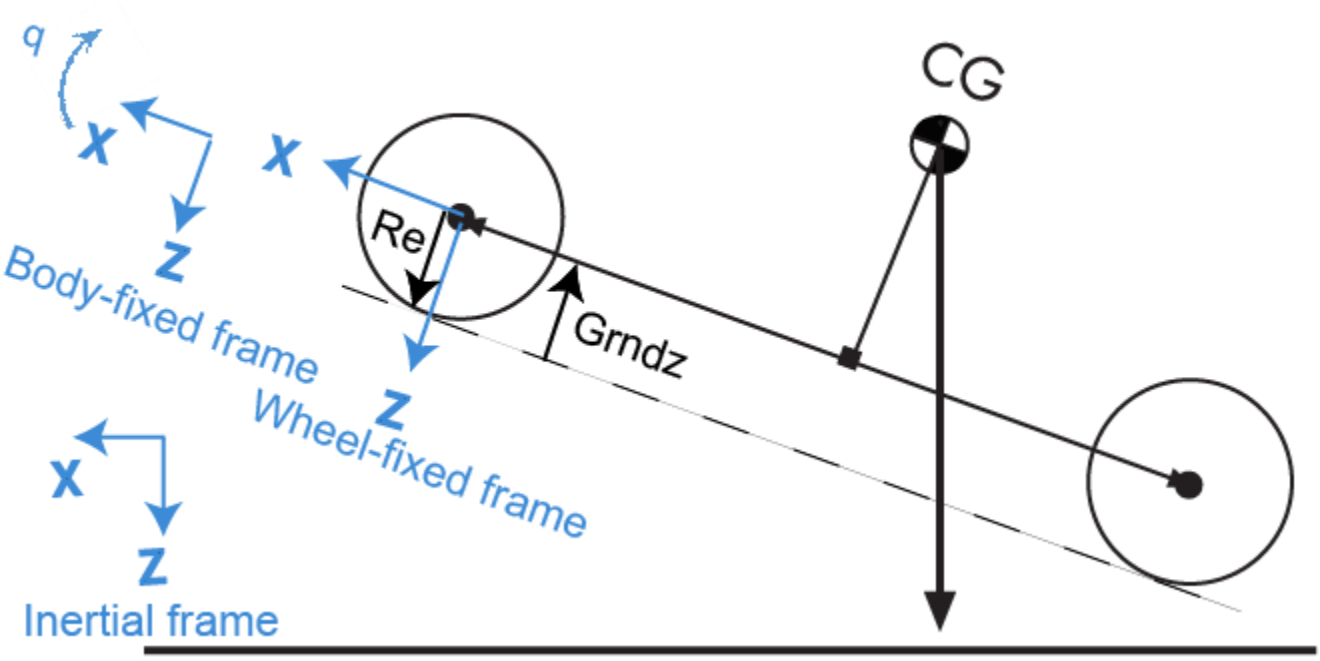
$$\ddot{z}m = Fztire - F_z - mg$$

When you disable the vertical degree-of-freedom, the input normal force from the vehicle passes directly to the longitudinal and rolling force calculations.

$$\ddot{z} = \dot{z} = m = 0$$

$$Fztire = mg$$

The block uses the wheel-fixed frame to resolve the vertical forces.




The equations use these variables.

| | |
|---|---|
| $Fztire$ | Tire normal force along the wheel-fixed $z$-axis |
| $m$ | Axle mass |
| $F_{zk}$ | Tire normal force due to wheel stiffness along the wheel-fixed $z$-axis |
| $F_{zb}$ | Tire normal force due to wheel damping along the wheel-fixed $z$-axis |
| $F_z$ | Suspension or vehicle normal force along the wheel-fixed $z$-axis |
| $P_{Tire}$ | Tire pressure |
| $z, \dot{z}, \ddot{z}$ | Tire displacement, velocity, and acceleration, respectively, along the wheel-fixed $z$-axis |

### Power Accounting

For the power accounting, the block implements these equations.

| Bus Signal | | | Description | Equations |
|---|---|---|---|---|
| `PwrInfo` | `PwrTrnsfrd` — Power transferred between blocks | `PwrRoad` | Tractive power applied from the axle | $P_{road} = F_x V_x$ |
| | | `PwrAxlTrq` | External torque applied by the axle to the wheel | $P_T = T\omega$ |

| Bus Signal | | | Description | Equations |
|---|---|---|---|---|
| | • Positive signals indicate flow into block<br>• Negative signals indicate flow out of block | `PwrFz` | Vertical force applied to the wheel by the vehicle or suspension | $P_{Fz} = F_z\dot{z}$ |
| | `PwrNotTrnsfrd` — Power crossing the block boundary, but not transferred<br>• Positive signals indicate an input<br>• Negative signals indicate a loss | `PwrSlip` | Tractive power loss | $P_\kappa = F_xV_x + ( - F_{cp}R_e + M_y)\omega$ |
| | | `PwrMyRoll` | Rolling resistance power | $P_{My} = M_y\omega$ |
| | | `PwrMyBrk` | Braking power | $P_{brk} = M_{brk}\omega$ |
| | | `PwrMyb` | Rolling viscous damping loss | $P_b = -b\omega^2$ |
| | | `PwrFzDamp` | Vertical damping power | $P_{Fzb} = F_{zb}\dot{z}$ |
| | `PwrStored` — Stored energy rate of change<br>• Positive signals indicate an increase<br>• Negative signals indicate a decrease | `PwrStoredzdot` | Rate of change of vertical kinetic energy | $P_{\dot{z}} = m\ddot{z}\dot{z}$ |
| | | `PwrStoredq` | Rate of change of rotational kinetic energy | $P_\omega = I_{yy}\dot{\omega}\omega$ |
| | | `PwrStoredFsFzSprng` | Rate of change of stored sidewall potential energy | $P_{Fzk} = F_{zk}\dot{z}_x$ |
| | | `PwrStoredGrvty` | Rate of change of gravitational potential energy | $P_g = -mg\dot{Z}$ |

The equations use these variables.

| | |
|---|---|
| $\omega$ | Wheel angular velocity |
| $b$ | Linear velocity force component |
| $F_x$ | Longitudinal force developed by the tire road interface due to slip |
| $F_{cp}$ | Tire slip force at contact patch |
| $F_z$ | Vehicle normal force |
| $F_{zb}$ | Tire normal force due to wheel damping |
| $F_{zk}$ | Tire normal force due to wheel stiffness |
| $I_{yy}$ | Wheel rotational inertia |
| $M_{brk}$ | Braking moment |
| $M_y$ | Rolling resistance torque |
| $R_e$ | Effective tire radius while under load and for a given pressure |
| $T$ | Axle torque applied on wheel |
| $V_x$ | Longitudinal axle velocity |

| | |
|---|---|
| $z, \dot{z}, \ddot{z}$ | Tire displacement, velocity, and acceleration, respectively |
| $\omega$ | Wheel angular velocity |
| $\dot{Z}$ | Vehicle vertical velocity along the vehicle-fixed $z$-axis |

## Ports

### Input

#### `BrkPrs` — Brake pressure

`scalar`

Brake pressure, in Pa.

##### Dependencies

To enable this port, for the **Brake Type** parameter, specify one of these types:

- `Disc`
- `Drum`
- `Mapped`

#### `AxlTrq` — Axle torque

`scalar`

Axle torque, $T_a$, about wheel spin axis, in N·m.

#### `Vx` — Velocity

`scalar`

Axle longitudinal velocity along vehicle(body)-fixed $x$-axis, in m/s.

#### `Fz` — Normal force

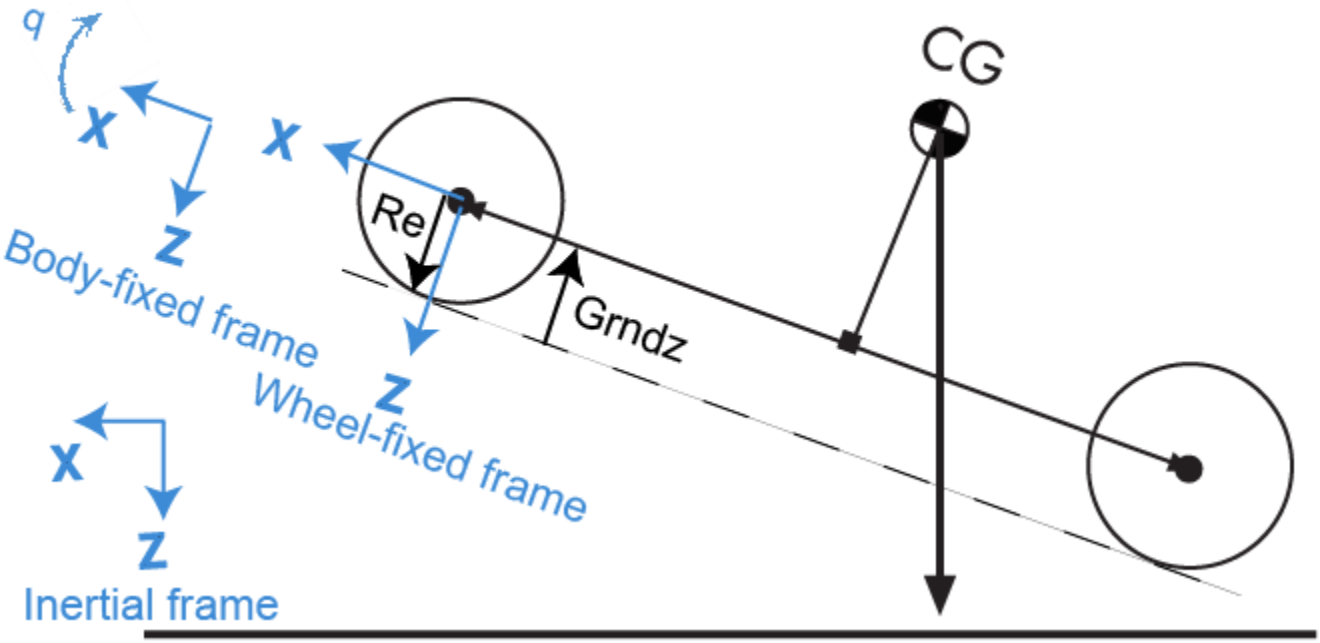
`scalar`

Absolute value of suspension or vehicle normal force along body-fixed $z$-axis, in N.

#### `Gnd` — Ground displacement

`scalar`

Ground displacement, `Grndz`, along negative wheel-fixed $z$-axis, in m.




**Dependencies**

To create `Gnd`:

- Set **Vertical Motion** to `Mapped stiffness and damping`.
- On the **Vertical** pane, select **Input ground displacement**.

### `lam_mux` — Friction scaling factor

`scalar`

Longitudinal friction scaling factor, dimensionless.

**Dependencies**

To enable this port, select **Input friction scale factor**.

### `TirePrs` — Tire pressure

`scalar`

Tire pressure, in Pa.

**Dependencies**

To enable this port:

- Set one of these parameters:
  - **Longitudinal Force** to `Magic Formula pure longitudinal slip`.
  - **Rolling Resistance** to `Pressure and velocity` or `Magic Formula`.
  - **Vertical Motion** to `Mapped stiffness and damping`.

- On the **Wheel Dynamics** pane, select **Input tire pressure**.

### `Tamb` — Ambient temperature

`scalar`

Ambient temperature, $T_{amb}$, in K.

#### Dependencies

To enable this port:

1. Set **Rolling Resistance** to `ISO 28580`.
2. On the **Rolling Resistance** pane, select to **Input ambient temperature**.

### Output

### `Info` — Bus signal

bus

Bus signal containing these block calculations.

| Signal | Description | Units |
|---|---|---|
| `AxlTrq` | Axle torque about body-fixed *y*-axis | N·m |
| `Omega` | Wheel angular velocity about body-fixed *y*-axis | rad/s |
| `Omegadot` | Wheel angular acceleration about body-fixed *y*-axis | rad/s^2 |
| `Fx` | Longitudinal vehicle force along body-fixed *x*-axis | N |
| `Fz` | Vertical vehicle force along body-fixed *z*-axis | N |
| `Fzb` | Tire normal force due to wheel damping along the wheel-fixed *z*-axis | N |
| `Fzk` | Tire normal force due to wheel stiffness along the wheel-fixed *z*-axis | N |
| `My` | Rolling resistance torque about body-fixed *y*-axis | N·m |
| `Myb` | Rolling resistance torque due to damping about body-fixed *y*-axis | N·m |
| `Kappa` | Slip ratio | NA |
| `Vx` | Vehicle longitudinal velocity along body-fixed *x*-axis | m/s |
| `Re` | Wheel effective radius along wheel-fixed *z*-axis | m |

| Signal | | | Description | Units |
|---|---|---|---|---|
| BrkTrq | | | Brake torque about body-fixed *y*-axis | N·m |
| BrkPrs | | | Brake pressure | Pa |
| z | | | Wheel vertical deflection along wheel-fixed *z*-axis | m |
| zdot | | | Wheel vertical velocity along wheel-fixed *z*-axis | m/s |
| zddot | | | Wheel vertical acceleration along wheel-fixed *z*-axis | m/s^2 |
| Gndz | | | Ground displacement along negative of wheel-fixed *z*-axis (positive input produces wheel lift) | m |
| GndFz | | | Vertical wheel force on ground along negative of wheel-fixed *z*-axis | N |
| TirePrs | | | Tire pressure | Pa |
| Fpatch | | | Tractive power applied from the axle | |
| PwrInfo | PwrTrnsfrd | PwrRoad | External torque applied by the axle to the wheel | W |
| | | PwrAxlTrq | Vertical force applied to the wheel by the vehicle or suspension | W |
| | | PwrFz | Tractive power loss | W |
| | PwrNotTrnsfrd | PwrSlip | Rolling resistance power | W |
| | | PwrMyRoll | Braking power | W |
| | | PwrMyBrk | Rolling viscous damping loss | W |
| | | PwrMyb | Vertical damping power | W |
| | | PwrFzDamp | Rate of change of vertical kinetic energy | W |
| | PwrStored | PwrStoredzdot | Rate of change of rotational kinetic energy | W |
| | | PwrStoredq | Rate of change of stored sidewall potential energy | W |
| | | PwrStoredFsFzSprng | Rate of change of gravitational potential energy | W |
| | | PwrStoredGrvty | Tractive power applied from the axle | W |

**Fx — Longitudinal axle force**
scalar

Longitudinal force acting on axle, along body-fixed *x*-axis, in N. Positive force acts to move the vehicle forward.

#### `Omega` — Wheel angular velocity

`scalar`

Wheel angular velocity, about body-fixed *y*-axis, in rad/s.

#### `z` — Wheel vertical deflection

`scalar`

Wheel vertical deflection along wheel-fixed *z*-axis, in m.

##### Dependencies

To enable this port, set **Vertical Motion** to `Mapped stiffness and damping`.

#### `zdot` — Wheel vertical velocity

`scalar`

Wheel vertical velocity along wheel-fixed *z*-axis, in m/s.

##### Dependencies

To enable this port, set **Vertical Motion** to `Mapped stiffness and damping`.

## Parameters

### Block Options

#### `Longitudinal Force` — Select type

`Magic Formula constant value` (default) | `Magic Formula pure longitudinal slip` | `Mapped force`

The block models longitudinal force as a function of wheel slip relative to the road surface. To calculate the longitudinal force, specify one of these **Longitudinal Force** parameters.

| Setting | Block Implementation |
|---|---|
| `Magic Formula constant value` | Magic Formula with constant coefficient for stiffness, shape, peak, and curvature. |
| `Magic Formula pure longitudinal slip` | Magic Formula with load-dependent coefficients that implement equations 4.E9 through 4.E18 in *Tire and Vehicle Dynamics*. |
| `Mapped force` | Lookup table that is a function of the normal force and wheel slip ratio. |

**Dependencies**

| Selecting | Enables These Parameters |
|---|---|
| `Magic Formula constant value` | **Pure longitudinal peak factor, Dx**<br>**Pure longitudinal shape factor, Cx**<br>**Pure longitudinal stiffness factor, Bx**<br>**Pure longitudinal curvature factor, Ex** |

| Selecting | Enables These Parameters |
|---|---|
| `Magic Formula pure longitudinal slip` | **Cfx shape factor, PCX1**<br>**Longitudinal friction at nominal normal load, PDX1**<br>**Frictional variation with load, PDX2**<br>**Frictional variation with camber, PDX3**<br>**Longitudinal curvature at nominal normal load, PEX1**<br>**Variation of curvature factor with load, PEX2**<br>**Variation of curvature factor with square of load, PEX3**<br>**Longitudinal curvature factor with slip, PEX4**<br>**Longitudinal slip stiffness at nominal normal load, PKX1**<br>**Variation of slip stiffness with load, PKX2**<br>**Slip stiffness exponent factor, PKX3**<br>**Horizontal shift in slip ratio at nominal normal load, PHX1**<br>**Variation of horizontal slip ratio with load, PHX2**<br>**Vertical shift in load at nominal normal load, PVX1**<br>**Variation of vertical shift with load, PVX2**<br>**Linear variation of longitudinal slip stiffness with tire pressure, PPX1**<br>**Quadratic variation of longitudinal slip stiffness with tire pressure, PPX2**<br>**Linear variation of peak longitudinal friction with tire pressure, PPX3**<br>**Quadratic variation of peak longitudinal friction with tire pressure, PPX4**<br>**Linear variation of longitudinal slip stiffness with tire pressure, PPX1**<br>**Slip speed decay function scaling factor, lam_muV**<br>**Brake slip stiffness scaling factor, lam_Kxkappa**<br>**Longitudinal shape scaling factor, lam_Cx**<br>**Longitudinal curvature scaling factor, lam_Ex** |

| Selecting | Enables These Parameters |
|---|---|
| | **Longitudinal horizontal shift scaling factor, lam_Hx**<br><br>**Longitudinal vertical shift scaling factor, lam_Vx** |
| `Mapped force` | **Slip ratio breakpoints, kappaFx**<br><br>**Normal force breakpoints, FzFx**<br><br>**Longitudinal force map, FxMap** |

**`Rolling Resistance` — Select type**
`None` (default) | `Pressure and velocity` | `ISO 28580` | `Magic Formula` | `Mapped torque`

To calculate the rolling resistance torque, specify one of these **Rolling Resistance** parameters.

| Setting | Block Implementation |
|---|---|
| `None` | None |
| `Pressure and velocity` | Method in *Stepwise Coastdown Methodology for Measuring Tire Rolling Resistance*. The rolling resistance is a function of tire pressure, normal force, and velocity. |
| `ISO 28580` | Method specified in ISO 28580:2018, *Passenger car, truck and bus tyre rolling resistance measurement method — Single point test and correlation of measurement results*. |
| `Magic Formula` | Magic formula equations from 4.E70 in *Tire and Vehicle Dynamics*. The magic formula is an empirical equation based on fitting coefficients. |
| `Mapped torque` | Lookup table that is a function of the normal force and spin axis longitudinal velocity. |

**Dependencies**

| Selecting | Parameters |
|---|---|
| `Pressure and velocity` | **Velocity independent force coefficient, aMy**<br><br>**Linear velocity force component, bMy**<br><br>**Quadratic velocity force component, cMy**<br><br>**Tire pressure exponent, alphaMy**<br><br>**Normal force exponent, betaMy** |

| Selecting | Parameters |
|---|---|
| `ISO 28580` | **Parasitic losses force, Fpl**<br>**Rolling resistance constant, Cr**<br>**Thermal correction factor, Kt**<br>**Measured temperature, Tmeas**<br>**Parasitic losses force, Fpl**<br>**Ambient temperature, Tamb** |
| `Magic Formula` | **Rolling resistance torque coefficient, QSY**<br>**Longitudinal force rolling resistance coefficient, QSY2**<br>**Linear rotational speed rolling resistance coefficient, QSY3**<br>**Quartic rotational speed rolling resistance coefficient, QSY4**<br>**Camber squared rolling resistance torque, QSY5**<br>**Load based camber squared rolling resistance torque, QSY6**<br>**Normal load rolling resistance coefficient, QSY7**<br>**Pressure load rolling resistance coefficient, QSY8**<br>**Rolling resistance scaling factor, lam_My** |
| `Mapped torque` | **Spin axis velocity breakpoints, VxMy**<br>**Normal force breakpoints, FzMy**<br>**Rolling resistance torque map, MyMap** |

**`Brake Type` — Select type**
`None` | `Disc` | `Drum` | `Mapped`

There are four types of Longitudinal Wheel blocks. Each block implements a different brake type.

| Block Name | Brake Type Setting | Brake Implementation |
|---|---|---|
| Longitudinal Wheel - No Brake | `None` | None |
| Longitudinal Wheel - Disc Brake | `Disc` | Brake that converts the brake cylinder pressure into a braking force. |
| Longitudinal Wheel - Drum Brake | `Drum` | Simplex drum brake that converts the applied force and brake geometry into a net braking torque. |

| Block Name | Brake Type Setting | Brake Implementation |
|---|---|---|
| Longitudinal Wheel - Mapped Brake | `Mapped` | Lookup table that is a function of the wheel speed and applied brake pressure. |

**`Vertical Motion` — Select type**
`None` (default) | `Mapped stiffness and damping`

To calculate vertical motion, specify one of these **Vertical Motion** parameters.

| Setting | Block Implementation |
|---|---|
| `None` | Block passes the applied chassis forces directly through to the rolling resistance and longitudinal force calculations. |
| `Mapped stiffness and damping` | Vertical motion depends on wheel stiffness and damping. Stiffness is a function of tire sidewall displacement and pressure. Damping is a function of tire sidewall velocity and pressure. |

| Selecting | Enables These Parameters | Creates These Output Ports |
|---|---|---|
| `Mapped stiffness and damping` | **Wheel and unsprung mass, m**<br>**Initial deflection, zo**<br>**Initial velocity, zdoto**<br>**Gravitational acceleration, g**<br>**Vertical deflection breakpoints, zFz**<br>**Pressure breakpoints, pFz**<br>**Force due to deflection, Fzz**<br>**Vertical velocity breakpoints, zdotFz**<br>**Force due to velocity, Fzzdot**<br>**Ground displacement, Gndz**<br>**Input ground displacement** | `z`<br>`zdot` |

**`Longitudinal scaling factor, lam_x` — Friction scaling factor**
`1` (default)

Longitudinal friction scaling factor, dimensionless.

**Dependencies**

To enable this parameter, clear **Input friction scale factor**.

**`Input friction scale factor` — Selection**
`Off` (default)

Create input port for longitudinal friction scaling factor.

#### Dependencies

Selecting this parameter:

- Creates input port `lam_mux`.
- Disables parameter **Longitudinal scaling factor, lam_x**.

### Wheel Dynamics

**`Axle viscous damping coefficient, br` — Damping**
`0.001` (default) | `scalar`

Axle viscous damping coefficient, *br*, in N·m·s/rad.

**`Wheel inertia, Iyy` — Inertia**
`0.8` (default) | `scalar`

Wheel inertia, in kg·m^2.

**`Wheel initial angular velocity, omegao` — Wheel speed**
`0` (default) | `scalar`

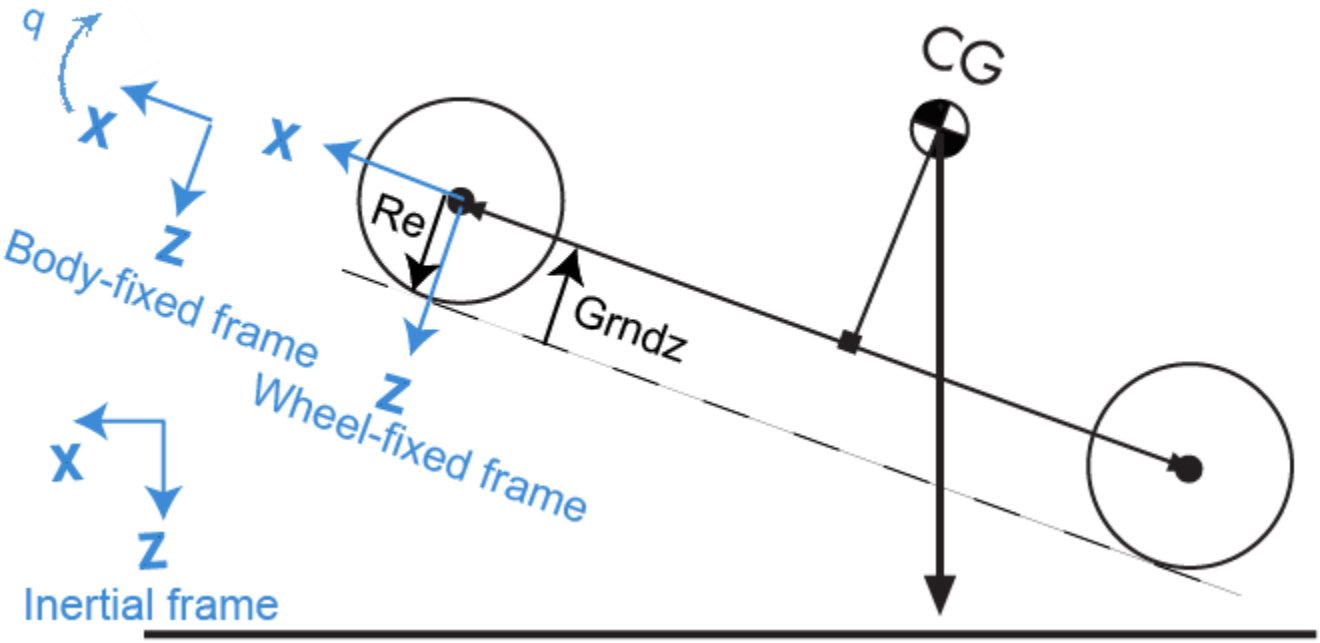
Initial angular velocity of wheel, along body-fixed *y*-axis, in rad/s.

**`Relaxation length, Lrel` — Relaxation length**
`0.5` (default) | `scalar`

Wheel relaxation length, in m.

**`Loaded radius, Re` — Loaded radius**
`0.3` (default) | `scalar`

Loaded wheel radius, Re, in m.




**`Unloaded radius, UNLOADED_RADIUS` — Unloaded radius**
`0.4` (default) | `scalar`

Unloaded wheel radius, in m.

**Dependencies**

To create this parameter, set **Rolling Resistance** to `Pressure and velocity` or `Magic Formula`.

**`Nominal longitudinal speed, LONGVL` — Speed**
`16` (default) | `scalar`

Nominal longitudinal speed along body-fixed *x*-axis, in m/s.

**Dependencies**

To enable this parameter, set **Longitudinal Force** to `Magic Formula pure longitudinal slip`.

**`Nominal camber angle, gamma` — Camber**
`0` (default) | `scalar`

Nominal camber angle, in rad.

**Dependencies**

To enable this parameter, set either:

- **Longitudinal Force** to `Magic Formula pure longitudinal slip`.
- **Rolling Resistance** to `Magic Formula`.

### Nominal pressure, NOMPRES — Pressure

`220000` (default) | `scalar`

Nominal pressure, in Pa.

#### Dependencies

To enable this parameter, set either:

- **Longitudinal Force** to `Magic Formula pure longitudinal slip`.
- **Rolling Resistance** to `Magic Formula`.

### Pressure, press — Pressure

`220000` (default) | `scalar`

Pressure, in Pa.

#### Dependencies

To enable this parameter:

- Set one of these:
  - **Longitudinal Force** to `Magic Formula pure longitudinal slip`.
  - **Rolling Resistance** to `Pressure and velocity` or `Magic Formula`.
  - **Vertical Motion** to `Mapped stiffness and damping`.
- On the **Wheel Dynamics** pane, clear **Input tire pressure**.

## Longitudinal

#### Magic Formula Constant Value

### Pure longitudinal peak factor, Dx — Factor

`1` (default) | `scalar`

Pure longitudinal peak factor, dimensionless.

The coefficients are based on empirical tire data. These values are typical sets of constant Magic Formula coefficients for common road conditions.

| Surface | B | C | D | E |
|---|---|---|---|---|
| Dry tarmac | 10 | 1.9 | 1 | 0.97 |
| Wet tarmac | 12 | 2.3 | 0.82 | 1 |
| Snow | 5 | 2 | 0.3 | 1 |
| Ice | 4 | 2 | 0.1 | 1 |

#### Dependencies

To create this parameter, select the **Longitudinal Force** parameter `Magic Formula constant value`.

### Pure longitudinal shape factor, Cx — Factor

`1.65` (default) | `scalar`

Pure longitudinal shape factor, dimensionless.

The coefficients are based on empirical tire data. These values are typical sets of constant Magic Formula coefficients for common road conditions.

| Surface | B | C | D | E |
|---|---|---|---|---|
| Dry tarmac | 10 | 1.9 | 1 | 0.97 |
| Wet tarmac | 12 | 2.3 | 0.82 | 1 |
| Snow | 5 | 2 | 0.3 | 1 |
| Ice | 4 | 2 | 0.1 | 1 |

#### Dependencies

To create this parameter, select the **Longitudinal Force** parameter `Magic Formula constant value`.

### `Pure longitudinal stiffness factor, Bx` — Factor

`10` (default) | `scalar`

Pure longitudinal stiffness factor, dimensionless.

The coefficients are based on empirical tire data. These values are typical sets of constant Magic Formula coefficients for common road conditions.

| Surface | B | C | D | E |
|---|---|---|---|---|
| Dry tarmac | 10 | 1.9 | 1 | 0.97 |
| Wet tarmac | 12 | 2.3 | 0.82 | 1 |
| Snow | 5 | 2 | 0.3 | 1 |
| Ice | 4 | 2 | 0.1 | 1 |

#### Dependencies

To create this parameter, select the **Longitudinal Force** parameter `Magic Formula constant value`.

### `Pure longitudinal curvature factor, Ex` — Factor

`0.01` (default) | `scalar`

Pure longitudinal curvature factor, dimensionless.

The coefficients are based on empirical tire data. These values are typical sets of constant Magic Formula coefficients for common road conditions.

| Surface | B | C | D | E |
|---|---|---|---|---|
| Dry tarmac | 10 | 1.9 | 1 | 0.97 |
| Wet tarmac | 12 | 2.3 | 0.82 | 1 |
| Snow | 5 | 2 | 0.3 | 1 |
| Ice | 4 | 2 | 0.1 | 1 |

#### Dependencies

To create this parameter, select the **Longitudinal Force** parameter `Magic Formula constant value`.

#### Magic Formula Pure Longitudinal Slip

**`Cfx shape factor, PCX1` — Factor**
`1.6` (default) | `scalar`

Cfx shape factor, PCX1, dimensionless.

#### Dependencies

To create this parameter, select the **Longitudinal Force** parameter `Magic Formula pure longitudinal slip`.

**`Longitudinal friction at nominal normal load, PDX1` — Factor**
`1` (default) | `scalar`

Longitudinal friction at nominal normal load, PDX1, dimensionless.

#### Dependencies

To create this parameter, select the **Longitudinal Force** parameter `Magic Formula pure longitudinal slip`.

**`Frictional variation with load, PDX2` — Factor**
`-0.08` (default) | `scalar`

Frictional variation with load, PDX2, dimensionless.

#### Dependencies

To create this parameter, select the **Longitudinal Force** parameter `Magic Formula pure longitudinal slip`.

**`Frictional variation with camber, PDX3` — Factor**
`0` (default) | `scalar`

Frictional variation with camber, PDX3, 1/rad^2.

#### Dependencies

To create this parameter, select the **Longitudinal Force** parameter `Magic Formula pure longitudinal slip`.

**`Longitudinal curvature at nominal normal load, PEX1` — Factor**
`0.112` (default) | `scalar`

Longitudinal curvature at nominal normal load, PEX1, dimensionless.

#### Dependencies

To create this parameter, select the **Longitudinal Force** parameter `Magic Formula pure longitudinal slip`.

**`Variation of curvature factor with load, PEX2` — Factor**
`0.313` (default) | `scalar`

Variation of curvature factor with load, PEX2, dimensionless.

**Dependencies**

To create this parameter, select the **Longitudinal Force** parameter `Magic Formula pure longitudinal slip`.

**`Variation of curvature factor with square of load, PEX3` — Factor**
`0` (default) | `scalar`

Variation of curvature factor with square of load, PEX3, dimensionless.

**Dependencies**

To create this parameter, select the **Longitudinal Force** parameter `Magic Formula pure longitudinal slip`.

**`Longitudinal curvature factor with slip, PEX4` — Factor**
`0.0016` (default) | `scalar`

Longitudinal curvature factor with slip, PEX4, dimensionless.

**Dependencies**

To create this parameter, select the **Longitudinal Force** parameter `Magic Formula pure longitudinal slip`.

**`Longitudinal slip stiffness at nominal normal load, PKX1` — Factor**
`21.7` (default) | `scalar`

Longitudinal slip stiffness at nominal normal load, PKX1, dimensionless.

**Dependencies**

To create this parameter, select the **Longitudinal Force** parameter `Magic Formula pure longitudinal slip`.

**`Variation of slip stiffness with load, PKX2` — Factor**
`13.77` (default) | `scalar`

Variation of slip stiffness with load, PKX2, dimensionless.

**Dependencies**

To create this parameter, select the **Longitudinal Force** parameter `Magic Formula pure longitudinal slip`.

**`Slip stiffness exponent factor, PKX3` — Factor**
`-0.412` (default) | `scalar`

Slip stiffness exponent factor, PKX3, dimensionless.

**Dependencies**

To create this parameter, select the **Longitudinal Force** parameter `Magic Formula pure longitudinal slip`.

**Horizontal shift in slip ratio at nominal normal load, PHX1 — Factor**
`2.1585E-4` (default) | `scalar`

Horizontal shift in slip ratio at nominal normal load, PHX1, dimensionless.

**Dependencies**

To create this parameter, select the **Longitudinal Force** parameter `Magic Formula pure longitudinal slip`.

**Variation of horizontal slip ratio with load, PHX2 — Factor**
`0.00115` (default) | `scalar`

Variation of horizontal slip ratio with load, PHX2, dimensionless.

**Dependencies**

To create this parameter, select the **Longitudinal Force** parameter `Magic Formula pure longitudinal slip`.

**Vertical shift in load at nominal normal load, PVX1 — Factor**
`1.5973E-5` (default) | `scalar`

Vertical shift in load at nominal normal load, PVX1, dimensionless.

**Dependencies**

To create this parameter, select the **Longitudinal Force** parameter `Magic Formula pure longitudinal slip`.

**Variation of vertical shift with load, PVX2 — Factor**
`1.043E-4` (default) | `scalar`

Variation of vertical shift with load, PVX2, dimensionless.

**Dependencies**

To create this parameter, select the **Longitudinal Force** parameter `Magic Formula pure longitudinal slip`.

**Linear variation of longitudinal slip stiffness with tire pressure, PPX1 — Factor**
`-0.3489` (default) | `scalar`

Linear variation of longitudinal slip stiffness with tire pressure, PPX1, dimensionless.

**Dependencies**

To create this parameter, select the **Longitudinal Force** parameter `Magic Formula pure longitudinal slip`.

**Quadratic variation of longitudinal slip stiffness with tire pressure, PPX2 — Factor**
`0.382` (default) | `scalar`

Quadratic variation of longitudinal slip stiffness with tire pressure, PPX2, dimensionless.

#### Dependencies

To create this parameter, select the **Longitudinal Force** parameter `Magic Formula pure longitudinal slip`.

### `Linear variation of peak longitudinal friction with tire pressure, PPX3` — Factor

`-0.09634` (default) | `scalar`

Linear variation of peak longitudinal friction with tire pressure, PPX3, dimensionless.

#### Dependencies

To create this parameter, select the **Longitudinal Force** parameter `Magic Formula pure longitudinal slip`.

### `Quadratic variation of peak longitudinal friction with tire pressure, PPX4` — Factor

`0.06447` (default) | `scalar`

Quadratic variation of peak longitudinal friction with tire pressure, PPX4, dimensionless.

#### Dependencies

To create this parameter, select the **Longitudinal Force** parameter `Magic Formula pure longitudinal slip`.

### `Slip speed decay function scaling factor, lam_muV` — Factor

`1` (default) | `scalar`

Slip speed decay function scaling factor, lam_muV, dimensionless.

#### Dependencies

To create this parameter, select the **Longitudinal Force** parameter `Magic Formula pure longitudinal slip`.

### `Brake slip stiffness scaling factor, lam_Kxkappa` — Factor

`1` (default) | `scalar`

Brake slip stiffness scaling factor, lam_Kxkappa, dimensionless.

#### Dependencies

To create this parameter, select the **Longitudinal Force** parameter `Magic Formula pure longitudinal slip`.

### `Longitudinal shape scaling factor, lam_Cx` — Factor

`1` (default) | `scalar`

Longitudinal shape scaling factor, lam_Cx, dimensionless.

#### Dependencies

To create this parameter, select the **Longitudinal Force** parameter `Magic Formula pure longitudinal slip`.

**Longitudinal curvature scaling factor, lam_Ex — Factor**
0 (default) | scalar

Longitudinal curvature scaling factor, lam_Ex, dimensionless.

**Dependencies**

To create this parameter, select the **Longitudinal Force** parameter `Magic Formula pure longitudinal slip`.

**Longitudinal horizontal shift scaling factor, lam_Hx — Factor**
1 (default) | scalar

Longitudinal horizontal shift scaling factor, lam_Hx, dimensionless.

**Dependencies**

To create this parameter, select the **Longitudinal Force** parameter `Magic Formula pure longitudinal slip`.

**Longitudinal vertical shift scaling factor, lam_Vx — Factor**
1 (default) | scalar

Longitudinal vertical shift scaling factor, lam_Vx, dimensionless.

**Dependencies**

To create this parameter, select the **Longitudinal Force** parameter `Magic Formula pure longitudinal slip`.

**Mapped Force**

**Slip ratio breakpoints, kappaFx — Breakpoints**
vector

Slip ratio breakpoints, dimensionless.

**Dependencies**

To create this parameter, select the **Longitudinal Force** parameter `Mapped force`.

**Normal force breakpoints, FzFx — Breakpoints**
vector

Normal force breakpoints, N.

**Dependencies**

To create this parameter, select the **Longitudinal Force** parameter `Mapped force`.

**Longitudinal force map, FxMap — Lookup table**
array

Longitudinal force versus slip ratio and normal force, N.

**Dependencies**

To create this parameter, select the **Longitudinal Force** parameter `Mapped force`.

### Rolling Resistance

#### Pressure and Velocity

**`Velocity independent force coefficient, aMy` — Force coefficient**
`8e-4` (default) | `scalar`

Velocity-independent force coefficient, $a$, in s/m.

#### Dependencies

To create this parameter, select the **Rolling Resistance** parameter `Pressure and velocity`.

**`Linear velocity force component, bMy` — Force component**
`.001` (default) | `scalar`

Linear velocity force component, $b$, in s/m.

#### Dependencies

To create this parameter, select the **Rolling Resistance** parameter `Pressure and velocity`.

**`Quadratic velocity force component, cMy` — Force component**
`1.6e-4` (default) | `scalar`

Quadratic velocity force component, $c$, in s^2/m^2.

#### Dependencies

To create this parameter, select the **Rolling Resistance** parameter `Pressure and velocity`.

**`Tire pressure exponent, alphaMy` — Pressure exponent**
`-0.003` (default) | `scalar`

Tire pressure exponent, $\alpha$, dimensionless.

#### Dependencies

To create this parameter, select the **Rolling Resistance** parameter `Pressure and velocity`.

**`Normal force exponent, betaMy` — Force exponent**
`0.97` (default) | `scalar`

Normal force exponent, $\beta$, dimensionless.

#### Dependencies

To create this parameter, select the **Rolling Resistance** parameter `Pressure and velocity`.

#### ISO 28580

**`Parasitic losses force, Fpl` — Force loss**
`10` (default) | `scalar`

Parasitic force loss, $F_{pl}$, in N.

#### Dependencies

To create this parameter, select the **Rolling Resistance** parameter `ISO 28580`.

**`Rolling resistance constant, Cr` — Constant**
`1e-3` (default) | `scalar`

Rolling resistance constant, $C_r$, in N/kN. ISO 28580 specifies the rolling resistance unit as one newton of tractive resistance for every kilonewtons of normal load.

**Dependencies**

To create this parameter, select the **Rolling Resistance** parameter `ISO 28580`.

**`Thermal correction factor, Kt` — Correction factor**
`.008` (default) | `scalar`

Thermal correction factor, $K_t$, in 1/degC.

**Dependencies**

To create this parameter, select the **Rolling Resistance** parameter `ISO 28580`.

**`Measured temperature, Tmeas` — Temperature**
`298.15` (default) | `scalar`

Measured temperature, $T_{meas}$, in K.

**Dependencies**

To create this parameter, select the **Rolling Resistance** parameter `ISO 28580`.

**`Ambient temperature, Tamb` — Temperature**
`298.15` (default) | `scalar`

Measured temperature, $T_{amb}$, in K.

**Dependencies**

To create this parameter, select the **Rolling Resistance** parameter `ISO 28580`.

**`Input ambient temperature` — Selection**
`off` (default) | `on`

Select to create input port `Tamb`.

**Dependencies**

To create this parameter, select the **Rolling Resistance** parameter `ISO 28580`.

**Magic Formula**

**`Rolling resistance torque coefficient, QSY1` — Torque coefficient**
`0.007` (default) | `scalar`

Rolling resistance torque coefficient, dimensionless.

**Dependencies**

To create this parameter, select the **Rolling Resistance** parameter `Magic Formula`.

**`Longitudinal force rolling resistance coefficient, QSY2` — Force resistance coefficient**
`0` (default) | `scalar`

Longitudinal force rolling resistance coefficient, dimensionless.

**Dependencies**

To create this parameter, select the **Rolling Resistance** parameter `Magic Formula`.

**`Linear rotational speed rolling resistance coefficient, QSY3` — Linear speed coefficient**
`0.0015` (default) | `scalar`

Linear rotational speed rolling resistance coefficient, dimensionless.

**Dependencies**

To create this parameter, select the **Rolling Resistance** parameter `Magic Formula`.

**`Quartic rotational speed rolling resistance coefficient, QSY4` — Quartic speed coefficient**
`8.5e-05` (default) | `scalar`

Quartic rotational speed rolling resistance coefficient, dimensionless.

**Dependencies**

To create this parameter, select the **Rolling Resistance** parameter `Magic Formula`.

**`Camber squared rolling resistance torque, QSY5` — Camber resistance torque**
`0` (default) | `scalar`

Camber squared rolling resistance torque, in 1/rad^2.

**Dependencies**

To create this parameter, select the **Rolling Resistance** parameter `Magic Formula`.

**`Load based camber squared rolling resistance torque, QSY6` — Load resistance torque**
`0` (default) | `scalar`

Load based camber squared rolling resistance torque, in 1/rad^2.

**Dependencies**

To create this parameter, select the **Rolling Resistance** parameter `Magic Formula`.

**`Normal load rolling resistance coefficient, QSY7` — Normal resistance coefficient**
`0.9` (default) | `scalar`

Normal load rolling resistance coefficient, dimensionless.

**Dependencies**

To create this parameter, select the **Rolling Resistance** parameter `Magic Formula`.

**`Pressure load rolling resistance coefficient, QSY8` — Pressure resistance coefficient**
`-0.4` (default) | `scalar`

Pressure load rolling resistance coefficient, dimensionless.

**Dependencies**

To create this parameter, select the **Rolling Resistance** parameter `Magic Formula`.

**`Rolling resistance scaling factor, lam_My` — Scale**
`1` (default) | `scalar`

Rolling resistance scaling factor, dimensionless.

**Dependencies**

To create this parameter, select the **Rolling Resistance** parameter `Magic Formula`.

**Mapped**

**`Spin axis velocity breakpoints, VxMy` — Breakpoints**
`-20:1:20` (default) | `vector`

Spin axis velocity breakpoints, in m/s.

**Dependencies**

To create this parameter, select the **Rolling Resistance** parameter `Mapped torque`.

**`Normal force breakpoints, FzMy` — Breakpoints**
`0:200:1e4` (default) | `vector`

Normal force breakpoints, in N.

**Dependencies**

To create this parameter, select the **Rolling Resistance** parameter `Mapped torque`.

**`Rolling resistance torque map, MyMap` — Lookup table**
`array`

Rolling resistance torque versus axle speed and normal force, in N·m.

**Dependencies**

To create this parameter, select the **Rolling Resistance** parameter `Mapped torque`.

**Brake**

**`Static friction coefficient, mu_static` — Static friction**
`.3` (default) | `scalar`

Static friction coefficient, dimensionless.

**Dependencies**

To enable this parameter, for the **Brake Type** parameter, specify one of these types:

- `Disc`
- `Drum`
- `Mapped`

**`Kinetic friction coefficient, mu_kinetic` — Kinetic friction**
`.2` (default) | `scalar`

Kinematic friction coefficient, dimensionless.

**Dependencies**

To enable this parameter, for the **Brake Type** parameter, specify one of these types:

- `Disc`
- `Drum`
- `Mapped`

**Disc**

**`Disc brake actuator bore, disc_abore` — Bore distance**
`.05` (default) | `scalar`

Disc brake actuator bore, in m.

**Dependencies**

To enable the disc brake parameters, select `Disc` for the **Brake Type** parameter.

**`Brake pad mean radius, Rm` — Radius**
`.177` (default) | `scalar`

Brake pad mean radius, in m.

**Dependencies**

To enable the disc brake parameters, select `Disc` for the **Brake Type** parameter.

**`Number of brake pads, num_pads` — Count**
`2` (default) | `scalar`

Number of brake pads.

**Dependencies**

To enable the disc brake parameters, select `Disc` for the **Brake Type** parameter.

**Drum**

**`Drum brake actuator bore, disc_abore` — Bore distance**
`0.0508` (default) | `scalar`

Drum brake actuator bore, in m.

**Dependencies**

To enable the drum brake parameters, select `Drum` for the **Brake Type** parameter.

**Shoe pin to drum center distance, drum_a — Distance**
`0.123` (default) | `scalar`

Shoe pin to drum center distance, in m.

**Dependencies**

To enable the drum brake parameters, select `Drum` for the **Brake Type** parameter.

**Shoe pin center to force application point distance, drum_c — Distance**
`0.212` (default) | `scalar`

Shoe pin center to force application point distance, in m.

**Dependencies**

To enable the drum brake parameters, select `Drum` for the **Brake Type** parameter.

**Drum internal radius, drum_r — Radius**
`0.15` (default) | `scalar`

Drum internal radius, in m.

**Dependencies**

To enable the drum brake parameters, select `Drum` for the **Brake Type** parameter.

**Shoe pin to pad start angle, drum_theta1 — Angle**
`0` (default) | `scalar`

Shoe pin to pad start angle, in deg.

**Dependencies**

To enable the drum brake parameters, select `Drum` for the **Brake Type** parameter.

**Shoe pin to pad end angle, drum_theta2 — Angle**
`126` (default) | `scalar`

Shoe pin to pad end angle, in deg.

**Dependencies**

To enable the drum brake parameters, select `Drum` for the **Brake Type** parameter.

**Mapped**

**Brake actuator pressure breakpoints, brake_p_bpt — Breakpoints**
`vector`

Brake actuator pressure breakpoints, in bar.

**Dependencies**

To enable the mapped brake parameters, select `Mapped` for the **Brake Type** parameter.

**Wheel speed breakpoints, brake_n_bpt — Breakpoints**
`vector`

Wheel speed breakpoints, in rpm.

**Dependencies**

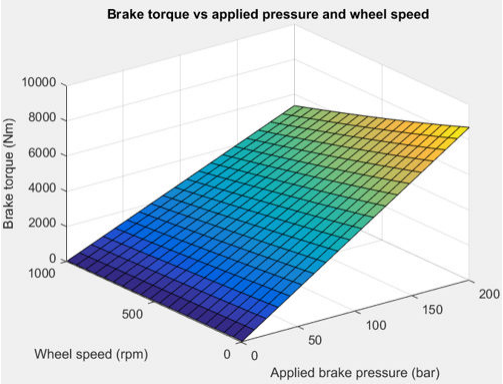
To enable the mapped brake parameters, select `Mapped` for the **Brake Type** parameter.

### `Brake torque map, f_brake_t` — Lookup table

`array`

The lookup table for the brake torque, $f_{brake}(P, N)$, is a function of applied brake pressure and wheel speed, where:

- $T$ is brake torque, in N·m.
- $P$ is applied brake pressure, in bar.
- $N$ is wheel speed, in rpm.




**Dependencies**

To enable the mapped brake parameters, select `Mapped` for the **Brake Type** parameter.

## Vertical

### `Nominal normal force, FNOMIN` — Force

`2000` (default) | `scalar`

Nominal rated wheel load along wheel-fixed $z$-axis, in N.

**Dependencies**

To enable this parameter, set either:

- **Longitudinal Force** to `Magic Formula pure longitudinal slip`.
- **Rolling Resistance** to `Magic Formula`.

### `Nominal rated load scaling factor, lam_Fzo` — Factor

`1` (default) | `scalar`

Nominal rated load scaling factor, dimensionless. Used to scale the normal for specific applications and load conditions.

**Dependencies**

To enable this parameter, set **Longitudinal Force** to `Magic Formula pure longitudinal slip`.

**`Wheel and unsprung mass, m` — Mass**
`10` (default) | `scalar`

Wheel and unsprung mass, in kg. Used in the vertical motion calculations.

**Dependencies**

To enable this parameter, set **Vertical Motion** to `Mapped stiffness and damping`.

**`Initial deflection, zo` — Deflection**
`0` (default) | `scalar`

Initial axle displacement along wheel-fixed *z*-axis, in m.

**Dependencies**

To enable this parameter, set **Vertical Motion** to `Mapped stiffness and damping`.

**`Initial velocity, zdoto` — Velocity**
`0` (default) | `scalar`

Initial axle velocity along wheel-fixed *z*-axis, in m.

**Dependencies**

To enable this parameter, set **Vertical Motion** to `Mapped stiffness and damping`.

**`Gravitational acceleration, g` — Gravity**
`9.81` (default) | `scalar`

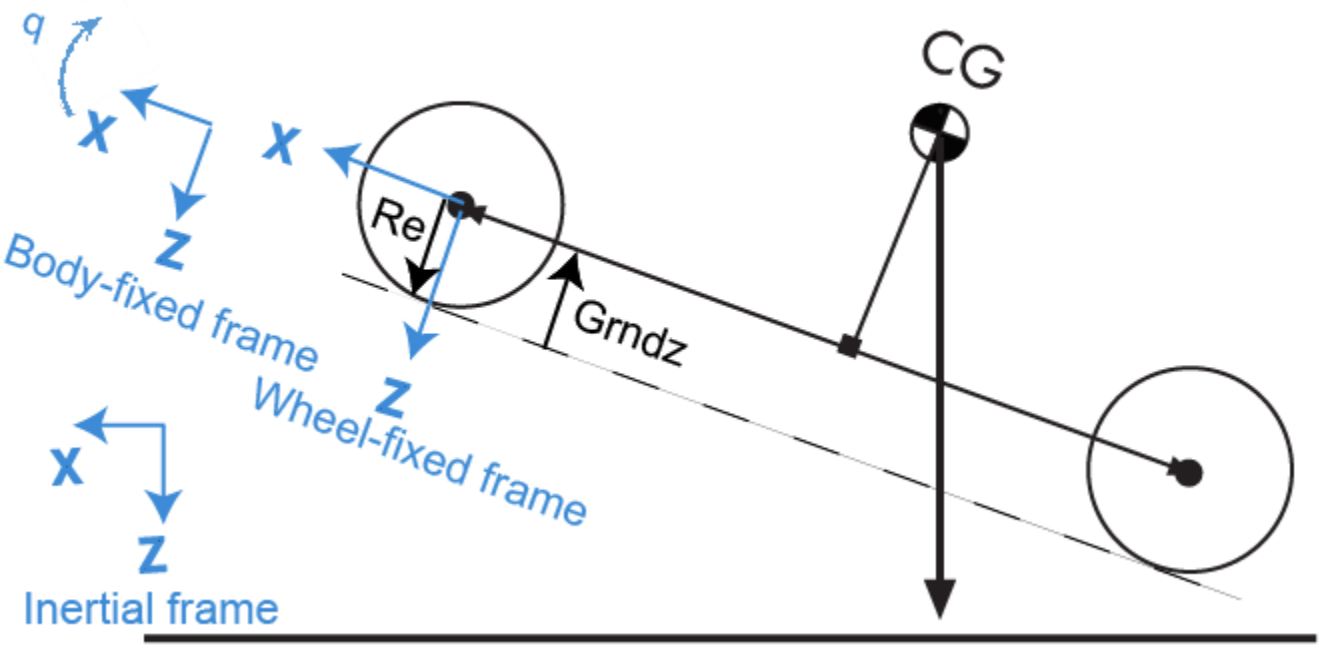
Gravitational acceleration, in m/s^2.

**Dependencies**

To enable this parameter, set **Vertical Motion** to `Mapped stiffness and damping`.

**`Ground displacement, Gndz` — Displacement**
`0` (default) | `scalar`

Ground displacement, `Grndz`, along negative wheel-fixed *z*-axis, in m.




**Dependencies**

To enable this parameter, set **Vertical Motion** to `Mapped stiffness and damping`.

**Mapped Stiffness and Damping**

**`Vertical deflection breakpoints, zFz` — Breakpoints**
`[0 .01 .1]` (default) | `vector`

Vector of sidewall deflection breakpoints corresponding to the force table, in m.

**Dependencies**

To enable this parameter, set **Vertical Motion** to `Mapped stiffness and damping`.

**`Pressure breakpoints, pFz` — Breakpoints**
`[10000 1000000]` (default) | `vector`

Vector of pressure data points corresponding to the force table, in Pa.

**Dependencies**

To enable this parameter, set **Vertical Motion** to `Mapped stiffness and damping`.

**`Force due to deflection, Fzz` — Force**
`[0 1e3 1e4; 0 1e4 1e5]` (default) | `vector`

Force due to sidewall deflection and pressure along wheel-fixed *z*-axis, in N.

**Dependencies**

To enable this parameter, set **Vertical Motion** to `Mapped stiffness and damping`.

**`Vertical velocity breakpoints, zdotFz` — Breakpoints**
`[-20 0 20]` (default) | `scalar`

Vector of sidewall velocity breakpoints corresponding to the force due to velocity table, in m.

**Dependencies**

To enable this parameter, set **Vertical Motion** to `Mapped stiffness and damping`.

**`Force due to velocity, Fzzdot` — Force**
`[500 0 -500;250 0 -250]` (default) | `array`

Force due to sidewall velocity and pressure along wheel-fixed *z*-axis, in N.

**Dependencies**

To enable this parameter, set **Vertical Motion** to `Mapped stiffness and damping`.

**Simulation Setup**

**`Minimum normal force, FZMIN` — Force**
`0` (default) | `scalar`

Minimum normal force, in N. Used with all vertical force calculations.

**`Maximum normal force, FZMAX` — Force**
`10000` (default) | `scalar`

Maximum normal force, in N. Used with all vertical force calculations.

**`Max allowable slip ratio (absolute), kappamax` — Ratio**
`1.5` (default) | `scalar`

Maximum allowable absolute slip ratio, dimensionless.

**`Velocity tolerance used to handle low velocity situations, VXLOW` — Tolerance**
`1` (default) | `scalar`

Velocity tolerance used to handle low-velocity situations, in m/s.

**`Minimum ambient temperature, TMIN` — Tmin**
`0` (default) | `scalar`

Minimum ambient temperature, $T_{MIN}$, in K.

**Dependencies**

To create this parameter, select the **Rolling Resistance** parameter `ISO 28580`.

**`Maximum ambient temperature, TMAX` — Tmax**
`400` (default) | `scalar`

Maximum ambient temperature, $T_{MAX}$, in K.

**Dependencies**

To create this parameter, select the **Rolling Resistance** parameter `ISO 28580`.

## References

[1] Highway Tire Committee. *Stepwise Coastdown Methodology for Measuring Tire Rolling Resistance*. Standard J2452_199906. Warrendale, PA: SAE International, June 1999.

[2] Pacejka, H. B. *Tire and Vehicle Dynamics*. 3rd ed. Oxford, United Kingdom: SAE and Butterworth-Heinemann, 2012.

[3] Schmid, Steven R., Bernard J. Hamrock, and Bo O. Jacobson. "Chapter 18: Brakes and Clutches." *Fundamentals of Machine Elements, SI Version*. 3rd ed. Boca Raton, FL: CRC Press, 2014.

[4] Shigley, Joseph E., and Larry Mitchel. *Mechanical Engineering Design*. 4th ed. New York, NY: McGraw Hill, 1983.

[5] ISO 28580:2018. *Passenger car, truck and bus tyre rolling resistance measurement method -- Single point test and correlation of measurement results*. ISO (International Organization for Standardization), 2018.

## Extended Capabilities

### C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

## See Also

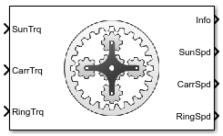
Drive Cycle Source | Longitudinal Driver

### Introduced in R2017a

# Planetary Gear

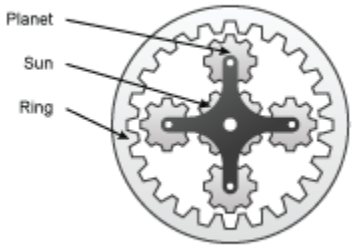
Ideal planetary gear with sun, ring, and carrier

**Library:** Powertrain Blockset / Drivetrain / Couplings




## Description

The Planetary Gear block implements an ideal planetary gear coupling consisting of a rigidly coupled sun, ring, and carrier gears. The block calculates the dynamic response to the sun, carrier, and ring input torques.




In fuel economy and powertrain studies, you can use the Planetary Gear block as a power-split device by coupling it to common driveline elements such as transmissions, engines, clutches, and differentials.

These equations of motion represent the dynamic response of the planetary gear.

$$\dot{\omega}_s J_s = \dot{\omega}_s b_s + T_s + T_{ps}$$
$$\dot{\omega}_c J_c = \dot{\omega}_c b_c + T_c + T_{pc}$$
$$\dot{\omega}_s J_r = \dot{\omega}_r b_r + T_r + T_{pr}$$
$$\dot{\omega}_p J_p = \omega_p b_p + T_{rp} + T_{sp} + T_{cp}$$

To reduce the equations of motion, the block uses these kinematic and geometric constraints.

$$\omega_c r_c = r_s \omega_s + r_p \omega_p$$
$$\omega_r r_r = r_c \omega_c + r_p \omega_p$$
$$r_c = r_s + r_p$$
$$r_r = r_c + r_p$$

### Power Accounting

For the power accounting, the block implements these equations.

| Bus Signal | | | Description | Equations |
|---|---|---|---|---|
| PwrInfo | PwrTrnsfrd — Power transferred between blocks<br>• Positive signals indicate flow into block<br>• Negative signals indicate flow out of block | PwrSun | Sun gear applied power | $\omega_s T_s$ |
| | | PwrCarr | Carrier gear applied power | $\omega_c T_c$ |
| | | PwrRing | Ring gear applied power | $\omega_r T_r$ |
| | PwrNotTrnsfrd — Power crossing the block boundary, but not transferred<br>• Positive signals indicate an input<br>• Negative signals indicate a loss | PwrDampLoss | Mechanical damping loss | $-\left(b_s\omega_s^2 + b_c\omega_c^2 + b_r\omega_r^2 + b_p\omega_p^2\right)$ |
| | PwrStored — Stored energy rate of change<br>• Positive signals indicate an increase<br>• Negative signals indicate a decrease | PwrStoredPlntry | Rate change in rotational kinetic energy | $\dot{\omega}_s\omega_s J_s + \dot{\omega}_c\omega_c J_c + \dot{\omega}_r\omega_r J_r + \dot{\omega}_p\omega_p J_p$ |

The equations use these variables.

| | |
|---|---|
| $\omega_c$, $\omega_p$, $\omega_r$, $\omega_s$ | Carrier, planet, ring, and sun gear angular speed |
| $r_c$, $r_p$, $r_r$, $r_s$ | Carrier, planet, ring, and sun gear angular radius |
| $J_c$, $J_p$, $J_r$, $J_s$ | Carrier, planet, ring, and sun gear inertia |
| $b_c$, $b_p$, $b_r$, $b_s$ | Carrier, planet, ring, and sun gear damping |
| $T_c$, $T_p$, $T_r$, $T_s$ | Applied carrier, planet, ring, and sun gear torque |

| | |
|---|---|
| $T_{ps}$ | Torque applied from planet gear on sun gear |
| $T_{pc}$ | Torque applied from planet gear on carrier gear |
| $T_{pr}$ | Torque applied from planet gear on ring gear |
| $T_{rp}$ | Torque applied from ring gear on planet gear |
| $T_{sp}$ | Torque applied from sun gear on planet gear |
| $T_{cp}$ | Torque applied from carrier gear on planet gear |

## Ports

### Input

#### `SunTrq` — Sun gear applied torque

`scalar`

Sun gear input torque, $T_s$, in N·m.

**Dependencies**

To create this port, for **Port Configuration**, select `Simulink`.

#### `CarrTrq` — Carrier gear applied torque

`scalar`

Carrier gear input torque, $T_c$, in N·m.

**Dependencies**

To create this port, for **Port Configuration**, select `Simulink`.

#### `RingTrq` — Ring gear applied torque

`scalar`

Ring gear applied torque, $T_r$, in N·m.

**Dependencies**

To create this port, for **Port Configuration**, select `Simulink`.

#### `C` — Carrier gear angular speed and torque

two-way connector port

Carrier gear angular speed, $\omega_c$, in rad/s. Carrier gear applied torque, $T_c$, in N·m.

**Dependencies**

To create this port, for **Port Configuration**, select `Two-way connection`.

### Output

#### `Info` — Bus signal

bus

Bus signal containing these block calculations.

| Signal | | | Description | Units |
|---|---|---|---|---|
| Sun | SunTrq | | Sun gear applied torque | N·m |
| | SunSpd | | Sun gear angular speed | rad/s |
| Carr | CarrTrq | | Carrier gear applied torque | N·m |
| | CarrSpd | | Carrier gear angular speed | rad/s |
| Ring | RingTrq | | Ring gear applied torque | N·m |
| PwrInfo | PwrTrnsfrd | PwrSun | Sun gear applied power | W |
| | | PwrCarr | Carrier gear applied power | W |
| | | PwrRing | Ring gear applied power | W |
| | PwrNotTrnsfrd | PwrDampLoss | Mechanical damping loss | W |
| | PwrStored | PwrStoredPlntry | Rate change in rotational kinetic energy | W |

### SunSpd — Sun gear angular speed

scalar

Sun gear angular speed, $\omega_s$, in rad/s.

#### Dependencies

To create this port, for **Port Configuration**, select `Simulink`.

### CarrSpd — Carrier gear angular speed

scalar

Carrier gear angular speed, $\omega_c$, in rad/s.

#### Dependencies

To create this port, for **Port Configuration**, select `Simulink`.

### RingSpd — Ring gear angular speed

scalar

Ring gear angular speed, $\omega_r$, in rad/s.

#### Dependencies

To create this port, for **Port Configuration**, select `Simulink`.

### S — Sun gear angular speed and torque

two-way connector port

Sun gear angular speed, $\omega_s$, in rad/s. Sun gear applied torque, $T_s$, in N·m.

#### Dependencies

To create this port, for **Port Configuration**, select `Two-way connection`.

### R — Ring gear angular speed and torque

two-way connector port

Ring gear angular speed, $\omega_r$, in rad/s. Ring gear applied torque, $T_r$, in N·m.

**Dependencies**

To create this port, for **Port Configuration**, select `Two-way connection`.

## Parameters

**Block Options**

**`Port Configuration` — Specify configuration**
`Simulink` (default) | `Two-way connection`

Specify the port configuration.

**Dependencies**

Specifying `Simulink` creates these ports:

- `SunTrq`
- `CarrTrq`
- `RingTrq`
- `SunSpd`
- `CarrSpd`
- `RingSpd`

Specifying `Two-way connection` creates these ports:

- `C`
- `S`
- `R`

**`Sun to planet ratio, Nsp` — Ratio**
`30/23` (default) | `scalar`

Sun-to-planet gear ratio, dimensionless.

**`Sun to ring ratio, Nsr` — Ratio**
`30/78` (default) | `scalar`

Sun-to-ring gear ratio, dimensionless.

**`Sun inertia, Js` — Inertia**
`.003` (default) | `scalar`

Sun gear inertia, $J_s$, in kg·m^2.

**`Planet inertia, Jp` — Inertia**
`.001` (default) | `scalar`

Planet gear inertia, $J_p$, in kg·m^2.

**`Ring inertia, Jr` — Inertia**
`.01` (default) | `scalar`

Ring gear inertia, $J_r$, in kg·m^2.

**`Carrier inertia, Jc` — Inertia**
`.002` (default) | `scalar`

Carrier gear inertia, $J_c$, in kg·m^2.

**`Sun viscous damping, bs` — Damping**
`.001` (default) | `scalar`

Sun gear viscous damping, $b_s$, N·m· s/rad.

**`Ring viscous damping, br` — Damping**
`.001` (default) | `scalar`

Ring gear viscous damping, $b_r$, N·m· s/rad.

**`Planet viscous damping, bp` — Damping**
`.001` (default) | `scalar`

Planet gear viscous damping, $b_p$, N·m· s/rad.

**`Carrier viscous damping, bc` — Damping**
`.001` (default) | `scalar`

Carrier gear viscous damping, $b_c$, N·m· s/rad.

**`Initial sun velocity, ws_o` — Angular speed**
`0` (default) | `scalar`

Initial sun gear angular speed, in rad/s.

**`Initial carrier velocity, wc_o` — Angular speed**
`0` (default) | `scalar`

Initial carrier gear angular speed, in rad/s.

## Extended Capabilities

### C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

## See Also

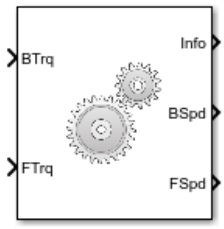
Disc Clutch | Gearbox | Rotational Inertia | Torque Converter | Torsional Compliance

**Introduced in R2017a**

# Gearbox

Ideal rotational gearbox

**Library:** Powertrain Blockset / Drivetrain / Couplings




## Description

The Gearbox block implements an ideal rotational gearbox. The block uses the gear inertias and damping to calculate the velocity response to the base and follower gear pair input torques.

In fuel economy and powertrain efficiency studies, you can use the Gearbox block to model ideal gear coupling and the power transfer between common driveline elements such as transmissions, engines, clutches, and differentials.

The Gearbox block uses these equations to approximate the transmission dynamics.

$$\dot{\omega}_B J_B = \omega_B b_B + \eta N T_F$$
$$\dot{\omega}_F J_F = \omega_F b_F + \eta T_F$$

This constraint equation reduces the system to a one DOF system.

$$\omega_B = N\omega_F$$

To express the ideal torque transfer, the block uses this relationship.

$$\eta N T_B + T_F = 0$$

### Efficiency

To account for the block efficiency, use the **Efficiency factors** parameter. This table summarizes the block implementation for each setting.

| Setting | Implementation |
|---|---|
| `Constant` | Constant efficiency that you can set with the **Constant efficiency factor, eta** parameter. |

| Setting | Implementation |
|---|---|
| `Driveshaft torque, temperature and speed` | Efficiency as a function of base gear input torque, air temperature, and driveshaft speed. Use these parameters to specify the lookup table and breakpoints:<br>• **Efficiency lookup table, eta_tbl**<br>• **Efficiency torque breakpoints, Trq_bpts**<br>• **Efficiency speed breakpoints, omega_bpts**<br>• **Efficiency temperature breakpoints, Temp_bpts**<br>For the air temperature, you can either:<br>• Select **Input temperature** to create an input port.<br>• Set a **Ambient temperature, Tamb** parameter value.<br>To select the interpolation method, use the **Interpolation method** parameter. For more information, see “Interpolation Methods”. |

### Power Accounting

For the power accounting, the block implements these equations.

| Bus Signal | | | Description | Variable | Equations |
|---|---|---|---|---|---|
| `PwrInfo` | `PwrTrnsfrd` — Power transferred between blocks<br>• Positive signals indicate flow into block<br>• Negative signals indicate flow out of block | `PwrBase` | Mechanical power from base shaft | $P_{Base}$ | $P_{Base} = \eta T_B \omega_B$ |
| | | `PwrFlwr` | Mechanical power from follower shaft | $P_{Flwr}$ | $P_{Flwr} = \eta T_F \omega_F$ |
| | `PwrNotTrnsfrd` — Power crossing the block boundary, but not transferred<br>• Positive signals indicate an input<br>• Negative signals indicate a loss | `PwrMechLoss` | Total power loss | $P_{ng}$ | $P_{ng} = -(P_t + P_d) + P$<br>$P_t = \eta T_B \omega_B + \eta T_F \omega_F$ |
| | | `PwrDampLoss` | Power loss due to damping | $P_d$ | $P_d = -\left(b_F \lvert\omega_F\rvert^2 + b_B \lvert\omega_B\rvert^2\right)$ |
| | `PwrStored` — Stored energy rate of change<br>• Positive signals indicate an increase<br>• Negative signals indicate a decrease | `PwrStoredShft` | Rate change of stored internal kinetic energy | $P_s$ | $P_s = (\omega_B \dot{\omega}_B J_B + \omega_F \dot{\omega}_F J_F)$ |

The equations use these variables.

| | |
|---|---|
| $T_B$ | Base gear input torque |
| $T_F$ | Follower gear output torque |
| $\omega_B$ | Base gear angular velocity |
| $\omega_F$ | Follower gear angular velocity |
| $J_B$ | Base gear rotational inertia |
| $J_F$ | Follower gear rotational inertia |
| $b_B$ | Base gear rotational viscous damping |
| $b_F$ | Follower gear rotational viscous damping |
| $N$ | Torque transmission gear ratio |
| $\eta$ | Gear efficiency |
| $P_t$ | Total power |
| $P_d$ | Power loss due to damping |
| $P_s$ | Rate change of stored internal kinetic energy |

## Ports

### Input

#### `BTrq` — Base gear input torque
`scalar`

Base gear input torque, $T_B$, in N·m.

##### Dependencies

To create this port, for **Port Configuration**, select `Simulink`.

#### `FTrq` — Follower gear output torque
`scalar`

Follower gear output torque, $T_F$, in N·m.

##### Dependencies

To create this port, for **Port Configuration**, select `Simulink`.

#### `B` — Base gear angular velocity and torque
two-way connector port

Base gear angular velocity, $\omega_B$, in rad/s. Base gear torque, $T_B$, in N·m.

##### Dependencies

To create this port, for **Port Configuration**, select `Two-way connection`.

#### `AirTemp` — Ambient air temperature
`scalar`

Ambient air temperature, $T_{air}$, in K.

#### Dependencies

To enable this port:

- Set **Efficiency factors** to `Driveshaft torque, speed and temperature`.
- Select **Input ambient temperature**.

### Output

#### `Info` — Bus signal

bus

Bus signal containing these block calculations.

| Signal | | | Description | Variable | Units |
|---|---|---|---|---|---|
| Base | `BaseTrq` | | Base gear input torque | $T_B$ | N·m |
| | `BaseSpd` | | Base gear angular velocity | $\omega_B$ | rad/s |
| Flwr | `FlwrTrq` | | Follower gear torque | $T_F$ | N·m |
| | `FlwrSpd` | | Follower gear angular velocity | $\omega_F$ | rad/s |
| PwrInfo | `PwrTrnsfrd` | `PwrBase` | Mechanical power from base shaft | $P_{Base}$ | W |
| | | `PwrFlwr` | Mechanical power from follower shaft | $P_{Flwr}$ | W |
| | `PwrNotTrnsfrd` | `PwrMechLoss` | Total gear power loss | $P_{ng}$ | W |
| | | `PwrDampLoss` | Power loss due to damping | $P_d$ | W |
| | `PwrStored` | `PwrStoredShft` | Rate change of stored internal kinetic energy | $P_s$ | W |

#### BSpd — Input gear angular velocity

scalar

Base gear angular velocity, $\omega_B$, in rad/s.

#### Dependencies

To create this port, for **Port Configuration**, select `Simulink`.

#### FSpd — Output gear angular velocity

scalar

Follower gear angular velocity, $\omega_F$, in rad/s.

#### Dependencies

To create this port, for **Port Configuration**, select `Simulink`.

#### F — Output gear angular velocity and torque

two-way connector port

Follower gear angular velocity, $\omega_F$, in rad/s. Follower gear torque, $T_F$, in N·m.

##### Dependencies

To create this port, for **Port Configuration**, select `Two-way connection`.

## Parameters

### Block Options

#### `Port Configuration` — Specify configuration

`Simulink` (default) | `Two-way connection`

Specify the port configuration.

##### Dependencies

Specifying `Simulink` creates these ports:

- `BSpd`
- `FSpd`
- `BTrq`
- `FTrq`

Specifying `Two-way connection` creates these ports:

- `B`
- `F`

#### `Efficiency factors` — Specify configuration

`Constant` (default) | `Driveshaft torque, speed and temperature`

To account for the block efficiency, use the **Efficiency factors** parameter. This table summarizes the block implementation for each setting.

| Setting | Implementation |
|---|---|
| `Constant` | Constant efficiency that you can set with the **Constant efficiency factor, eta** parameter. |

| Setting | Implementation |
|---|---|
| `Driveshaft torque, temperature and speed` | Efficiency as a function of base gear input torque, air temperature, and driveshaft speed. Use these parameters to specify the lookup table and breakpoints:<br>• **Efficiency lookup table, eta_tbl**<br>• **Efficiency torque breakpoints, Trq_bpts**<br>• **Efficiency speed breakpoints, omega_bpts**<br>• **Efficiency temperature breakpoints, Temp_bpts**<br>For the air temperature, you can either:<br>• Select **Input temperature** to create an input port.<br>• Set a **Ambient temperature, Tamb** parameter value.<br>To select the interpolation method, use the **Interpolation method** parameter. For more information, see “Interpolation Methods”. |

**`Interpolation method` — Method**
`Flat` | `Nearest` | `Linear point-slope` | `Linear Lagrange` | `Cubic spline`

For more information, see “Interpolation Methods”.

**Dependencies**

To enable this parameter, set **Efficiency factors** to `Driveshaft torque, speed and temperature`.

**`Output shaft rotates in same direction as input` — Rotation**
`off` (default) | `on`

Select to specify that the output shaft rotates in the same direction as the input.

**`Input ambient temperature` — Create input port**
`off` (default) | `on`

Select to create input port `AirTemp` for the ambient air temperature.

**Dependencies**

To enable this parameter, set **Efficiency factors** to `Driveshaft torque, speed and temperature`.

**`Input to output gear ratio, N` — Ratio**
`2` (default) | `scalar`

Base-to-follower gear ratio, dimensionless.

**`Input shaft inertia, J1` — Inertia**
`.01` (default) | `scalar`

Base shaft inertia, in kg·m^2.

**`Output shaft inertia, J2` — Inertia**
`.01` (default) | `scalar`

Follower shaft inertia, in kg·m^2.

**`Input shaft damping, b1` — Damping**
`.001` (default) | `scalar`

Base viscous shaft damping, in N·m· s/rad.

**`Output shaft damping, b2` — Damping**
`.001` (default) | `scalar`

Follower viscous shaft damping, in N·m· s/rad.

**`Input shaft initial velocity, w1_o` — Initial velocity**
`0` (default) | `scalar`

Base shaft initial velocity, in rad/s.

#### Efficiency

**`Constant efficiency factor, eta` — Efficiency**
`1` (default) | `scalar`

Constant efficiency, $\eta$.

#### Dependencies

To enable this parameter, set **Efficiency factors** to `Constant`.

**`Efficiency lookup table, eta_tbl` — Lookup table**
`M`-by-`N`-by-`L` array

Dimensionless array of values for efficiency as a function of:

- `M` input torques
- `N` input speed
- `L` air temperatures

Each value specifies the efficiency for a specific combination of torque, speed, and temperature. The array size must match the dimensions defined by the torque, speed, and temperature breakpoint vectors.

#### Dependencies

To enable this parameter, set **Efficiency factors** to `Driveshaft torque, speed and temperature`.

**`Efficiency torque breakpoints, Trq_bpts` — Torque breakpoints**
`[25, 50, 75, 100, 150, 200, 250]` (default) | 1-by-`M` vector

Vector of input torque, breakpoints for efficiency, in N·m.

#### Dependencies

To enable this parameter, set **Efficiency factors** to `Driveshaft torque, speed and temperature`.

**`Efficiency speed breakpoints, omega_bpts` — Speed breakpoints**
`[52.4 78.5 105 131 157 183 209 262 314 419 524]` (default) | 1-by-N vector

Vector of speed, breakpoints for efficiency, in rad/s.

**Dependencies**

To enable this parameter, set **Efficiency factors** to `Driveshaft torque, speed and temperature`.

**`Efficiency temperature breakpoints, Temp_bpts` — Temperature breakpoints**
`[290 358]` (default) | 1-by-L vector

Vector of ambient temperature breakpoints for efficiency, in K.

**Dependencies**

To enable this parameter, set **Efficiency factors** to `Driveshaft torque, speed and temperature`.

**`Air temperature, Tair` — Ambient air temperature**
`297.15` (default) | `scalar`

Ambient air temperature, $T_{air}$, in K.

**Dependencies**

To enable this parameter:

- Set **Efficiency factors** to `Driveshaft torque, speed and temperature`.
- Clear **Input ambient temperature**.

## Extended Capabilities

**C/C++ Code Generation**
Generate C and C++ code using Simulink® Coder™.

## See Also

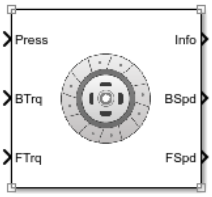
Disc Clutch | Planetary Gear | Rotational Inertia | Torque Converter | Torsional Compliance

**Introduced in R2017a**

# Disc Clutch

Idealized disc clutch coupler

**Library:** Powertrain Blockset / Drivetrain / Couplings




## Description

The Disc Clutch block implements an idealized disc clutch coupler. The block couples the rotary input and output shafts through an idealized friction model. To determine the output torque, the block uses friction parameters, relative slip velocity, and applied input pressure.

In fuel economy and powertrain efficiency studies, you can use the Disc Clutch block to model the mechanical power transfer between common driveline elements such as transmissions, engines, and differentials.

To approximate the torque response, the Disc Clutch block implements friction and dynamic models that depend on the clutch lockup condition. The block determines the locked or unlocked condition based on an idealized dry clutch friction model. This table summarizes the logic the block uses to determine the clutch condition.

| Clutch Condition | When |
|---|---|
| Unlocked | $\omega_i \neq \omega_o$<br>or<br>$T_{fmax} < \left\lvert \frac{J_o T_i - (J_o b_i - J_i b_o)\omega_{i/o}}{J_o + J_i} \right\rvert$ |
| Locked | $\omega_i = \omega_o$<br>and<br>$T_{fmax} < \left\lvert T_i - \frac{J_i(b_i + b_o)\omega_i}{J_o + J_i} + b_o\omega_i \right\rvert$ |

This table summarizes the friction and dynamic models that the block uses for locked or unlocked clutch conditions.

| Clutch Condition | Friction Model | Dynamic Model |
|---|---|---|
| Unlocked | $T_{fmax} = T_k$<br>where,<br>$T_k = N_{disc} P_c A_{eff} R_{eff} \mu_k \tanh[4(\omega_i - \omega_o)]$<br>$R_{eff} = \frac{2({R_o}^3 - {R_i}^3)}{3({R_o}^2 - {R_i}^2)}$<br>and $P_c = \max(P_c - P_{eng}, 0)$ | $\dot{\omega}_i J_i = T_i - T_f - \omega_i b_i$<br>$\dot{\omega}_o J_o = T_f + T_o - \omega_o b_o$ |
| Locked | $T_{fmax} = T_s$<br>where,<br>$T_s = N_{disc} P_c A_{eff} R_{eff} \mu_s$<br>$R_{eff} = \frac{2({R_o}^3 - {R_i}^3)}{3({R_o}^2 - {R_i}^2)}$ | $\dot{\omega}_i (J_o + J_i) = T_o - \omega_i (b_i + b_o) + T_i$<br>$\omega_i = \omega_o$ |

### Power Accounting

For the power accounting, the block implements these equations.

| Bus Signal | | | Description | Equations |
|---|---|---|---|---|
| `PwrInfo` | `PwrTrnsfrd` — Power transferred between blocks<br>• Positive signals indicate flow into block<br>• Negative signals indicate flow out of block | `PwrBase` | Applied base power | $\omega_i T_i$ |
| | | `PwrFlwr` | Applied follower output power | $\omega_o T_o$ |
| | `PwrNotTrnsfrd` — Power crossing the block boundary, but not transferred<br>• Positive signals indicate an input<br>• Negative signals indicate a loss | `PwrDampLoss` | Damping power loss | $-b_o \omega_o^2 \ - b_i \omega_i^2$ |
| | | `PwrCltchSlipLoss` | Clutch slip power loss | $-T_k(\omega_i - \omega_o)$ |
| | `PwrStored` — Stored energy rate of change<br>• Positive signals indicate an increase<br>• Negative signals indicate a decrease | `PwrStoredBase` | Rate change in base rotational kinetic energy | $\dot{\omega}_i \omega_i J_i$ |
| | | `PwrStoredFlwr` | Rate change in follower rotational kinetic energy | $\dot{\omega}_o \omega_o J_o$ |

The equations use these variables.

| | |
|---|---|
| $\omega_i$ | Input shaft angular speed |
| $\omega_o$ | Output shaft angular speed |

| | |
|---|---|
| $b_i$ | Input shaft viscous damping |
| $b_o$ | Output shaft viscous damping |
| $J_i$ | Input shaft moment of inertia |
| $J_o$ | Output shaft moment of inertia |
| $T_f$ | Frictional torque |
| $T_i$ | Net input torque |
| $T_k$ | Kinetic frictional torque |
| $T_o$ | Net output torque |
| $T_s$ | Static frictional torque |
| $T_{fmax}$ | Maximum frictional torque before slipping |
| $P_c$ | Applied clutch pressure |
| $P_{eng}$ | Engagement pressure |
| $A_{eff}$ | Effective area |
| $N_{disc}$ | Number of frictional discs |
| $R_{eff}$ | Effective clutch radius |
| $R_o$ | Annular disk outer radius |
| $R_i$ | Annular disk inner radius |
| $R_e$ | Effective tire radius while under load and for a given pressure |
| $\mu_s$ | Coefficient of static friction |
| $\mu_k$ | Coefficient of kinetic friction |

## Ports

### Input

#### `Press` — Applied clutch pressure

`scalar`

Base gear input torque, $P_c$, in N·m^2.

#### `BTrq` — Applied input torque

`scalar`

Applied input torque, $T_i$, typically from the engine crankshaft or dual mass flywheel damper, in N·m.

##### Dependencies

To create this port, for **Port Configuration**, select `Simulink`.

#### `FTrq` — Applied load torque

`scalar`

Applied load torque, $T_o$, typically from the differential or drive shaft, in N·m.

##### Dependencies

To create this port, for **Port Configuration**, select `Simulink`.

### B — Applied drive shaft angular speed and torque

two-way connector port

Applied drive shaft angular speed, $\omega_i$, in rad/s. Applied drive shaft torque, $T_i$, in N·m.

#### Dependencies

To create this port, for **Port Configuration**, select `Two-way connection`.

### Output

### `Info` — Bus signal

bus

Bus signal containing these block calculations.

| Signal | | | Description | Units |
|---|---|---|---|---|
| `Base` | `BTrq` | | Applied input torque, typically from the engine crankshaft or dual mass flywheel damper | N·m |
| | `BSpd` | | Applied drive shaft angular speed input | rad/s |
| `Flwr` | `FTrq` | | Applied load torque, typically from the differential | N·m |
| | `FSpd` | | Drive shaft angular speed output | rad/s |
| `Cltch` | `CltchForce` | | Applied clutch force | N |
| | `CltchLocked` | | Clutch lock status | NA |
| | `CltchSpdRatio` | | Clutch speed ratio | NA |
| | `CltchEta` | | Clutch power transmission efficiency | NA |
| `PwrInfo` | `PwrTrnsfrd` | `PwrBase` | Applied base power | W |
| | | `PwrFlwr` | Applied follower output power | W |
| | `PwrNotTrnsfrd` | `PwrDampLoss` | Damping power loss | W |
| | | `PwrCltchSlipLoss` | Clutch slip power loss | W |
| | `PwrStored` | `PwrStoredBase` | Rate change in base rotational kinetic energy | W |
| | | `PwrStoredFlwr` | Rate change in follower rotational kinetic energy | W |

### BSpd — Angular speed

`scalar`

Applied drive shaft angular speed input, $\omega_i$, in rad/s.

**Dependencies**

To create this port, for **Port Configuration**, select `Simulink`.

**FSpd — Angular speed**
`scalar`

Drive shaft angular speed output, $\omega_o$, in rad/s.

**Dependencies**

To create this port, for **Port Configuration**, select `Simulink`.

**F — Output velocity and torque**
two-way connector port

Output drive shaft angular speed, $\omega_{oi}$, in rad/s. Output drive shaft torque, $T_o$, in N·m.

**Dependencies**

To create this port, for **Port Configuration**, select `Two-way connection`.

## Parameters

**Block Options**

**`Port Configuration` — Specify configuration**
`Simulink` (default) | `Two-way connection`

Specify the port configuration.

**Dependencies**

Specifying `Simulink` creates these ports:

- `BSpd`
- `FSpd`
- `BTrq`
- `FTrq`

Specifying `Two-way connection` creates these ports:

- `B`
- `F`

**`Clutch force equivalent net radius, Reff` — Radius**
`1` (default) | `scalar`

Clutch force equivalent net radius, in m.

**`Number of disks, Ndisk` — Ratio**
`1` (default) | `scalar`

Number of disks, dimensionless.

**`Effective applied pressure area, Aeff` — Pressure area**
`.01` (default) | `scalar`

Effective applied pressure area, in m^2.

**`Engagement pressure threshold, Peng` — Pressure threshold**
`0` (default) | `scalar`

Pressure to engage clutch, in Pa.

**`Input shaft inertia, Jin` — Inertia**
`.1` (default) | `scalar`

Input shaft inertia, in kg·m^2.

**`Output shaft inertia, Jout` — Inertia**
`.1` (default) | `scalar`

Output shaft inertia, in kg·m^2.

**`Kinetic friction coefficient, muk` — Coefficient**
`.3` (default) | `scalar`

Kinetic friction coefficient, dimensionless.

**`Static friction coefficient, mus` — Coefficient**
`.5` (default) | `scalar`

Static friction coefficient, dimensionless.

**`Input shaft viscous damping, bin` — Damping**
`.001` (default) | `scalar`

Input shaft viscous damping, in N·m· s/rad.

**`Output shaft viscous damping, bout` — Damping**
`.001` (default) | `scalar`

Output shaft viscous damping, in N·m· s/rad.

**`Initial input shaft velocity, win_o` — Initial velocity**
`0` (default) | `scalar`

Input shaft initial velocity, in rad/s.

**`Initial output shaft velocity, wout_o` — Initial velocity**
`0` (default) | `scalar`

Input shaft initial velocity, in rad/s.

**`Clutch actuation time constant, tauC` — Constant**
`.01` (default) | `scalar`

Clutch actuation time constant, in s.

**`Clutch initially locked` — Select to initially lock clutch**
`off` (default) | `on`

Select to lock clutch initially.

## Extended Capabilities

**C/C++ Code Generation**
Generate C and C++ code using Simulink® Coder™.

## See Also
Planetary Gear | Rotational Inertia | Torque Converter | Torsional Compliance

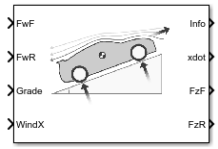
**Introduced in R2017a**

# Vehicle Dynamics Blocks

# Vehicle Body 1DOF Longitudinal

Two-axle vehicle in forward and reverse motion

**Library:** Powertrain Blockset / Vehicle Dynamics
Vehicle Dynamics Blockset / Vehicle Body




## Description

The Vehicle Body 1DOF Longitudinal block implements a one degree-of-freedom (1DOF) rigid vehicle body with constant mass undergoing longitudinal (that is, forward and reverse) motion. Use the block:

- In powertrain and fuel economy studies to represent the vehicle inertial and drag loads when weight transfer from vertical and pitch motions are negligible.
- To determine the engine torque and power required for the vehicle to follow a specified drive cycle.

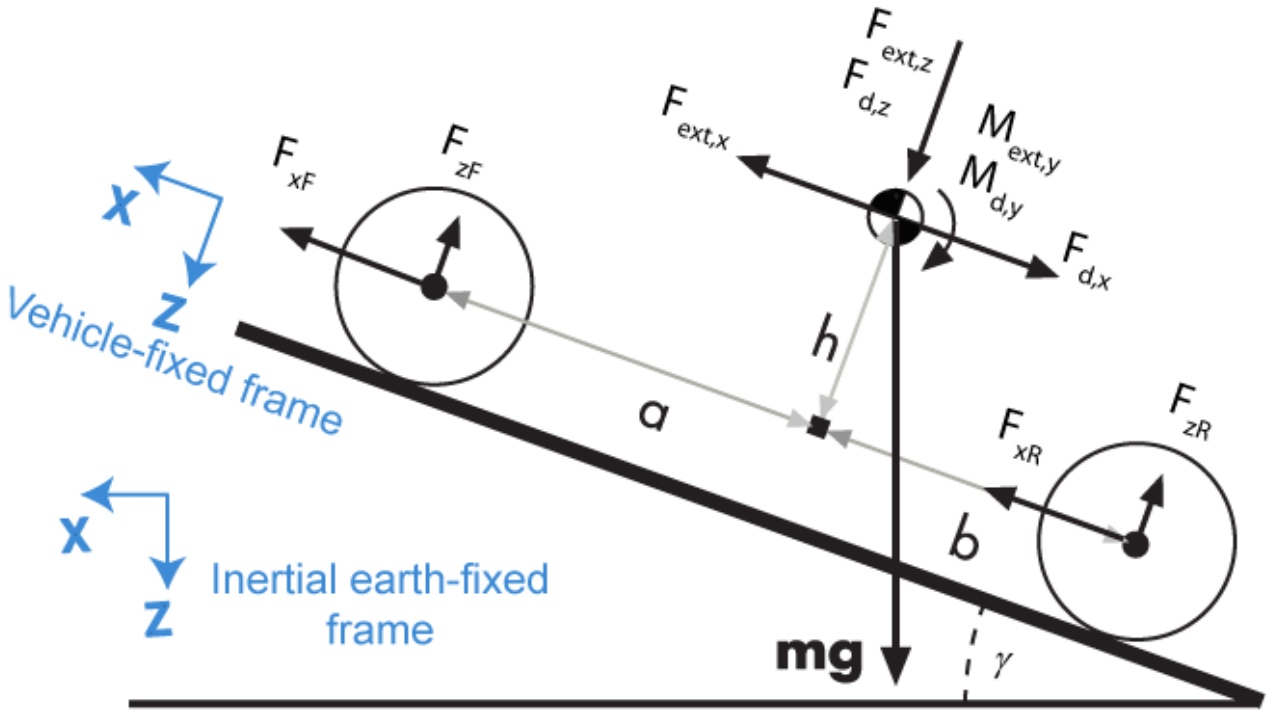
You can select block options to create input ports for external forces, moments, air temperature, and wind speed.

| Block Option Setting | External Input Ports | Description |
|---|---|---|
| **External forces** | `FExt` | External force applied to vehicle CG in the vehicle-fixed frame. |
| **External moments** | `MExt` | External moment about vehicle CG in the vehicle-fixed frame. |
| **Air temperature** | `AirTemp` | Ambient air temperature. Consider this option if you want to vary the temperature during run-time. |
| **Wind X,Y,Z** | `WindXYZ` | Wind speed along earth-fixed *X*-, *Y*-, and *Z*-axes.<br><br>If you do not select this option, the block implements input port `WindX` — Longitudinal wind speed along the earth-fixed *X*-axis. |

### Vehicle Body Model

The vehicle axles are parallel and form a plane. The longitudinal direction lies in this plane and is perpendicular to the axles. If the vehicle is traveling on an inclined slope, the normal direction is not parallel to gravity but is always perpendicular to the axle-longitudinal plane.

The block uses the net effect of all the forces and torques acting on it to determine the vehicle motion. The longitudinal tire forces push the vehicle forward or backward. The weight of the vehicle acts through its center of gravity (CG). The grade angle changes the direction of the resolved gravitational force acting on the vehicle CG. Similarly, the block resolves the resistive aerodynamic drag force on the vehicle CM.




The Vehicle Body 1DOF Longitudinal block implements these equations.

$$F_b = m\ddot{x}$$

$$F_b = F_{xF} + F_{xR} - F_{d,x} + F_{ext,x} - mg\sin\gamma$$

Zero normal acceleration and zero pitch torque determine the normal force on each front and rear axles.

$$F_{zF} = \frac{-M_{ext,y} - M_{d,y} + b(F_{d,z} + F_{ext,z} + mg\cos\gamma) - h(-F_{ext,x} + F_{d,x} + mg\sin\gamma + m\ddot{x})}{N_F(a+b)}$$

$$F_{zR} = \frac{M_{ext,y} + M_{d,y} + a(F_{d,z} + F_{ext,z} + mg\cos\gamma) + h(-F_{ext,x} + F_{d,x} + mg\sin\gamma + m\ddot{x})}{N_R(a+b)}$$

The wheel normal forces satisfy this equation.

$$N_F F_{zF} + N_R F_{zR} - F_{ext,z} = mg\cos\gamma$$

### Wind and Drag Forces

The block subtracts the wind speeds from the vehicle velocity components to obtain a net relative airspeed. To calculate the drag force and moments acting on the vehicle, the block uses the net relative airspeed.

$$F_{d,x} = \frac{1}{2TR} C_d A_f P_{abs} (\dot{x}$$

$$F_{d,z} = \frac{1}{2TR} C_l A_f P_{abs} (\dot{x}$$

$$M_{d,y} = \frac{1}{2TR} C_{pm} A_f P_{abs} (\dot{x} (a + b)$$

By default, to calculate the wind speed along the vehicle-fixed $x$-axis, the block uses the longitudinal wind speed along the earth-fixed $X$-axis. If you select **WindX,Y,Z**, the block uses the wind speed along the earth-fixed $X$-, $Y$-, $Z$-axes.

### Power Accounting

For the power accounting, the block implements these equations.

| Bus Signal | | | Description | Equations |
|---|---|---|---|---|
| `PwrInfo` | `PwrTrnsfrd` — Power transferred between blocks<br>• Positive signals indicate flow into block<br>• Negative signals indicate flow out of block | `PwrFxExt` | Externally applied force power | $P_{FxExt} = F_{xExt} \dot{x}$ |
| | | `PwrFwFx` | Longitudinal force power applied at the front axle | $P_{FwFx} = F_{wF} \dot{x}$ |
| | | `PwrFwRx` | Longitudinal force power applied at the rear axle | $P_{FwRx} = F_{wR} \dot{x}$ |
| | `PwrNotTrnsfrd` — Power crossing the block boundary, but not transferred<br>• Positive signals indicate an input<br>• Negative signals indicate a loss | `PwrFxDrag` | Drag force power | $P_d = -\frac{0.5 C_d A_f P_{abs} \left(\dot{x}^2 - w_x\right)^2}{287.058T} \dot{x}$ |
| | `PwrStored` — Stored energy rate of change<br>• Positive signals indicate an increase<br>• Negative signals indicate a decrease | `wrStoredGrvty` | Rate change in gravitational potential energy | $P_g = -mg\dot{Z}$ |
| | | `PwrStoredxdot` | Rate in change of longitudinal kinetic energy | $P_{\dot{x}} = m\ddot{x}\dot{x}$ |

The equations use these variables.

$F_{xf}$, $F_{xr}$ Longitudinal forces on each wheel at the front and rear ground contact points, respectively

| | |
|---|---|
| $F_{zf}$, $F_{zr}$ | Normal load forces on each wheel at the front and rear ground contact points, respectively |
| $F_{wF}$, $F_{wR}$ | Longitudinal force on front and rear axles along vehicle-fixed $x$-axis |
| $F_{xExt}$, $F_{wR}$ | External force along the vehicle-fixed $x$-axis |
| $F_{d,x}$, $F_{d,z}$ | Longitudinal and normal drag force on vehicle CG |
| $M_{d,y}$ | Torque due to drag on vehicle about the vehicle-fixed $y$-axis |
| $F_d$ | Aerodynamic drag force |
| $V_x$ | Velocity of the vehicle. When $V_x > 0$, the vehicle moves forward. When $V_x < 0$, the vehicle moves backward. |
| $N_f$, $N_r$ | Number of wheels on front and rear axle, respectively |
| $\gamma$ | Angle of road grade |
| $m$ | Vehicle body mass |
| $a$,$b$ | Distance of front and rear axles, respectively, from the normal projection point of vehicle CG onto the common axle plane |
| $h$ | Height of vehicle CG above the axle plane |
| $C_d$ | Frontal air drag coefficient |
| $A_f$ | Frontal area |
| $P_{abs}$ | Absolute pressure |
| $\rho$ | Mass density of air |
| $x$, $\dot{x}$, $\ddot{x}$ | Vehicle longitudinal position, velocity, and acceleration along the vehicle-fixed $x$-axis |
| $w_x$ | Wind speed along the vehicle-fixed $x$-axis |
| $\dot{Z}$ | Vehicle vertical velocity along the vehicle-fixed $z$-axis |

## Limitations

The Vehicle Body 1DOF Longitudinal block lets you model only longitudinal dynamics, parallel to the ground and oriented along the direction of motion. The vehicle is assumed to be in pitch and normal equilibrium. The block does not model pitch or vertical movement. To model a vehicle with three degrees-of-freedom (DOF), use the Vehicle Body 3DOF Longitudinal.

## Ports

### Input

#### FExt — External force on vehicle CG

`array`

External forces applied to vehicle CG, $F_{xext}$, $F_{yext}$, $F_{zext}$, in vehicle-fixed frame, in N. Signal vector dimensions are `[1x3]` or `[3x1]`.

##### Dependencies

To enable this port, select **External forces**.

### MExt — External moment about vehicle CG

array

External moment about vehicle CG, $M_x$, $M_y$, $M_z$, in the vehicle-fixed frame, in N·m. Signal vector dimensions are `[1x3]` or `[3x1]`.

#### Dependencies

To enable this port, select **External moments**.

### FwF — Total longitudinal force on front axle

scalar

Longitudinal force on the front axle, $F_{xf}$, along vehicle-fixed x-axis, in N.

### FwR — Total longitudinal force on rear axle

scalar

Longitudinal force on the rear axle, $Fw_R$, along vehicle-fixed x-axis, in N.

### Grade — Road grade angle

scalar

Road grade angle, $\gamma$, in deg.

### WindX — Longitudinal wind speed

scalar

Longitudinal wind speed, $W_w$, along earth-fixed X-axis, in m/s.

#### Dependencies

To enable this port, clear **Wind X,Y,Z components**.

### WindXYZ — Wind speed

array

Wind speed, $W_w$, $W_{wY}$, $W_{wZ}$ along inertial $X$-, $Y$-, and $Z$-axes, in m/s. Signal vector dimensions are `[1x3]` or `[3x1]`.

#### Dependencies

To enable this port, select **Wind X,Y,Z components**.

### AirTemp — Ambient air temperature

scalar

Ambient air temperature, $T_{air}$, in K. Considering this option if you want to vary the temperature during run-time.

#### Dependencies

To enable this port, select **Air temperature**.

## Output

### Info — Bus signal

bus

Bus signal containing these block values.

| Signal | | | | Description | Value | Units |
|---|---|---|---|---|---|---|
| InertFrm | Cg | Disp | X | Vehicle CG displacement along earth-fixed X-axis | Computed | m |
| | | | Y | Vehicle CG displacement along earth-fixed Y-axis | 0 | m |
| | | | Z | Vehicle CG displacement along earth-fixed Z-axis | Computed | m |
| | | Vel | Xdot | Vehicle CG velocity along earth-fixed X-axis | Computed | m/s |
| | | | Ydot | Vehicle CG velocity along earth-fixed Y-axis | 0 | m/s |
| | | | Zdot | Vehicle CG velocity along earth-fixed Z-axis | Computed | m/s |
| | | Ang | phi | Rotation of vehicle-fixed frame about the earth-fixed X-axis (roll) | 0 | rad |
| | | | theta | Rotation of vehicle-fixed frame about the earth-fixed Y-axis (pitch) | Computed (input - grade angle) | rad |
| | | | psi | Rotation of vehicle-fixed frame about the earth-fixed Z-axis (yaw) | 0 | rad |
| | FrntAxl | Disp | X | Front axle displacement along the earth-fixed X-axis | Computed | m |
| | | | Y | Front axle displacement along the earth-fixed Y-axis | 0 | m |
| | | | Z | Front axle displacement along the earth-fixed Z-axis | Computed | m |
| | | Vel | Xdot | Front axle velocity along the earth-fixed X-axis | Computed | m/s |
| | | | Ydot | Front axle velocity along the earth-fixed Y-axis | 0 | m/s |
| | | | Zdot | Front axle velocity along the earth-fixed Z-axis | Computed | m/s |
| | RearAxl | Disp | X | Rear axle displacement along the earth-fixed X-axis | Computed | m |
| | | | Y | Rear axle displacement along the earth-fixed Y-axis | 0 | m |
| | | | Z | Rear axle displacement along the earth-fixed Z-axis | Computed | m |

| Signal | | | | Description | Value | Units |
|---|---|---|---|---|---|---|
| | | Vel | Xdot | Rear axle velocity along the earth-fixed X-axis | Computed | m/s |
| | | | Ydot | Rear axle velocity along the earth-fixed Y-axis | 0 | m/s |
| | | | Zdot | Rear axle velocity along the earth-fixed Z-axis | Computed | m/s |
| BdyFrm | Cg | Disp | x | Vehicle CG displacement along the vehicle-fixed x-axis | Computed | m |
| | | | y | Vehicle CG displacement along the vehicle-fixed y-axis | 0 | m |
| | | | z | Vehicle CG displacement along the vehicle-fixed z-axis | 0 | m |
| | | Vel | xdot | Vehicle CG velocity along the vehicle-fixed x-axis | Computed | m/s |
| | | | ydot | Vehicle CG velocity along the vehicle-fixed y-axis | 0 | m/s |
| | | | zdot | Vehicle CG velocity along the vehicle-fixed z-axis | 0 | m/s |
| | | AngVel | p | Vehicle angular velocity about the vehicle-fixed x-axis (roll rate) | 0 | rad/s |
| | | | q | Vehicle angular velocity about the vehicle-fixed y-axis (pitch rate) | 0 | rad/s |
| | | | r | Vehicle angular velocity about the vehicle-fixed z-axis (yaw rate) | 0 | rad/s |
| | | Accel | ax | Vehicle CG acceleration along the vehicle-fixed x-axis | Computed | gn |
| | | | ay | Vehicle CG acceleration along the vehicle-fixed y-axis | 0 | gn |
| | | | az | Vehicle CG acceleration along the vehicle-fixed z-axis | 0 | gn |
| | Forces | Body | Fx | Net force on vehicle CG along the vehicle-fixed x-axis | 0 | N |

| Signal | | | | | Description | Value | Units |
|---|---|---|---|---|---|---|---|
| | | | Fy | | Net force on vehicle CG along the vehicle-fixed y-axis | 0 | N |
| | | | Fz | | Net force on vehicle CG along the vehicle-fixed z-axis | 0 | N |
| | | Ext | Fx | | External force on vehicle CG along the vehicle-fixed x-axis | Computed | N |
| | | | Fy | | External force on vehicle CG along the vehicle-fixed y-axis | Computed | N |
| | | | Fz | | External force on vehicle CG along the vehicle-fixed z-axis | Computed | N |
| | | FrntAxl | Fx | | Longitudinal force on front axle, along the vehicle-fixed x-axis | 0 | N |
| | | | Fy | | Lateral force on front axle, along the vehicle-fixed y-axis | 0 | N |
| | | | Fz | | Normal force on front axle, along the vehicle-fixed z-axis | Computed | N |
| | | RearAxl | Fx | | Longitudinal force on rear axle, along the vehicle-fixed x-axis | 0 | N |
| | | | Fy | | Lateral force on rear axle, along the vehicle-fixed y-axis | 0 | N |
| | | | Fz | | Normal force on rear axle, along the vehicle-fixed z-axis | Computed | N |
| | | Tires | FrntTire | Fx | Front tire force, along the vehicle-fixed x-axis | 0 | N |
| | | | | Fy | Front tire force, along the vehicle-fixed y-axis | 0 | N |
| | | | | Fz | Front tire force, along the vehicle-fixed z-axis | Computed | N |
| | | | RearTire | Fx | Rear tire force, along the vehicle-fixed x-axis | 0 | N |
| | | | | Fy | Rear tire force, along the vehicle-fixed y-axis | 0 | N |

| Signal | | | | | Description | Value | Units |
|---|---|---|---|---|---|---|---|
| | | | | Fz | Rear tire force, along the vehicle-fixed z-axis | Computed | N |
| | | Drag | Fx | | Drag force on vehicle CG along the vehicle-fixed x-axis | Computed | N |
| | | | Fy | | Drag force on vehicle CG along the vehicle-fixed y-axis | Computed | N |
| | | | Fz | | Drag force on vehicle CG along the vehicle-fixed z-axis | Computed | N |
| | | Grvty | Fx | | Gravity force on vehicle CG along the vehicle-fixed x-axis | Computed | N |
| | | | Fy | | Gravity force on vehicle CG along the vehicle-fixed y-axis | 0 | N |
| | | | Fz | | Gravity force on vehicle CG along the vehicle-fixed z-axis | Computed | N |
| | Moments | Body | Mx | | Net moment on vehicle CG about the vehicle-fixed x-axis | 0 | N·m |
| | | | My | | Net moment on vehicle CG about the vehicle-fixed y-axis | 0 | N·m |
| | | | Mz | | Net moment on vehicle CG about the vehicle-fixed z-axis | 0 | N·m |
| | | Drag | Mx | | Drag moment on vehicle CG about the vehicle-fixed x-axis | Computed | N·m |
| | | | My | | Drag moment on vehicle CG about the vehicle-fixed y-axis | Computed | N·m |
| | | | Mz | | Drag moment on vehicle CG about the vehicle-fixed z-axis | Computed | N·m |
| | | Ext | Fx | | External moment on vehicle CG about the vehicle-fixed x-axis | Computed | N·m |
| | | | Fy | | External moment on vehicle CG about the vehicle-fixed y-axis | Computed | N·m |

| Signal | | | | Description | Value | Units |
|---|---|---|---|---|---|---|
| | | | Fz | External moment on vehicle CG about the vehicle-fixed z-axis | Computed | N·m |
| | FrntAxl | Disp | x | Front axle displacement along the vehicle-fixed x-axis | Computed | m |
| | | | y | Front axle displacement along the vehicle-fixed y-axis | 0 | m |
| | | | z | Front axle displacement along the vehicle-fixed z-axis | Computed | m |
| | | Vel | xdot | Front axle velocity along the vehicle-fixed x-axis | Computed | m/s |
| | | | ydot | Front axle velocity along the vehicle-fixed y-axis | 0 | m/s |
| | | | zdot | Front axle velocity along the vehicle-fixed z-axis | Computed | m/s |
| | | Steer | WhlAngFL | Front left wheel steering angle | Computed | rad |
| | | | WhlAngFR | Front right wheel steering angle | Computed | rad |
| | RearAxl | Disp | x | Rear axle displacement along the vehicle-fixed x-axis | Computed | m |
| | | | y | Rear axle displacement along the vehicle-fixed y-axis | 0 | m |
| | | | z | Rear axle displacement along the vehicle-fixed z-axis | Computed | m |
| | | Vel | xdot | Rear axle velocity along the vehicle-fixed x-axis | Computed | m/s |
| | | | ydot | Rear axle velocity along the vehicle-fixed y-axis | 0 | m/s |
| | | | zdot | Rear axle velocity along the vehicle-fixed z-axis | Computed | m/s |
| | | Steer | WhlAngRL | Rear left wheel steering angle | Computed | rad |
| | | | WhlAngRR | Rear right wheel steering angle | Computed | rad |
| | Pwr | PwrExt | | Applied external power | Computed | W |
| | | Drag | | Power loss due to drag | Computed | W |

| Signal | | | Description | Value | Units |
|---|---|---|---|---|---|
| PwrInfo | PwrTrnsfrd | PwrFxExt | Externally applied force power | Computed | W |
| | | PwrFwFx | Longitudinal force power applied at the front axle | Computed | W |
| | | PwrFwRx | Longitudinal force power applied at the rear axle | Computed | W |
| | PwrNotTrnsfrd | PwrFxDrag | Drag force power | Computed | W |
| | PwrStored | wrStoredGrvty | Rate change in gravitational potential energy | Computed | W |
| | | PwrStoredxdot | Rate in change of longitudinal kinetic energy | Computed | W |

**xdot — Vehicle body longitudinal velocity**
scalar

Vehicle body longitudinal velocity along the earth-fixed reference frame X-axis, in m/s.

**FzF — Front axle normal force**
scalar

Normal load force on the front axle, $F_{zf}$, along vehicle-fixed z-axis, in N.

**FzR — Rear axle normal force**
scalar

Normal force on rear axle, $F_{zr}$, along the vehicle-fixed z-axis, in N.

## Parameters

**Options**

**External forces — FExt input port**
off (default) | on

Specify to create input port FExt.

**External moments — MExt input port**
off (default) | on

Specify to create input port MExt.

**Air temperature — AirTemp input port**
off (default) | on

Specify to create input port AirTemp.

**Wind X,Y,Z components — WindXYZ input port**
off (default) | on

Specify to create input port `WindXYZ`.

#### Longitudinal

**`Number of wheels on front axle, NF` — Front wheel count**
`2` (default) | `scalar`

Number of wheels on front axle, $N_F$. The value is dimensionless.

**`Number of wheels on rear axle, NR` — Rear wheel count**
`2` (default) | `scalar`

Number of wheels on rear axle, $N_R$. The value is dimensionless.

**`Mass, m` — Vehicle mass**
`1500` (default) | `scalar`

Vehicle mass, $M$, in kg.

**`Horizontal distance from CG to front axle, a` — Front axle distance**
`1.4` (default) | `scalar`

Horizontal distance $a$ from the vehicle CG to the front wheel axle, in m.

**`Horizontal distance from CG to rear axle, b` — Rear axle distance**
`1.8` (default) | `scalar`

Horizontal distance $b$ from the vehicle CG to the rear wheel axle, in m.

**`CG height above axles, h` — Height**
`.35` (default) | `scalar`

Height of vehicle CG above the ground, $h$, in m.

**`Drag coefficient, Cd` — Drag**
`.3` (default) | `scalar`

Air drag coefficient, $C_d$. The value is dimensionless.

**`Lift coefficient, Cl` — Lift**
`0` (default) | `scalar`

Air lift coefficient, $C_l$. The value is dimensionless.

**`Pitch drag moment coefficient, Cpm` — Pitch drag**
`0` (default) | `scalar`

Pitch drag moment coefficient, $C_{pm}$. The value is dimensionless.

**`Frontal area, Af` — Area**
`4` (default) | `scalar`

Effective vehicle cross-sectional area, $A$, to calculate the aerodynamic drag force on the vehicle, in $m^2$.

**`Initial position, x_o` — Position**
`0` (default) | `scalar`

Vehicle body longitudinal initial position along the vehicle-fixed x-axis, $x_o$, in m.

**`Initial velocity, xdot_o` — Velocity**
`0` (default) | `scalar`

Vehicle body longitudinal initial velocity along the vehicle-fixed x-axis, $\dot{x}_0$, in m/s.

#### Environment

**`Absolute air pressure, Pabs` — Pressure**
`101325` (default) | `scalar`

Environmental air absolute pressure, $P_{abs}$, in Pa.

**`Air temperature, T` — Ambient air temperature**
`273` (default) | `scalar`

Ambient air temperature, $T_{air}$, in K.

#### Dependencies

To enable this parameter, clear **Air temperature**.

**`Gravitational acceleration, g` — Gravity**
`9.81` (default) | `scalar`

Gravitational acceleration, $g$, in m/s$^2$.

## Extended Capabilities

#### C/C++ Code Generation
Generate C and C++ code using Simulink® Coder™.

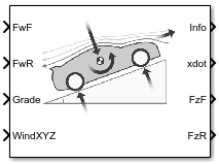
## See Also
Vehicle Body 3DOF Longitudinal | Vehicle Body Total Road Load

#### Introduced in R2017a

# Vehicle Body 3DOF Longitudinal

3DOF rigid vehicle body to calculate longitudinal, vertical, and pitch motion

**Library:** Powertrain Blockset / Vehicle Dynamics
Vehicle Dynamics Blockset / Vehicle Body




## Description

The Vehicle Body 3DOF Longitudinal block implements a three degrees-of-freedom (3DOF) rigid vehicle body model with configurable axle stiffness to calculate longitudinal, vertical, and pitch motion. The block accounts for body mass, aerodynamic drag, road incline, and weight distribution between the axles due to acceleration and the road profile.

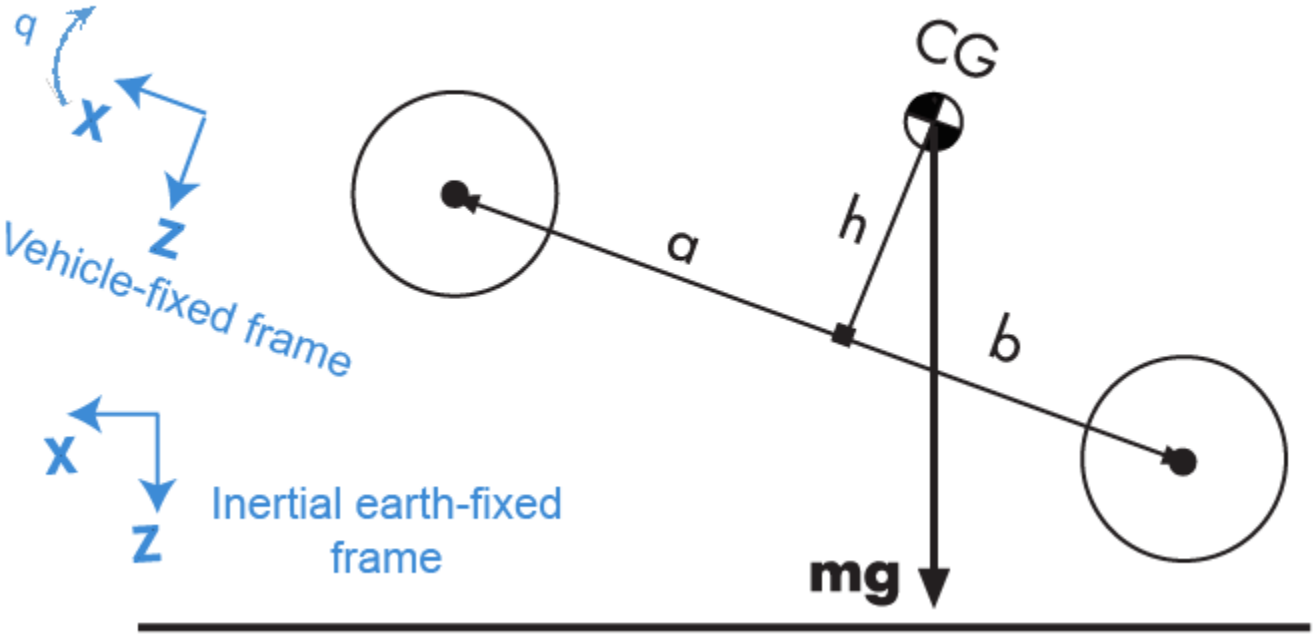
You can specify the type of axle attachment to the vehicle:

- Grade angle — Vertical axle displacement from road surface to axles remains constant. The block uses tabular stiffness and damping parameters to model the suspension forces acting between the vehicle body and axles.
- Axle displacement — Axles have input-provided vertical displacement and velocity with respect to the road grade. The block uses tabular stiffness and damping parameters to model the suspension forces acting between the vehicle body and axle.
- External suspension — Axles have externally applied forces for coupling the vehicle body to custom suspension models.

If the weight transfer from vertical and pitch motions are not negligible, consider using this block to represent vehicle motion in powertrain and fuel economy studies. For example, in studies with heavy breaking or acceleration or road profiles that contain larger vertical changes.

The block uses rigid-body vehicle motion, suspension system forces, and wind and drag forces to calculate the normal forces on the front and rear axles. The block resolves the force components and moments on the rigid vehicle body frame:

$$F_x = F_{wF} + F_{wR} - F_{d,x} - F_{sx,F} - F_{sx,R} + F_{g,x}$$
$$F_z = F_{d,z} - F_{sz,F} - F_{sz,R} + F_{g,z}$$
$$M_y = aF_{sz,F} - bF_{sz,R} + h\left(F_{wF} + F_{wR} + F_{sx,F} + F_{sx,R}\right) - M_{d,y}$$




### Rigid-Body Vehicle Motion

The vehicle axles are parallel and form a plane. The longitudinal direction lies in this plane and is perpendicular to the axles. If the vehicle is traveling on an inclined slope, the normal direction is not parallel to gravity but is always perpendicular to the axle-longitudinal plane.

The block uses the net effect of all the forces and torques acting on it to determine the vehicle motion. The longitudinal tire forces push the vehicle forward or backward. The weight of the vehicle acts through its center of gravity (CG). Depending on the inclined angle, the weight pulls the vehicle to the ground and either forward or backward. Whether the vehicle travels forward or backward, aerodynamic drag slows it down. For simplicity, the drag is assumed to act through the CG.

The Vehicle Body 3DOF Longitudinal implements these equations.

$$\ddot{x} = \frac{F_x}{m} - qz$$

$$\ddot{z} = \frac{F_z}{m} - qx$$

$$\dot{q} = \frac{M_y}{I_{yy}}$$

$$\dot{\theta} = q$$

### Suspension System Forces

If you configure the block with the **Ground interaction type** parameter `Grade angle` or `Axle displacement, velocity`, the block uses nonlinear stiffness and damping parameters to model the suspension system.

The front and rear axle suspension forces are given by:

$$Fs_F = N_F[Fk_F + Fb_F]$$
$$Fs_R = N_R[Fk_R + Fb_R]$$

The block uses lookup tables to implement the front and rear suspension stiffness. To account for kinematic and material nonlinearities, including collisions with end-stops, the tables are functions of the stroke.

$$Fk_F = f(dZ_F)$$
$$Fk_R = f(dZ_R)$$

The block uses lookup tables to implement the front and rear suspension damping. To account for nonlinearities, compression, and rebound, the tables are functions of the stroke rate.

$$Fb_F = f(d\dot{Z}_F)$$
$$Fb_R = f(d\dot{Z}_R)$$

The stroke is the difference in the vehicle vertical and axle positions. The stroke rate is the difference in the vertical and axle velocities.

$$dZ_F = Z_F - \bar{Z}_F$$
$$dZ_R = Z_R - \bar{Z}_R$$
$$d\dot{Z}_F = \dot{Z}_F - \dot{\bar{Z}}_F$$
$$d\dot{Z}_R = \dot{Z}_R - \dot{\bar{Z}}_R$$

When the **Ground interaction type** parameter is `Grade angle`, the axle vertical positions ($\bar{Z}_F, \bar{Z}_R$) and velocities ($\dot{\bar{Z}}_F, \dot{\bar{Z}}_R$) are set to `0`.

### Wind and Drag Forces

The block subtracts the wind speeds from the vehicle velocity components to obtain a net relative airspeed. To calculate the drag force and moments acting on the vehicle, the block uses the net relative airspeed:

$$F_{d,x} = \frac{1}{2TR} C_d A_f P_{abs}(\dot{x}$$
$$F_{d,z} = \frac{1}{2TR} C_l A_f P_{abs}(\dot{x}$$
$$M_{d,y} = \frac{1}{2TR} C_{pm} A_f P_{abs}(\dot{x}(a + b)$$

### Power Accounting

For the power accounting, the block implements these equations.

| Bus Signal | | | Description | Equations |
|---|---|---|---|---|
| PwrInfo | PwrTrnsfrd — Power transferred between blocks<br>• Positive signals indicate flow into block<br>• Negative signals indicate flow out of block | PwrFxExt | Externally applied longitudinal force power | $P_{FxExt} = F_{xExt}\dot{x}$ |
| | | PwrFzExt | Externally applied longitudinal force power | $P_{FzExt} = F_{zExt}\dot{z}$ |
| | | PwrMyExt | Externally applied pitch moment power | $P_{MzExt} = M_{zExt}\dot{\theta}$ |
| | | PwrFwFx | Longitudinal force applied at the front axle | $P_{FwFx} = F_{wF}\dot{x}$ |
| | | PwrFwRx | Longitudinal force applied at the rear axle | $P_{FwRx} = F_{wR}\dot{x}$ |
| | PwrNotTrnsfrd — Power crossing the block boundary, but not transferred<br>• Positive signals indicate an input<br>• Negative signals indicate a loss | PwrFsF | Internal power transferred between suspension and vehicle body at the front axle | $P_{Fs,F} = -P_{FwFx} + P_{FsbF} + P_{Fsk,F} + F_{xF}\dot{x}_F + F_{zF}\dot{z}_F$ |
| | | PwrFsR | Internal power transferred between suspension and vehicle body at the rear axle | $P_{Fs,R} = -P_{FwRx} + P_{Fsb,R} + P_{Fsk,R} + F_{xF}\dot{x}_F + F_{zF}\dot{z}_F$ |
| | | PwrFxDrag | Longitudinal drag force power | $P_{d,x} = F_{d,x}\dot{x}$ |
| | | PwrFzDrag | Vertical drag force power | $P_{d,z} = F_{d,z}\dot{z}$ |
| | | PwrMyDrag | Drag pitch moment power | $P_{d,My} = M_{d,y}\dot{\theta}$ |
| | | PwrFsb | Total suspension damping power | $P_{Fsb} = \sum_{i=F,R} F_{sb,i}\dot{z}_i$ |
| | PwrStored — Stored energy rate of change<br>• Positive signals indicate an increase<br>• Negative signals indicate a decrease | PwrStoredGrvty | Rate change in gravitational potential energy | $P_g = -mg\dot{Z}$ |
| | | PwrStoredxdot | Rate of change of longitudinal kinetic energy | $P_{\dot{x}} = m\ddot{x}\dot{x}$ |
| | | PwrStoredzdot | Rate of change of longitudinal kinetic energy | $P_{\dot{z}} = m\ddot{z}\dot{z}$ |

| Bus Signal | | | Description | Equations |
|---|---|---|---|---|
| | | PwrStoredq | Rate of change of rotational pitch kinetic energy | $P_{\dot{\theta}} = I_{yy}\ddot{\theta}\dot{\theta}$ |
| | | PwrStoredFsFzSprng | Stored spring energy from front suspension | $P_{FskF} = F_{sk,F}\dot{z}_F$ |
| | | PwrStoredFsRzSprng | Stored spring energy from rear suspension | $P_{FskF} = F_{sk,R}\dot{z}_R$ |

The equations use these variables.

| | |
|---|---|
| $F_x$ | Longitudinal force on vehicle |
| $F_z$ | Normal force on vehicle |
| $M_y$ | Torque on vehicle about the vehicle-fixed *y*-axis |
| $F_{wF}$, $F_{wR}$ | Longitudinal force on front and rear axles along vehicle-fixed *x*-axis |
| $F_{d,x}$, $F_{d,z}$ | Longitudinal and normal drag force on vehicle CG |
| $F_{sx,F}$, $F_{sx,R}$ | Longitudinal suspension force on front and rear axles |
| $F_{sz,F}$, $F_{sz,R}$ | Normal suspension force on front and rear axles |
| $F_{g,x}$,$F_{g,z}$ | Longitudinal and normal gravitational force on vehicle along the vehicle-fixed frame |
| $M_{d,y}$ | Torque due to drag on vehicle about the vehicle-fixed *y*-axis |
| $a$,$b$ | Distance of front and rear axles, respectively, from the normal projection point of vehicle CG onto the common axle plane |
| $h$ | Height of vehicle CG above the axle plane along vehicle-fixed *z*-axis |
| $Fs_F$, $Fs_R$ | Front and rear axle suspension force along vehicle-fixed *z*-axis |
| $Z_{wF}$, $Z_{wR}$ | Front and rear vehicle normal position along earth-fixed *z*-axis |
| $\Theta$ | Vehicle pitch angle about the vehicle-fixed *y*-axis |
| $m$ | Vehicle body mass |
| $N_F$, $N_R$ | Number of front and rear wheels |
| $I_{yy}$ | Vehicle body moment of inertia about the vehicle-fixed *y*-axis |
| $x$, $\dot{x}$, $\ddot{x}$ | Vehicle longitudinal position, velocity, and acceleration along the vehicle-fixed *x*-axis |
| $z$, $\dot{z}$, $\ddot{z}$ | Vehicle normal position, velocity, and acceleration along the vehicle-fixed *z*-axis |
| $Fk_F$, $Fk_R$ | Front and rear wheel suspension stiffness force along vehicle-fixed *z*-axis |
| $Fb_F$, $Fb_R$ | Front and rear wheel suspension damping force along vehicle-fixed *z*-axis |
| $Z_F$, $Z_R$ | Front and rear vehicle vertical position along earth-fixed *Z*-axis |
| $\dot{Z}_F$, $\dot{Z}_R$ | Front and rear vehicle vertical velocity along vehicle-fixed *z*-axis |
| $\bar{Z}_F$, $\bar{Z}_R$ | Front and rear wheel axle vertical position along vehicle-fixed *z*-axis |
| $\dot{\bar{Z}}_F$, $\dot{\bar{Z}}_R$ | Front and rear wheel axle vertical velocity along earth-fixed *z*-axis |

| | |
|---|---|
| $dZ_F$, $dZ_R$ | Front and rear axle suspension deflection along vehicle-fixed $z$-axis |
| $d\dot{Z}_F, d\dot{Z}_R$ | Front and rear axle suspension deflection rate along vehicle-fixed $z$-axis |
| $C_d$ | Frontal air drag coefficient acting along the vehicle-fixed $x$-axis |
| $C_l$ | Lateral air drag coefficient acting along the vehicle-fixed $z$-axis |
| $C_{pm}$ | Air drag pitch moment acting about the vehicle-fixed $y$-axis |
| $A_f$ | Frontal area |
| $P_{abs}$ | Environmental absolute pressure |
| $R$ | Atmospheric specific gas constant |
| $T$ | Environmental air temperature |
| $w_x$ | Wind speed along the vehicle-fixed $x$-axis |

## Ports

### Input

#### FExt — External force on vehicle CG

`array`

External forces applied to vehicle CG, $F_{xext}$, $F_{yext}$, $F_{zext}$, in vehicle-fixed frame, in N. Signal vector dimensions are `[1x3]` or `[3x1]`.

##### Dependencies

To enable this port, select **External forces**.

#### MExt — External moment about vehicle CG

`array`

External moment about vehicle CG, $M_x$, $M_y$, $M_z$, in the vehicle-fixed frame, in N·m. Signal vector dimensions are `[1x3]` or `[3x1]`.

##### Dependencies

To enable this port, select **External moments**.

#### FwF — Total longitudinal force on the front axle

`scalar`

Longitudinal force on the front axle, $Fw_F$, along vehicle-fixed $x$-axis, in N.

#### FwR — Total longitudinal force on the rear axle

`scalar`

Longitudinal force on the rear axle, $Fw_R$, along vehicle-fixed $x$-axis, in N.

#### Grade — Road grade angle

`scalar`

Road grade angle, $\gamma$, in deg.

#### FsF — Suspension force on front axle per wheel

`vector`

Suspension force on front axle, $Fs_F$, along the vehicle-fixed $z$-axis, in N.

**Dependencies**

To enable this port, for the **Ground interaction type** parameter, select `External suspension`.

### FsR — Suspension force on rear axle per wheel

`vector`

Suspension force on rear axle, $Fs_R$, along the vehicle-fixed $z$-axis, in N.

**Dependencies**

To enable this port, for the **Ground interaction type** parameter, select `External suspension`.

### WindXYZ — Wind speed

`array`

Wind speed, $W_X$, $W_Y$, $W_Z$ along earth-fixed $X$-, $Y$-, and $Z$-axes, in m/s. Signal vector dimensions are `[1x3]` or `[3x1]`.

### AirTemp — Ambient air temperature

`scalar`

Ambient air temperature, $T_{air}$, in K. Considering this option if you want to vary the temperature during run-time.

**Dependencies**

To enable this port, select **Air temperature**.

### zF,R — Forward and rear axle positions

`vector`

Forward and rear axle positions along the vehicle-fixed $z$-axis, $\bar{Z}_F, \bar{Z}_R$, in m.

**Dependencies**

To enable this port, for the **Ground interaction type** parameter, select `Axle displacement, velocity`.

### zdotF,R — Forward and rear axle velocities

`vector`

Forward and rear axle velocities along the vehicle-fixed $z$-axis, $\dot{\bar{Z}}_F, \dot{\bar{Z}}_R$, in m/s.

**Dependencies**

To enable this port, for the **Ground interaction type** parameter, select `Axle displacement, velocity`.

## Output

### Info — Bus signal

bus

Bus signal containing these block values.

| Signal | | | | Description | Value | Units |
|---|---|---|---|---|---|---|
| InertFrm | Cg | Disp | X | Vehicle CG displacement along earth-fixed *X*-axis | Computed | m |
| | | | Y | Vehicle CG displacement along earth-fixed *Y*-axis | 0 | m |
| | | | Z | Vehicle CG displacement along earth-fixed *Z*-axis | Computed | m |
| | | Vel | Xdot | Vehicle CG velocity along earth-fixed *X*-axis | Computed | m/s |
| | | | Ydot | Vehicle CG velocity along earth-fixed *Y*-axis | 0 | m/s |
| | | | Zdot | Vehicle CG velocity along earth-fixed *Z*-axis | Computed | m/s |
| | | Ang | phi | Rotation of vehicle-fixed frame about the earth-fixed *X*-axis (roll) | 0 | rad |
| | | | theta | Rotation of vehicle-fixed frame about the earth-fixed *Y*-axis (pitch) | Computed | rad |
| | | | psi | Rotation of vehicle-fixed frame about the earth-fixed *Z*-axis (yaw) | 0 | rad |
| | FrntAxl | Disp | X | Front axle displacement along the earth-fixed *X*-axis | Computed | m |
| | | | Y | Front axle displacement along the earth-fixed *Y*-axis | 0 | m |
| | | | Z | Front axle displacement along the earth-fixed *Z*-axis | Computed | m |
| | | Vel | Xdot | Front axle velocity along the earth-fixed *X*-axis | Computed | m/s |
| | | | Ydot | Front axle velocity along the earth-fixed *Y*-axis | 0 | m/s |
| | | | Zdot | Front axle velocity along the earth-fixed *Z*-axis | Computed | m/s |
| | RearAxl | Disp | X | Rear axle displacement along the earth-fixed *X*-axis | Computed | m |
| | | | Y | Rear axle displacement along the earth-fixed *Y*-axis | 0 | m |
| | | | Z | Rear axle displacement along the earth-fixed *Z*-axis | Computed | m |
| | | Vel | Xdot | Rear axle velocity along the earth-fixed *X*-axis | Computed | m/s |
| | | | Ydot | Rear axle velocity along the earth-fixed *Y*-axis | 0 | m/s |

| Signal | | | | Description | Value | Units |
|---|---|---|---|---|---|---|
| | | | Zdot | Rear axle velocity along the earth-fixed *Z*-axis | Computed | m/s |
| BdyFrm | Cg | Disp | x | Vehicle CG displacement along the vehicle-fixed *x*-axis | Computed | m |
| | | | y | Vehicle CG displacement along the vehicle-fixed *y*-axis | 0 | m |
| | | | z | Vehicle CG displacement along the vehicle-fixed *z*-axis | Computed | m |
| | | Vel | xdot | Vehicle CG velocity along the vehicle-fixed *x*-axis | Computed | m/s |
| | | | ydot | Vehicle CG velocity along the vehicle-fixed *y*-axis | 0 | m/s |
| | | | zdot | Vehicle CG velocity along the vehicle-fixed *z*-axis | Computed | m/s |
| | | AngVel | p | Vehicle angular velocity about the vehicle-fixed *x*-axis (roll rate) | 0 | rad/s |
| | | | q | Vehicle angular velocity about the vehicle-fixed y-axis (pitch rate) | Computed | rad/s |
| | | | r | Vehicle angular velocity about the vehicle-fixed *z*-axis (yaw rate) | 0 | rad/s |
| | | Accel | ax | Vehicle CG acceleration along the vehicle-fixed *x*-axis | Computed | gn |
| | | | ay | Vehicle CG acceleration along the vehicle-fixed *y*-axis | 0 | gn |
| | | | az | Vehicle CG acceleration along the vehicle-fixed *z*-axis | Computed | gn |
| | Forces | Body | Fx | Net force on vehicle CG along the vehicle-fixed *x*-axis | Computed | N |
| | | | Fy | Net force on vehicle CG along the vehicle-fixed *y*-axis | 0 | N |
| | | | Fz | Net force on vehicle CG along the vehicle-fixed *z*-axis | Computed | N |

| Signal | | | | | Description | Value | Units |
|---|---|---|---|---|---|---|---|
| | | Ext | Fx | | External force on vehicle CG along the vehicle-fixed *x*-axis | Computed | N |
| | | | Fy | | External force on vehicle CG along the vehicle-fixed *y*-axis | Computed | N |
| | | | Fz | | External force on vehicle CG along the vehicle-fixed *z*-axis | Computed | N |
| | | FrntAxl | Fx | | Longitudinal force on front axle, along the vehicle-fixed *x*-axis | Computed | N |
| | | | Fy | | Lateral force on front axle, along the vehicle-fixed *y*-axis | 0 | N |
| | | | Fz | | Normal force on front axle, along the vehicle-fixed *z*-axis | Computed | N |
| | | RearAxl | Fx | | Longitudinal force on rear axle, along the vehicle-fixed *x*-axis | Computed | N |
| | | | Fy | | Lateral force on rear axle, along the vehicle-fixed *y*-axis | 0 | N |
| | | | Fz | | Normal force on rear axle, along the vehicle-fixed *z*-axis | Computed | N |
| | | Tires | FrntTire | Fx | Front tire force, along the vehicle-fixed *x*-axis | 0 | N |
| | | | | Fy | Front tire force, along the vehicle-fixed *y*-axis | 0 | N |
| | | | | Fz | Front tire force, along the vehicle-fixed *z*-axis | Computed | N |
| | | | RearTire | Fx | Rear tire force, along the vehicle-fixed *x*-axis | 0 | N |
| | | | | Fy | Rear tire force, along the vehicle-fixed *y*-axis | 0 | N |
| | | | | Fz | Rear tire force, along the vehicle-fixed *z*-axis | Computed | N |
| | | Drag | Fx | | Drag force on vehicle CG along the vehicle-fixed *x*-axis | Computed | N |

| Signal | | | | Description | Value | Units |
|---|---|---|---|---|---|---|
| | | | Fy | Drag force on vehicle CG along the vehicle-fixed *y*-axis | Computed | N |
| | | | Fz | Drag force on vehicle CG along the vehicle-fixed *z*-axis | Computed | N |
| | | Grvty | Fx | Gravity force on vehicle CG along the vehicle-fixed *x*-axis | Computed | N |
| | | | Fy | Gravity force on vehicle CG along the vehicle-fixed *y*-axis | `0` | N |
| | | | Fz | Gravity force on vehicle CG along the vehicle-fixed *z*-axis | Computed | N |
| | Moments | Body | Mx | Body moment on vehicle CG about the vehicle-fixed *x*-axis | `0` | N·m |
| | | | My | Body moment on vehicle CG about the vehicle-fixed *y*-axis | Computed | N·m |
| | | | Mz | Body moment on vehicle CG about the vehicle-fixed *z*-axis | `0` | N·m |
| | | Drag | Mx | Drag moment on vehicle CG about the vehicle-fixed *x*-axis | `0` | N·m |
| | | | My | Drag moment on vehicle CG about the vehicle-fixed *y*-axis | Computed | N·m |
| | | | Mz | Drag moment on vehicle CG about the vehicle-fixed *z*-axis | `0` | N·m |
| | | Ext | Fx | External moment on vehicle CG about the vehicle-fixed *x*-axis | `Computed` | N·m |
| | | | Fy | External moment on vehicle CG about the vehicle-fixed *y*-axis | `Computed` | N·m |
| | | | Fz | External moment on vehicle CG about the vehicle-fixed *z*-axis | `Computed` | N·m |
| | FrntAxl | Disp | x | Front axle displacement along the vehicle-fixed *x*-axis | Computed | m |

| Signal | | | | Description | Value | Units |
|---|---|---|---|---|---|---|
| | | | y | Front axle displacement along the vehicle-fixed *y*-axis | 0 | m |
| | | | z | Front axle displacement along the vehicle-fixed *z*-axis | Computed | m |
| | | Vel | xdot | Front axle velocity along the vehicle-fixed *x*-axis | Computed | m/s |
| | | | ydot | Front axle velocity along the vehicle-fixed *y*-axis | 0 | m/s |
| | | | zdot | Front axle velocity along the vehicle-fixed *z*-axis | Computed | m/s |
| | | Steer | WhlAngFL | Front left wheel steering angle | Computed | rad |
| | | | WhlAngFR | Front right wheel steering angle | Computed | rad |
| | RearAxl | Disp | x | Rear axle displacement along the vehicle-fixed *x*-axis | Computed | m |
| | | | y | Rear axle displacement along the vehicle-fixed *y*-axis | 0 | m |
| | | | z | Rear axle displacement along the vehicle-fixed *z*-axis | Computed | m |
| | | Vel | xdot | Rear axle velocity along the vehicle-fixed *x*-axis | Computed | m/s |
| | | | ydot | Rear axle velocity along the vehicle-fixed *y*-axis | 0 | m/s |
| | | | zdot | Rear axle velocity along the vehicle-fixed *z*-axis | Computed | m/s |
| | | Steer | WhlAngRL | Rear left wheel steering angle | Computed | rad |
| | | | WhlAngRR | Rear right wheel steering angle | Computed | rad |
| | Pwr | PwrExt | | Applied external power | Computed | W |
| | | Drag | | Power loss due to drag | Computed | W |
| PwrInfo | PwrTrnsfrd | PwrFxExt | | Externally applied longitudinal force power | Computed | W |
| | | PwrFzExt | | Externally applied longitudinal force power | Computed | W |
| | | PwrMyExt | | Externally applied pitch moment power | Computed | W |

| Signal | | | Description | Value | Units |
|---|---|---|---|---|---|
| | | PwrFwFx | Longitudinal force applied at the front axle | Computed | W |
| | | PwrFwRx | Longitudinal force applied at the rear axle | Computed | W |
| | PwrNotTrnsfrd | PwrFsF | Internal power transferred between suspension and vehicle body at the front axle | Computed | W |
| | | PwrFsR | Internal power transferred between suspension and vehicle body at the rear axle | Computed | W |
| | | PwrFxDrag | Longitudinal drag force power | Computed | W |
| | | PwrFzDrag | Vertical drag force power | Computed | W |
| | | PwrMyDrag | Drag pitch moment power | Computed | W |
| | | PwrFsb | Total suspension damping power | Computed | W |
| | PwrStored | PwrStoredGrvty | Rate change in gravitational potential energy | Computed | W |
| | | PwrStoredxdot | Rate of change of longitudinal kinetic energy | Computed | W |
| | | PwrStoredzdot | Rate of change of longitudinal kinetic energy | Computed | W |
| | | PwrStoredq | Rate of change of rotational pitch kinetic energy | Computed | W |
| | | PwrStoredFsFzSprng | Stored spring energy from front suspension | Computed | W |
| | | PwrStoredFsRzSprng | Stored spring energy from rear suspension | Computed | W |

**xdot — Vehicle longitudinal velocity**
scalar

Vehicle CG velocity along the vehicle-fixed $x$-axis, in m/s.

**FzF — Front axle normal force**
scalar

Normal force on front axle, $Fz_F$, along the vehicle-fixed $z$-axis, in N.

**FzR — Rear axle normal force**
scalar

Normal force on rear axle, $Fz_R$, along the vehicle-fixed $z$-axis, in N.

## Parameters

#### Options

**`External forces` — `FExt` input port**
`off` (default) | `on`

Specify to create input port `FExt`.

**`External moments` — `MExt` input port**
`off` (default) | `on`

Specify to create input port `MExt`.

**`Air temperature` — `AirTemp` input port**
`off` (default) | `on`

Specify to create input port `AirTemp`.

#### Longitudinal

**`Number of wheels on front axle, NF` — Front wheel count**
`2` (default) | `scalar`

Number of wheels on front axle, $N_F$. The value is dimensionless.

**`Number of wheels on rear axle, NR` — Rear wheel count**
`2` (default) | `scalar`

Number of wheels on rear axle, $N_R$. The value is dimensionless.

**`Mass, m` — Vehicle mass**
`1200` (default) | `scalar`

Vehicle mass, $m$, in kg.

**`Horizontal distance from CG to front axle, a` — Front axle distance**
`1.4` (default) | `scalar`

Horizontal distance $a$ from the vehicle CG to the front wheel axle, in m.

**`Horizontal distance from CG to rear axle, b` — Rear axle distance**
`1.8` (default) | `scalar`

Horizontal distance $b$ from the vehicle CG to the rear wheel axle, in m.

**`CG height above axles, h` — Height**
`0.35` (default) | `scalar`

Height of vehicle CG above the axles, $h$, in m.

**`Drag coefficient, Cd` — Drag**
`.3` (default) | `scalar`

Air drag coefficient, $C_d$. The value is dimensionless.

**`Frontal area, Af` — Area**
`2` (default) | `scalar`

Effective vehicle cross-sectional area, $A_f$ to calculate the aerodynamic drag force on the vehicle, in m^2.

**`Initial position, x_o` — Position**
`0` (default) | `scalar`

Vehicle body longitudinal initial position along earth-fixed $x$-axis, $x_o$, in m.

**`Initial velocity, xdot_o` — Velocity**
`0` (default) | `scalar`

Vehicle body longitudinal initial velocity along earth-fixed $x$-axis, $\dot{x}_0$, in m/s.

#### Vertical

**`Lift coefficient, Cl` — Lift**
`.1` (default) | `scalar`

Lift coefficient, $C_l$. The value is dimensionless.

**`Initial vertical position, z_o` — Position**
`-.35` (default) | `scalar`

Initial vertical CG position, $z_o$, along the vehicle-fixed $z$-axis, in m.

**`Initial vertical velocity, zdot_o` — Velocity**
`0` (default) | `scalar`

Initial vertical CG velocity, $zdot_o$, along the vehicle-fixed $z$-axis, in m.

#### Pitch

**`Inertia, Iyy` — About body y-axis**
`3500` (default) | `scalar`

Vehicle body moment of inertia about body $z$-axis.

**`Pitch drag moment coefficient, Cpm` — Drag coefficient**
`.1` (default) | `scalar`

Pitch drag moment coefficient. The value is dimensionless.

**`Initial pitch angle, theta_o` — Pitch**
`0` (default) | `scalar`

Initial pitch angle about body $z$-axis, in rad.

**`Initial angular velocity, q_o` — Pitch velocity**
`0` (default) | `scalar`

Initial vehicle body angular velocity about body $z$-axis, in rad/s.

#### Suspension

**`Front axle stiffness force data, FskF` — Force**
`[-50, -1, 0, 2, 3, 52].*1.5e4` (default) | `vector`

Front axle stiffness force data, $Fk_F$, in N.

**Dependencies**

To enable this parameter, for the **Ground interaction type** parameter, select `Grade angle` or `Axle displacement, velocity`.

**`Front axle displacement data, dzsF` — Displacement**
`[-5e-3, -1e-4, 0, .2, .2001, .2051]` (default) | `vector`

Front axle displacement data, in m.

**Dependencies**

To enable this parameter, for the **Ground interaction type** parameter, select `Grade angle` or `Axle displacement, velocity`.

**`Front axle damping force data, FsbF` — Damping force**
`[-10000 -100 -10 0 10 100 10000]` (default) | `vector`

Front axle damping force, in N.

**Dependencies**

To enable this parameter, for the **Ground interaction type** parameter, select `Grade angle` or `Axle displacement, velocity`.

**`Front axle velocity data, dzdotsF` — Velocity**
`[-10 -1 -.1 0 .1 1 10]` (default) | `vector`

Front axle velocity data, in m/s.

**Dependencies**

To enable this parameter, for the **Ground interaction type** parameter, select `Grade angle` or `Axle displacement, velocity`.

**`Rear axle stiffness force data, FskR` — Force**
`[-50, -1, 0, 2, 3, 52].*1e4` (default) | `vector`

Rear axle stiffness force data, in N.

**Dependencies**

To enable this parameter, for the **Ground interaction type** parameter, select `Grade angle` or `Axle displacement, velocity`.

**`Rear axle displacement data, dzsR` — Displacement**
`[-5e-3, -1e-4, 0, .2, .2001, .2051]` (default) | `vector`

Rear axle displacement data, in m.

#### Dependencies

To enable this parameter, for the **Ground interaction type** parameter, select `Grade angle` or `Axle displacement, velocity`.

**`Rear axle damping force data, FsbR` — Damping force**
`[-10000 -100 -10 0 10 100 10000]` (default) | `vector`

Rear axle damping force, in N.

#### Dependencies

To enable this parameter, for the **Ground interaction type** parameter, select `Grade angle` or `Axle displacement, velocity`.

**`Rear axle velocity data, dzdotsR` — Velocity**
`[-10 -1 -.1 0 .1 1 10]` (default) | `vector`

Rear axle velocity data, in m/s.

#### Dependencies

To enable this parameter, for the **Ground interaction type** parameter, select `Grade angle` or `Axle displacement, velocity`.

#### Environment

**`Absolute air pressure, Pabs` — Pressure**
`101325` (default) | `scalar`

Environmental air absolute pressure, $P_{abs}$, in Pa.

**`Air temperature, Tair` — Ambient air temperature**
`273` (default) | `scalar`

Ambient air temperature, $T_{air}$, in K.

#### Dependencies

To enable this parameter, clear **Air temperature**.

**`Gravitational acceleration, g` — Gravity**
`9.81` (default)

Gravitational acceleration, $g$, in $m/s^2$.

## References

[1] Gillespie, Thomas. *Fundamentals of Vehicle Dynamics*. Warrendale, PA: Society of Automotive Engineers, 1992.

[2] Vehicle Dynamics Standards Committee. *Vehicle Dynamics Terminology*. SAE J670. Warrendale, PA: Society of Automotive Engineers, 2008.

[3] Technical Committee. *Road vehicles — Vehicle dynamics and road-holding ability — Vocabulary*. ISO 8855:2011. Geneva, Switzerland: International Organization for Standardization, 2011.

## Extended Capabilities

### C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

## See Also

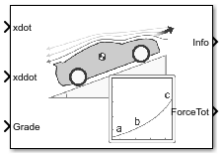
Vehicle Body 1DOF Longitudinal | Vehicle Body Total Road Load

**Introduced in R2017a**

# Vehicle Body Total Road Load

Vehicle motion using coast-down testing coefficients

**Library:** Powertrain Blockset / Vehicle Dynamics
Vehicle Dynamics Blockset / Vehicle Body




## Description

The Vehicle Body Total Road Load block implements a one degree-of-freedom (1DOF) rigid vehicle model using coast-down testing coefficients. You can use this block in a vehicle model to represent the load that the driveline and chassis applies to a transmission or engine. It is suitable for system-level performance, component sizing, fuel economy, or drive cycle tracking studies. The block calculates the dynamic powertrain load with minimal parameterization or computational cost.

You can configure the block for kinematic, force, or total power input.

- Kinematic — Block uses the vehicle longitudinal velocity and acceleration to calculate the tractive force and power.
- Force — Block uses the tractive force to calculate the vehicle longitudinal displacement and velocity.
- Power — Block uses the engine or transmission power to calculate the vehicle longitudinal displacement and velocity.

### Dynamics

To calculate the total road load acting on the vehicle, the block implements this equation.

$$F_{road} = a + b\dot{x} + c\dot{x}^2 + mg\sin(\theta)$$

To determine the coefficients $a$, $b$, and $c$, you can use a test procedure similar to the one described in *Road Load Measurement and Dynamometer Simulation Using Coastdown Techniques*. You can also use Simulink® Design Optimization™ to fit the coefficients to measured data.

To calculate the vehicle motion, the block uses Newton's law for rigid bodies.

$$F_{total} = m\ddot{x} + F_{road}$$

Total power input is a product of the total force and longitudinal velocity. Power due to road and gravitational forces is a product of the road force and longitudinal velocity.

$$P_{total} = F_{total}\dot{x}$$
$$P_{road} = F_{road}\dot{x}$$

### Power Accounting

For the power accounting, the block implements these equations.

| Bus Signal | | | Description | Variable | Equations |
|---|---|---|---|---|---|
| PwrInfo | PwrTrnsfrd — Power transferred between blocks<br>• Positive signals indicate flow into block<br>• Negative signals indicate flow out of block | PwrFxExt | Externally applied force power | $P_{FxExt}$ | $P_{FxExt} = F_{total}\dot{x}$ |
| | PwrNotTrnsfrd — Power crossing the block boundary, but not transferred<br>• Positive signals indicate an input<br>• Negative signals indicate a loss | PwrFxDrag | Drag force power | $P_D$ | $P_d = -(a + b\dot{x} + c\dot{x}^2)\dot{x}$ |
| | PwrStored — Stored energy rate of change<br>• Positive signals indicate an increase<br>• Negative signals indicate a decrease | wrStoredGrvty | Rate change in gravitational potential energy | $P_g$ | $P_g = -mg\dot{Z}$ |
| | | PwrStoredxdot | Rate in change of longitudinal kinetic energy | $P_{xdot}$ | $P_{\dot{x}} = m\ddot{x}\dot{x}$ |

The equations use these variables.

| | |
|---|---|
| $a$ | Steady-state rolling resistance coefficient |
| $b$ | Viscous driveline and rolling resistance coefficient |
| $c$ | Aerodynamic drag coefficient |
| $g$ | Gravitational acceleration |
| $x$ | Vehicle longitudinal displacement with respect to ground, in the vehicle-fixed frame |
| $\dot{x}$ | Vehicle longitudinal velocity with respect to ground, in the vehicle-fixed frame |
| $\ddot{x}$ | Vehicle longitudinal acceleration with respect to ground, vehicle-fixed frame |
| $m$ | Vehicle body mass |
| $\Theta$ | Road grade angle |
| $F_{total}$ | Total force acting on vehicle |
| $F_{road}$ | Resistive road load due to losses and gravitational load |
| $P_{total}$ | Total tractive input power |
| $P_{road}$ | Total power due to losses and gravitational load |
| $\dot{Z}$ | Vehicle vertical velocity along the vehicle-fixed z-axis |

## Ports

### Input

#### xdot — Vehicle longitudinal velocity
scalar

Vehicle total longitudinal velocity, $\dot{x}$, in m/s.

##### Dependencies

To enable this port, for the **Input Mode** parameter, select `Kinematic`.

#### xddot — Vehicle longitudinal acceleration
scalar

Vehicle total longitudinal acceleration, $\ddot{x}$, in m/s^2.

##### Dependencies

To enable this port, for the **Input Mode** parameter, select `Kinematic`.

#### PwrTot — Tractive input power
scalar

Tractive input power, $P_{total}$, in W.

##### Dependencies

To enable this port, for the **Input Mode** parameter, select `Power`.

#### ForceTot — Tractive input force
scalar

Tractive input force, $F_{total}$, in N.

##### Dependencies

To enable this port, for the **Input Mode** parameter, select `Force`.

#### Grade — Road grade angle
scalar

Road grade angle, $\Theta$, in deg.

### Output

#### Info — Bus signal
bus

Bus signal containing these block calculations.

| Signal | | | | Description | Value | Units |
|---|---|---|---|---|---|---|
| Inner | Cg | Disp | X | Vehicle CG displacement along earth-fixed X-axis | Computed | m |

| Signal | | | | Description | Value | Units |
|---|---|---|---|---|---|---|
| tFrm | | | Y | Vehicle CG displacement along earth-fixed Y-axis | 0 | m |
| | | | Z | Vehicle CG displacement along earth-fixed Z-axis | Computed | m |
| | | Vel | Xdot | Vehicle CG velocity along earth-fixed X-axis | Computed | m/s |
| | | | Ydot | Vehicle CG velocity along earth-fixed Y-axis | 0 | m/s |
| | | | Zdot | Vehicle CG velocity along earth-fixed Z-axis | Computed | m/s |
| | | Ang | phi | Rotation of vehicle-fixed frame about the earth-fixed X-axis (roll) | 0 | rad |
| | | | theta | Rotation of vehicle-fixed frame about the earth-fixed Y-axis (pitch) | Computed | rad |
| | | | psi | Rotation of vehicle-fixed frame about the earth-fixed Z-axis (yaw) | 0 | rad |
| BdyFrm | Cg | Disp | x | Vehicle CG displacement along the vehicle-fixed x-axis | Computed | m |
| | | | y | Vehicle CG displacement along the vehicle-fixed y-axis | 0 | m |
| | | | z | Vehicle CG displacement along the vehicle-fixed z-axis | 0 | m |
| | | Vel | xdot | Vehicle CG velocity along the vehicle-fixed x-axis | Computed | m/s |
| | | | ydot | Vehicle CG velocity along the vehicle-fixed y-axis | 0 | m/s |
| | | | zdot | Vehicle CG velocity along the vehicle-fixed z-axis | 0 | m/s |
| | | Acc | ax | Vehicle CG acceleration along the vehicle-fixed x-axis | Computed | gn |
| | | | ay | Vehicle CG acceleration along the vehicle-fixed y-axis | 0 | gn |
| | | | az | Vehicle CG acceleration along the vehicle-fixed z-axis | 0 | gn |
| | Forces | Body | Fx | Net force on vehicle CG along the vehicle-fixed x-axis | Computed | N |
| | | | Fy | Net force on vehicle CG along the vehicle-fixed y-axis | 0 | N |
| | | | Fz | Net force on vehicle CG along the vehicle-fixed z-axis | 0 | N |
| | | Ext | Fx | External force on vehicle CG along the vehicle-fixed x-axis | Computed | N |

| Signal | | | | Description | Value | Units |
|---|---|---|---|---|---|---|
| | | | Fy | External force on vehicle CG along the vehicle-fixed y-axis | 0 | N |
| | | | Fz | External force on vehicle CG along the vehicle-fixed z-axis | 0 | N |
| | | Drag | Fx | Drag force on vehicle CG along the vehicle-fixed x-axis | Computed | N |
| | | | Fy | Drag force on vehicle CG along the vehicle-fixed y-axis | 0 | N |
| | | | Fz | Drag force on vehicle CG along the vehicle-fixed z-axis | 0 | N |
| | | Grvty | Fx | Gravity force on vehicle CG along the vehicle-fixed x-axis | Computed | N |
| | | | Fy | Gravity force on vehicle CG along the vehicle-fixed y-axis | 0 | N |
| | | | Fz | Gravity force on vehicle CG along the vehicle-fixed z-axis | Computed | N |
| | Pwr | PwrExt | | Applied external power | Computed | W |
| | | Drag | | Power loss due to drag | Computed | W |
| PwrInfo | PwrTrnsfrd | PwrFxExt | | Externally applied force power | $P_{FxExt}$ | W |
| | PwrNotTrnsfrd | PwrFxDrag | | Drag force power | $P_D$ | W |
| | PwrStored | wrStoredGrvty | | Rate change in gravitational potential energy | $P_g$ | W |
| | | PwrStoredxdot | | Rate in change of longitudinal kinetic energy | $P_{xdot}$ | W |

### xdot — Vehicle longitudinal velocity

scalar

Vehicle total longitudinal velocity, $\dot{x}$, in m/s.

#### Dependencies

To enable this port, for the **Input Mode** parameter, select Power or Force.

### ForceTot — Tractive input force

scalar

Tractive input force, $F_{total}$, in N.

#### Dependencies

To enable this port, for the **Input Mode** parameter, select Kinematic.

## Parameters

**`Input Mode` — Specify input mode**
`Kinematic` (default) | `Force` | `Power`

Specify the input type.

- `Kinematic` — Block uses the vehicle longitudinal velocity and acceleration to calculate the tractive force and power. Use this configuration for powertrain, driveline, and braking system design, or component sizing.
- `Force` — Block uses the tractive force to calculate the vehicle longitudinal displacement and velocity. Use this configuration for system-level performance, fuel economy, or drive cycle tracking studies.
- `Power` — Block uses the engine or transmission power to calculate the vehicle longitudinal displacement and velocity. Use this configuration for system-level performance, fuel economy, or drive cycle tracking studies.

**Dependencies**

This table summarizes the port and input mode configurations.

| Input Mode | Creates Ports |
|---|---|
| `Kinematic` | `xdot`<br>`xddot` |
| `Force` | `Force` |
| `Power` | `Power` |

**`Mass` — Vehicle body mass**
`1200` (default) | `scalar`

Vehicle body mass, *m*, in kg.

**`Rolling resistance coefficient, a` — Rolling**
`196` (default) | `scalar`

Steady-state rolling resistance coefficient, *a*, in N.

**`Rolling and driveline resistance coefficient, b` — Rolling and driveline**
`2.232` (default) | `scalar`

Viscous driveline and rolling resistance coefficient, *b*, in N*s/m.

**`Aerodynamic drag coefficient, c` — Drag**
`0.389` (default) | `scalar`

Aerodynamic drag coefficient, *c*, in N·s^2/m.

**`Gravitational acceleration, g` — Gravity**
`9.81` (default) | `scalar`

Gravitational acceleration, *g*, in m/s^2.

**`Initial position, x_o` — Position**
`0` (default) | `scalar`

Vehicle longitudinal initial position, in m.

**`Initial velocity, xdot_o` — Velocity**
`0` (default) | `scalar`

Vehicle longitudinal initial velocity with respect to ground, in m/s.

## References

[1] Gillespie, Thomas. *Fundamentals of Vehicle Dynamics*. Warrendale, PA: Society of Automotive Engineers (SAE), 1992.

[2] Light Duty Vehicle Performance And Economy Measure Committee. *Road Load Measurement and Dynamometer Simulation Using Coastdown Techniques*. Standard J1263_201003. SAE International, March 2010.

## Extended Capabilities

### C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

## See Also

Drive Cycle Source | Vehicle Body 1DOF Longitudinal | Vehicle Body 3DOF Longitudinal

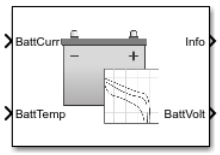
### Introduced in R2017a

# Energy Storage Blocks

# Datasheet Battery

Lithium-ion, lithium-polymer, or lead-acid battery

**Library:** Powertrain Blockset / Energy Storage and Auxiliary Drive / Datasheet Battery




## Description

The Datasheet Battery block implements a lithium-ion, lithium-polymer, or lead-acid battery that you can parameterize using manufacturer data. To create the open-circuit voltage and internal resistance parameters that you need for the block, use the manufacturer discharge characteristics by temperature data. For an example, see “Generate Parameter Data for Datasheet Battery Block”.

To determine the battery output voltage, the block uses lookup tables for the battery open-circuit voltage and the internal resistance. The lookup tables are functions of the state-of charge (SOC) and battery temperature, characterizing the battery performance at various operating points:

$$E_m = f(SOC)$$
$$R_{int} = f(T, SOC)$$

To calculate the voltage, the block implements these equations.

$$V_T = E_m + I_{batt}R_{int}$$
$$I_{batt} = \frac{I_{in}}{N_p}$$
$$V_{out} = \begin{cases} N_s V_T & \text{unfiltered} \\ \dfrac{V_{out}}{\tau s + 1} & \text{filtered} \end{cases}$$
$$SOC = \frac{1}{Cap_{batt}} \int_0^t I_{batt} dt$$
$$Ld_{AmpHr} = \int_0^t I_{batt} dt$$

Positive current indicates battery discharge. Negative current indicates battery charge.

### Power Accounting

For the power accounting, the block implements these equations.

| Bus Signal | | | Description | Equations |
|---|---|---|---|---|
| `PwrInfo` | `PwrTrnsfrd` — Power transferred between blocks<br>• Positive signals indicate flow into block<br>• Negative signals indicate flow out of block | `PwrLdBatt` | Battery network power | $V_{batt} = V_{out}$ OR $\frac{V_{out}}{\tau s + 1}$<br>$P_{batt} = -V_{batt} I_{batt}$<br>$P_{LdBatt} = -P_{batt}$ |
| | `PwrNotTrnsfrd` — Power crossing the block boundary, but not transferred<br>• Positive signals indicate an input<br>• Negative signals indicate a loss | `PwrLossBatt` | Battery network power loss | $P_{LossBatt} = -N_p N_s I_{batt}^2 R_{int}$ |
| | `PwrStored` — Stored energy rate of change<br>• Positive signals indicate an increase<br>• Negative signals indicate a decrease | `PwrStoredBatt` | Battery network power stored | $P_{StoredBatt} = P_{Batt} + P_{LossBatt}$ |

The equations use these variables.

| | |
|---|---|
| $SOC$ | State-of-charge |
| $E_m$ | Battery open-circuit voltage |
| $I_{batt}$ | Per module battery current |
| $P_{LdBatt}$ | Battery network power |
| $P_{batt}$ | Battery power |
| $P_{LossBatt}$ | Battery network power loss |
| $P_{StoredBatt}$ | Battery network power stored |
| $I_{in}$ | Combined current flowing from the battery network |
| $R_{int}$ | Battery internal resistance |
| $N_s$ | Number of cells in series |
| $N_p$ | Number of cells in parallel |
| $V_{out}$, $V_{batt}$ | Combined voltage of the battery network |
| $V_T$ | Per module battery voltage |
| $Cap_{batt}$ | Battery capacity |
| $Ld_{AmpHr}$ | Battery energy |

## Ports

### Inputs

#### `CapInit` — Battery capacity

`scalar`

Rated battery capacity at the nominal temperature, $Cap_{batt}$, in Ah.

#### Dependencies

To create this port, select `External Input` for the **Initial battery capacity** parameter.

### `BattCurr` — Battery load current

`scalar`

Combined current flowing from the battery network, $I_{in}$, in A.

### `BattTemp` — Battery temperature

`scalar`

Temperature measured at the battery housing, $T$, in K.

#### Output

### `Info` — Bus signal

bus

Bus signal containing these block calculations.

| Signal | | | Description | Variable | Units |
|---|---|---|---|---|---|
| `BattCurr` | | | Combined current flowing from the battery network | $I_{batt}$ | A |
| `BattAmpHr` | | | Battery energy | $Ld_{AmpHr}$ | A*h |
| `BattSoc` | | | State-of-charge capacity | $SOC$ | NA |
| `BattVolt` | | | Combined voltage of the battery network | $V_{out}$ | V |
| `BattPwr` | | | Battery network power | $P_{batt}$ | W |
| `PwrInfo` | `PwrTrnsfrd` | `PwrLdBatt` | Battery network power | $P_{LdBatt}$ | W |
| | `PwrNotTrnsfrd` | `PwrLossBatt` | Battery network power loss | $P_{LossBatt}$ | W |
| | `PwrStored` | `PwrStoredBatt` | Battery network power stored | $P_{StoredBatt}$ | W |

### `BattVolt` — Battery output voltage

`scalar`

Combined voltage of the battery network, $V_{out}$, in V.

## Parameters

#### Block Options

### `Initial battery capacity` — Input or parameter

`Parameter` (default) | `External Input`

Initial battery capacity, $Cap_{batt}$, in Ah.

#### Dependencies

| Block Parameter Initial battery capacity Option | Creates |
|---|---|
| `External Input` | Input port `CapInit` |
| `Parameter` | Parameter **Initial battery capacity, BattCapInit** |

**`Output battery voltage` — Unfiltered or Filter**
`Unfiltered` (default) | `Filtered`

Select `Filtered` to apply a first-order filter to the output batter voltage.

#### Dependencies

Setting **Output battery voltage** parameter to `Filtered` creates these parameters:

- **Output battery voltage time constant, Tc**
- **Output battery voltage initial value, Vinit**

**`Rated capacity at nominal temperature, BattChargeMax` — Constant**
`100` (default) | `scalar`

Rated battery capacity at the nominal temperature, in Ah.

**`Open circuit voltage table data, Em` — 1-D lookup table**
1-by-`P` matrix

Open-circuit voltage data curve, $E_m$, as a function of the discharged capacity for `P` operating points, in V.

**`Open circuit voltage breakpoints 1, CapLUTBp` — Breakpoints**
1-by-`P` matrix

Discharge capacity breakpoints for `P` operating points, dimensionless.

Although this parameter is the same as the **Battery capacity breakpoints 2, CapSOCBp** parameter, the block uses unique parameters for calibration flexibility.

**`Internal resistance table data, RInt` — 2-D lookup table**
`N`-by-`M` matrix

Internal resistance map, $R_{int}$, as a function of `N` temperatures and `M` SOCs, in ohms.

**`Battery temperature breakpoints 1, BattTempBp` — Breakpoints**
`[243.1 253.1 263.1 273.1 283.1 298.1 313.1]` (default) | 1-by-`N` matrix

Battery temperature breakpoints for `N` temperatures, in K.

**`Battery capacity breakpoints 2, CapSOCBp` — Breakpoints**
`[0 0.2 0.4 0.6 0.8 1]` (default) | 1-by-`M` matrix

Battery capacity breakpoints for `M` SOCs, dimensionless.

Although this parameter is the same as the **Open circuit voltage breakpoints 1, CapLUTBp** parameter, the block uses unique parameters for calibration flexibility.

**`Number of cells in series, Ns` — Integer**
`1` (default) | `scalar`

Number of cells in series, dimensionless, $N_s$.

**`Number of cells in parallel, Np` — Integer**
`1` (default) | `scalar`

Number of cells in parallel, dimensionless, $N_p$.

**`Initial battery capacity, BattCapInit` — Capacity**
`100` (default) | `scalar`

Initial battery capacity, $Cap_{batt}$, in Ah.

**Dependencies**

| Block Parameter Initial battery capacity Option | Creates |
|---|---|
| `External Input` | Input port `CapInit` |
| `Parameter` | Parameter **Initial battery capacity, BattCapInit** |

**`Output battery voltage time constant, Tc` — Filter time constant**
`1/1000` (default) | `scalar`

Output battery voltage time constant, $T_c$, in s. Used in a first-order voltage filter.

**Dependencies**

Setting **Output battery voltage** parameter to `Filtered` creates these parameters:

- **Output battery voltage time constant, Tc**
- **Output battery voltage initial value, Vinit**

**`Output battery voltage initial value` — Filter initial voltage**
`4.221` (default) | `scalar`

Output battery voltage initial value, $V_{init}$, in V. Used in a first-order voltage filter.

**Dependencies**

Setting **Output battery voltage** parameter to `Filtered` creates these parameters:

- **Output battery voltage time constant, Tc**
- **Output battery voltage initial value, Vinit**

## References

[1] Arrhenius, S.A. “Über die Dissociationswärme und den Einflusß der Temperatur auf den Dissociationsgrad der Elektrolyte.” *Journal of Physical Chemistry*. 4 (1889): 96–116.

[2] Connors, K. *Chemical Kinetics*. New York: VCH Publishers, 1990.

[3] Ji, Yan, Yancheng Zhang, and Chao-Yang Wang. *Journal of the Electrochemical Society*. Volume 160, Issue 4 (2013), A636-A649.

## Extended Capabilities

### C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

## See Also

Equivalent Circuit Battery | Estimation Equivalent Circuit Battery

### Topics

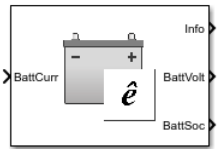
“Generate Parameter Data for Datasheet Battery Block”
Battery Modeling

### Introduced in R2017a

# Estimation Equivalent Circuit Battery

Resistor-capacitor (RC) circuit battery that creates lookup tables

**Library:** Powertrain Blockset / Energy Storage and Auxiliary Drive / Network Battery




## Description

The Estimation Equivalent Circuit Battery block implements a resistor-capacitor (RC) circuit battery model that you can use to create lookup tables for the Equivalent Circuit Battery block. The lookup tables are functions of the state-of-charge (SOC).

The Estimation Equivalent Circuit Battery block calculates the combined voltage of the network battery using parameter lookup tables. The tables are functions of the SOC. To acquire the SOC, the block integrates the charge and discharge currents.

Specifically, the block implements these parameters as lookup tables that are functions of the SOC:

- Series resistance, $R_o$=ƒ($SOC$)
- Battery open-circuit voltage, $E_m$=ƒ($SOC$)
- Network resistance, $R_n$=ƒ($SOC$)
- Network capacitance, $C_n$=ƒ($SOC$)

To calculate the combined voltage of the battery network, the block uses these equations.

$$V_T = E_m - I_{batt}R_o - \sum_{1}^{n} V_n$$

$$V_n = \int_0^t \left[\frac{I_{batt}}{C_n} - \frac{V_n}{R_n C_n}\right] dt$$

$$SOC = \frac{-1}{C_{batt}} \int_0^t I_{batt} dt$$

$$I_{batt} = I_{in}$$

$$V_{out} = V_T$$

Positive current indicates battery discharge. Negative current indicates battery charge.

The equations use these variables.

| | |
|---|---|
| $SOC$ | State-of-charge |
| $E_m$ | Battery open-circuit voltage |

| | |
|---|---|
| $I_{batt}$ | Per module battery current |
| $I_{in}$ | Combined current flowing from the battery network |
| $R_o$ | Series resistance |
| $n$ | Number of RC pairs in series |
| $V_{out}$, $V_T$ | Combined voltage of the battery network |
| $V_n$ | Voltage for $n$-th RC pair |
| $R_n$ | Resistance for $n$-th RC pair |
| $C_n$ | Capacitance for $n$-th RC pair |
| $C_{batt}$ | Battery capacity |

## Ports

### Inputs

#### `BattCurr` — Battery network current

`scalar`

Combined current flowing from the battery network, $I_{in}$, in A.

### Output

#### `Info` — Bus signal

bus

Bus signal containing these block calculations.

| Signal | Description | Variable | Units |
|---|---|---|---|
| `CapVolt` | Voltage for $n$-th RC pair | $V_n$ | V |

#### `BattVolt` — Battery output voltage

`scalar`

Combined voltage of the battery network, $V_{out}$, in V.

#### `BattSoc` — Battery SOC

`scalar`

Battery state-of-charge, *SOC*.

## Parameters

### Core Battery

#### `Number of series RC pairs` — RC pairs

1 (default) | 2 | 3 | 4 | 5

Number of series RC pairs. For lithium, typically 1 or 2.

#### `Open circuit voltage Em table data, Em` — Voltage table

`[3.8 3.8 3.8 3.8 3.8 3.8]` (default) | `array`

Open-circuit voltage table, $E_m$, in V. Function of SOC.

**Series resistance table data, R0 — Resistance**
`[0.01 0.01 0.01 0.01 0.01 0.01]` (default) | `array`

Series resistance table, $R_o$, in ohms. Function of SOC.

**State of charge breakpoints, SOC_BP — SOC breakpoints**
`[0 .2 .4 .6 .8 1]` (default) | `vector`

State-of-charge (SOC) breakpoints, dimensionless.

**Battery capacity, BattCap — Capacity**
`27.6250` (default) | `scalar`

Battery capacity, $C_{batt}$, in Ah.

**Initial battery capacity, BattCapInit — Capacity**
`27.6250` (default) | `scalar`

Initial battery capacity, $C_{batto}$, in Ah.

**Initial capacitor voltage, InitialCapVoltage — Voltage**
`0` (default) | `vector`

Initial capacitor voltage, in V. Dimension of vector must equal the **Number of series RC pairs**.

#### R and C Table Data

**Network resistance table data, R*n* — Lookup table**
`[0.005 0.005 0.005 0.005 0.005 0.005]` (default) | `array`

Network resistance table data for *n*-th RC pair, as a function of SOC, in ohms.

**Network capacitance table data, C*n* — Lookup table**
`[10000 10000 10000 10000 10000 10000]` (default) | `array`

Network capacitance table data for *n*-th RC pair, as a function of SOC, in F.

#### Cell Limits

**Upper Integrator Voltage Limit, Vu — Maximum**
`Inf` (default) | `scalar`

Upper voltage limit, in V.

**Lower Integrator Voltage Limit, Vl — Minimum**
`Inf` (default) | `scalar`

Lower voltage limit, in V.

## References

[1] Ahmed, R., J. Gazzarri, R. Jackey, S. Onori, S. Habibi, et al. "Model-Based Parameter Identification of Healthy and Aged Li-ion Batteries for Electric Vehicle Applications." *SAE International Journal of Alternative Powertrains*. doi:10.4271/2015-01-0252, 4(2):2015.

[2] Gazzarri, J., N. Shrivastava, R. Jackey, and C. Borghesani. "Battery Pack Modeling, Simulation, and Deployment on a Multicore Real Time Target." *SAE International Journal of Aerospace*. doi:10.4271/2014-01-2217, 7(2):2014.

[3] Huria, T., M. Ceraolo, J. Gazzarri, and R. Jackey. "High fidelity electrical model with thermal dependence for characterization and simulation of high power lithium battery cells." *IEEE® International Electric Vehicle Conference*. March 2012, pp. 1–8.

[4] Huria, T., M. Ceraolo, J. Gazzarri, and R. Jackey. "Simplified Extended Kalman Filter Observer for SOC Estimation of Commercial Power-Oriented LFP Lithium Battery Cells." *SAE Technical Paper 2013-01-1544*. doi:10.4271/2013-01-1544, 2013.

[5] Jackey, R. "A Simple, Effective Lead-Acid Battery Modeling Process for Electrical System Component Selection." *SAE Technical Paper 2007-01-0778*. doi:10.4271/2007-01-0778, 2007.

[6] Jackey, R., G. Plett, and M. Klein. "Parameterization of a Battery Simulation Model Using Numerical Optimization Methods." *SAE Technical Paper 2009-01-1381*. doi:10.4271/2009-01-1381, 2009.

[7] Jackey, R., M. Saginaw, T. Huria, M. Ceraolo, P. Sanghvi, and J. Gazzarri. "Battery Model Parameter Estimation Using a Layered Technique: An Example Using a Lithium Iron Phosphate Cell." *SAE Technical Paper 2013-01-1547*. Warrendale, PA: SAE International, 2013.

## Extended Capabilities

### C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

## See Also

Datasheet Battery | Equivalent Circuit Battery

### Topics

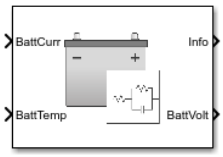
“Generate Parameter Data for Equivalent Circuit Battery Block”
Battery Modeling

### Introduced in R2017a

# Equivalent Circuit Battery

Resistor-capacitor (RC) circuit battery

**Library:** Powertrain Blockset / Energy Storage and Auxiliary Drive / Network Battery




## Description

The Equivalent Circuit Battery block implements a resistor-capacitor (RC) circuit battery that you can parameterize using equivalent circuit modeling (ECM). To simulate the state-of-charge (SOC) and terminal voltage, the block uses load current and internal core temperature.

The Equivalent Circuit Battery block calculates the combined voltage of the network battery using parameter lookup tables. The tables are functions of the SOC and battery temperature. You can use the Estimation Equivalent Circuit Battery block to help create the lookup tables.

Specifically, the Equivalent Circuit Battery block implements these parameters as lookup tables that are functions of the SOC and battery temperature:

- Series resistance, $R_o=f(SOC,T)$
- Battery open-circuit voltage, $E_m=f(SOC,T)$
- Battery capacity, $C_{batt}=f(T)$
- Network resistance, $R_n=f(SOC,T)$
- Network capacitance, $C_n=f(SOC,T)$

To calculate the combined voltage of the battery network, the block uses these equations.

$$V_T = E_m - I_{batt}R_o - \sum_1^n V_n$$

$$V_n = \int_0^t \left[\frac{I_{batt}}{C_n} - \frac{V_n}{R_n C_n}\right] dt$$

$$SOC = \frac{-1}{C_{batt}} \int_0^t I_{batt} dt$$

$$I_{batt} = \frac{I_{in}}{N_p}$$

$$V_{out} = N_s V_T$$

$$P_{BattLoss} = {I_{batt}}^2 R_0 + \sum_1^n \frac{{V_n}^2}{R_n}$$

$$Ld_{AmpHr} = \int_0^t I_{batt} dt$$

Positive current indicates battery discharge. Negative current indicates battery charge.

### Power Accounting

For the power accounting, the block implements these equations.

| Bus Signal | | | Description | Equations |
|---|---|---|---|---|
| `PwrInfo` | `PwrTrnsfrd` — Power transferred between blocks<br>• Positive signals indicate flow into block<br>• Negative signals indicate flow out of block | `PwrLdBatt` | Battery network power | $V_{batt} = V_{out}$ OR $\frac{V_{out}}{\tau s + 1}$<br>$P_{batt} = -V_{batt} I_{batt}$<br>$P_{LdBatt} = -P_{batt}$ |
| | `PwrNotTrnsfrd` — Power crossing the block boundary, but not transferred<br>• Positive signals indicate an input<br>• Negative signals indicate a loss | `PwrLossBatt` | Battery network power loss | $P_{LossBatt} = -\left({I_{batt}}^2 R_0 + \sum_1^n \frac{{V_n}^2}{R_n}\right)$ |
| | `PwrStored` — Stored energy rate of change<br>• Positive signals indicate an increase<br>• Negative signals indicate a decrease | `PwrStoredBatt` | Battery network power stored | $P_{StoredBatt} = P_{Batt} + P_{LossBatt}$ |

The equations use these variables.

| | |
|---|---|
| $SOC$ | State-of-charge |

| | |
|---|---|
| $E_m$ | Battery open-circuit voltage |
| $I_{batt}$ | Per module battery current |
| $I_{in}$ | Combined current flowing from the battery network |
| $R_o$ | Series resistance |
| $N_p$ | Number parallel branches |
| $N_p$ | Number of RC pairs in series |
| $V_{out}$, $V_T$ | Combined voltage of the battery network |
| $V_n$ | Voltage for $n$-th RC pair |
| $R_n$ | Resistance for $n$-th RC pair |
| $C_n$ | Capacitance for $n$-th RC pair |
| $C_{batt}$ | Battery capacity |
| $P_{batt}$ | Battery power |
| $P_{LossBatt}$ | Negative of battery network power loss |
| $P_{BattLoss}$ | Battery network power loss |
| $P_{StoredBatt}$ | Battery network power stored |
| $P_{LdBatt}$ | Battery network power |
| $T$ | Battery temperature |

## Ports

### Inputs

#### `CapInit` — Battery capacity

`scalar`

Rated battery capacity at the nominal temperature, $Cap_{batt}$, in Ah.

##### Dependencies

To create this port, select `External Input` for the **Initial battery capacity** parameter.

#### `BattCurr` — Battery network current

`scalar`

Combined current flowing from the battery network, $I_{in}$, in A.

#### `BattTemp` — Battery temperature

`scalar`

Battery temperature, $T$, in K.

### Output

#### `Info` — Bus signal

bus

Bus signal containing these block calculations.

| Signal | | | Description | Variable | Units |
|---|---|---|---|---|---|
| `BattCurr` | | | Combined current flowing from the battery network | $I_{batt}$ | A |
| `BattAmpHr` | | | Battery energy | $Ld_{AmpHr}$ | A*h |
| `BattSoc` | | | State-of-charge capacity | $SOC$ | NA |
| `BattVolt` | | | Combined voltage of the battery network | $V_{out}$ | V |
| `BattPwr` | | | Battery power | $P_{batt}$ | W |
| `PwrInfo` | `PwrTrnsfrd` | `PwrLdBatt` | Battery network power | $P_{LdBatt}$ | W |
| | `PwrNotTrnsfrd` | `PwrLossBatt` | Battery network power loss | $P_{LossBatt}$ | W |
| | `PwrStored` | `PwrStoredBatt` | Battery network power stored | $P_{StoredBatt}$ | W |

**`BattVolt` — Battery output voltage**
`scalar`

Combined voltage of the battery network, $V_{out}$, in V.

## Parameters

#### Block Options

**`Initial battery capacity` — Input or parameter**
`Parameter` (default) | `External Input`

Initial battery capacity, $Cap_{batt}$, in Ah.

#### Dependencies

| Block Parameter Initial battery capacity Option | Creates |
|---|---|
| `External Input` | Input port `CapInit` |
| `Parameter` | Parameter **Initial battery capacity, BattCapInit** |

**`Output battery voltage` — Unfiltered or Filter**
`Unfiltered` (default) | `Filtered`

Select `Filtered` to apply a first-order filter to the output batter voltage.

#### Dependencies

Setting **Output battery voltage** parameter to `Filtered` creates these parameters:

- **Output battery voltage time constant, Tc**
- **Output battery voltage initial value, Vinit**

#### Core Battery

**`Number of series RC pairs` — RC pairs**
1 (default) | 2 | 3 | 4 | 5

Number of series RC pairs. For lithium, typically 1 or 2.

**`Open circuit voltage Em table data, Em` — Voltage table**
`[3.5042 3.5136; 3.5573 3.5646; 3.6009 3.6153; 3.6393 3.6565; 3.6742 3.6889; 3.7121 3.7214; 3.7937 3.8078; 3.8753 3.8945; 3.97 3.9859; 4.0764 4.0821; 4.1924 4.193]` (default) | `array`

Open circuit voltage table, $E_m$, in V. Function of SOC and battery temperature.

**`Series resistance table data, R0` — Resistance**
`array`

Series resistance table, $R_o$, in ohms. Function of SOC and battery temperature.

**`State of charge breakpoints, SOC_BP` — SOC breakpoints**
`[0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1]` (default) | `vector`

State-of-charge (SOC) breakpoints, dimensionless.

**`Temperature breakpoints, Temperature_BP` — Battery**
`[293.15 313.15]` (default) | `vector`

Battery temperature breakpoints, K.

**`Battery capacity table, BattCap` — Capacity**
`[28 28]` (default) | `array`

Battery capacity, $C_{batt}$, in Ah. Function of battery temperature.

**`Initial battery capacity, BattCapInit` — Capacity**
28 (default) | `scalar`

Initial battery capacity, $Cap_{batt}$, in Ah.

#### Dependencies

| Block Parameter Initial battery capacity Option | Creates |
|---|---|
| `External Input` | Input port `CapInit` |
| `Parameter` | Parameter **Initial battery capacity, BattCapInit** |

**`Initial capacitor voltage, InitialCapVoltage` — Voltage**
`0` (default) | `array`

Initial capacitor voltage, in V. Dimension of vector must equal the **Number of series RC pairs**.

**`Output battery voltage time constant, Tc` — Filter time constant**
`1/1000` (default) | `scalar`

Output battery voltage time constant, $T_c$, in s. Used in a first-order voltage filter.

#### Dependencies

Setting **Output battery voltage** parameter to `Filtered` creates these parameters:

- **Output battery voltage time constant, Tc**
- **Output battery voltage initial value, Vinit**

**`Output battery voltage initial value, Vinit` — Filter initial voltage**
`4.193` (default) | `scalar`

Output battery voltage initial value, $V_{init}$, in V. Used in a first-order voltage filter.

#### Dependencies

Setting **Output battery voltage** parameter to `Filtered` creates these parameters:

- **Output battery voltage time constant, Tc**
- **Output battery voltage initial value, Vinit**

#### R and C Table Data

**`Network resistance table data, Rn` — Lookup table**
`[0.010342 0.0012244; 0.0067316 0.0011396; 0.0051156 0.0012661; 0.0043447 0.0012265; 0.0038826 0.0011163; 0.0034226 0.0009968; 0.003346 0.0011458; 0.0033222 0.001345; 0.0033201 0.0013091; 0.0032886 0.0010986; 0.0028114 0.0010309]` (default) | `array`

Network resistance table data for *n*-th RC pair, in ohms, as a function of SOC and battery temperature.

**`Network capacitance table data, Cn` — Lookup table**
`[2287.7 11897; 6122 24515; 18460 42098; 20975 44453; 15254 33098; 10440 24492; 13903 32975; 16694 40007; 15784 35937; 12165 26430; 9118 24795]` (default) | `array`

Network capacitance table data for *n*-th RC pair, in F, as a function of SOC and battery temperature.

#### Cell Limits

**`Upper integrator voltage limit, Vu` — Maximum**
`-Inf` (default) | `scalar`

Upper voltage limit, in V.

**`Lower integrator voltage limit, Vl` — Minimum**
`-Inf` (default) | `scalar`

Lower voltage limit, in V.

## References

[1] Ahmed, R., J. Gazzarri, R. Jackey, S. Onori, S. Habibi, et al. "Model-Based Parameter Identification of Healthy and Aged Li-ion Batteries for Electric Vehicle Applications." *SAE International Journal of Alternative Powertrains*. doi:10.4271/2015-01-0252, 4(2):2015.

[2] Gazzarri, J., N. Shrivastava, R. Jackey, and C. Borghesani. "Battery Pack Modeling, Simulation, and Deployment on a Multicore Real Time Target." *SAE International Journal of Aerospace*. doi:10.4271/2014-01-2217, 7(2):2014.

[3] Huria, T., M. Ceraolo, J. Gazzarri, and R. Jackey. "High fidelity electrical model with thermal dependence for characterization and simulation of high power lithium battery cells." *IEEE International Electric Vehicle Conference*. March 2012, pp. 1–8.

[4] Huria, T., M. Ceraolo, J. Gazzarri, and R. Jackey. "Simplified Extended Kalman Filter Observer for SOC Estimation of Commercial Power-Oriented LFP Lithium Battery Cells." *SAE Technical Paper 2013-01-1544*. doi:10.4271/2013-01-1544, 2013.

[5] Jackey, R. "A Simple, Effective Lead-Acid Battery Modeling Process for Electrical System Component Selection." *SAE Technical Paper 2007-01-0778*. doi:10.4271/2007-01-0778, 2007.

[6] Jackey, R., G. Plett, and M. Klein. "Parameterization of a Battery Simulation Model Using Numerical Optimization Methods." *SAE Technical Paper 2009-01-1381*. doi:10.4271/2009-01-1381, 2009.

[7] Jackey, R., M. Saginaw, T. Huria, M. Ceraolo, P. Sanghvi, and J. Gazzarri. "Battery Model Parameter Estimation Using a Layered Technique: An Example Using a Lithium Iron Phosphate Cell." *SAE Technical Paper 2013-01-1547*. Warrendale, PA: SAE International, 2013.

## Extended Capabilities

### C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

## See Also

Datasheet Battery | Estimation Equivalent Circuit Battery

### Topics

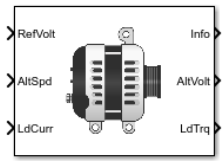
“Generate Parameter Data for Equivalent Circuit Battery Block”
Battery Modeling

### Introduced in R2017a

# Reduced Lundell Alternator

Reduced Lundell (claw-pole) alternator with an external voltage regulator

**Library:** Powertrain Blockset / Energy Storage and Auxiliary Drive / Alternator




## Description

The Reduced Lundell Alternator block implements a reduced Lundell (claw-pole) alternator with an external voltage regulator. The back-electromotive force (EMF) voltage is proportional to the input velocity and field current. The motor operates as a source torque to the internal combustion engine.

Use the Reduced Lundell Alternator block:

- To model an automotive electrical system
- In an engine model with a front-end accessory drive (FEAD)

The calculated motor shaft torque is in the opposite direction of the engine speed. You can:

- Tune the external voltage regulator to a desired bandwidth. The stator current and two diode drops reduce the stator voltage.
- Filter the load current to desired bandwidth. The load current has a lower saturation of 0 A.

The Reduced Lundell Alternator block implements equations for the electrical, control, and mechanical systems that use these variables.

### Electrical

To calculate voltages, the block uses these equations.

| Calculation | Equations |
|---|---|
| Alternator output voltage | $v_s = K_v i_f \omega - R_s i_s - 2V_d$ |
| Field winding voltage | $v_f = \quad R_f i_f + L_f \frac{di_f}{dt}$ |

### Control

The controller assumes no resistance or voltage drop.

| Calculation | Equations |
|---|---|
| Field winding voltage transform | $V_f(s) = R_f I_f(s) + sL_f I_f(s)$ |
| Field winding current transform | $I_f(s) = \frac{V_f(s)}{(R_f + sL_f)}$ |

| Calculation | Equations |
|---|---|
| Open loop electrical transfer function | $G(s) = \frac{V_s(s)}{V_f(s)} = \frac{K_v\omega}{(R_f + sL_f)}$ |
| Open loop voltage regulator transfer function | $G_C(s) = \frac{V_f(s)}{Vref(s)}$ |
| Closed loop transfer function | $T(s) = \frac{G(s)Gc(s)}{1 + G(s)Gc(s)}$ |
| Closed loop controller design | $T(s) = \frac{1}{\tau s + 1} \rightarrow \quad G(s)Gc(s) = \frac{1}{\tau s}$<br>$G_C(s) = K_g \left(K_p + \frac{K_i}{s}\right)$<br>$G(s)G_C(s) = \frac{K_v\omega}{(R_f + sL_f)} K_g \left(K_p + \frac{K_i}{s}\right)$<br>$K_p = L_f \ , K_i = R_f \ , and \quad K_g = \frac{2\pi f}{K_v\omega}$ |

### Mechanical

To calculate torques, the block uses these equations.

| Calculation | Equations |
|---|---|
| Electrical torque | $\tau_{elec} = (K_v i_f \omega) i_{load}$ |
| Frictional torque | $\tau_{friction} = K_b\omega$ |
| Windage torque | $\tau_{windage} = K_w\omega^2$ |
| Torque at start | $\tau_{start} = K_c$ when $\omega = 0$ |
| Motor shaft torque | $\tau_{mech} = \tau_{elec} + \tau_{friction} + \tau_{windage} + \tau_{start}$ |

### Power Accounting

For the power accounting, the block implements these equations.

| Bus Signal | | | Description | Variable | Equations |
|---|---|---|---|---|---|
| `PwrInfo` | `PwrTrnsfrd` — Power transferred between blocks<br>• Positive signals indicate flow into block<br>• Negative signals indicate flow out of block | `PwrMtr` | Mechanical power | $P_{mot}$ | $P_{mot} = \omega\tau_{mech}$ |
| | | `PwrBus` | Electrical power | $P_{bus}$ | $P_{bus} = -v_s i_{load}$ |

| Bus Signal | | | Description | Variable | Equations |
|---|---|---|---|---|---|
| | `PwrNotTrnsfrd` — Power crossing the block boundary, but not transferred<br>• Positive signals indicate an input<br>• Negative signals indicate a loss | `PwrLoss` | Motor power loss | $P_{loss}$ | $P_{loss} = \quad -(P_{mot} + P_{bus} - P_{ind})$ |
| | `PwrStored` — Stored energy rate of change<br>• Positive signals indicate an increase<br>• Negative signals indicate a decrease | `PwrInd` | Electrical winding loss | $P_{ind}$ | $P_{ind} = \quad L_f i_f \frac{di_f}{dt}$ |

The equations use these variables.

| | |
|---|---|
| $v_{ref}$ | Alternator output voltage command |
| $v_f$ | Field winding voltage |
| $i_f$ | Field winding current |
| $i_s$ | Stator winding current |
| $V_d$ | Diode voltage drop |
| $R_f$ | Field winding resistance |
| $R_s$ | Stator winding resistance |
| $L_f$ | Field winding inductance |
| $K_v$ | Voltage constant |
| $F_v$ | Voltage regulator bandwidth |
| $F_c$ | Input current filter bandwidth |
| $V_{fmax}$ | Field control voltage upper saturation limit |
| $V_{fmin}$ | Field control voltage lower saturation limit |
| $K_c$ | Coulomb friction coefficient |
| $K_b$ | Viscous friction coefficient |
| $K_w$ | Windage coefficient |
| $\omega$ | Motor shaft angular speed |
| $i_{load}$ | Alternator load current |
| $v_s$ | Alternator output voltage |
| $\tau_{mech}$, $T_{mech}$ | Motor shaft torque |

## Ports

### Inputs

#### `RefVolt` — Alternator output voltage command

`scalar`

Alternator output voltage command, in V.

#### `AltSpd` — Angular speed

`scalar`

Motor shaft input angular speed, in rad/s.

#### `LdCurr` — Alternator load current

`scalar`

Alternator load current, in A.

Do not connect the port to the alternator rated current, which is a constant value. The block uses the alternator load current as the stator winding current, $i_s$, to determine the alternator voltage and motor torque. If you connect the port to the rated alternator current, the block does not model the dynamic effect of load current changes on the voltage and motor torque.

### Output

#### `Info` — Bus signal

bus

Bus signal containing these block calculations.

| Signal | | | Description | Units |
|---|---|---|---|---|
| `FldVolt` | | | Field winding voltage | A |
| `FldFlux` | | | Field flux | Wb |
| `PwrInfo` | `PwrTrnsfrd` | `PwrMtr` | Mechanical power | W |
| | | `PwrBus` | Electrical power | W |
| | `PwrNotTrnsfrd` | `PwrLoss` | Motor power loss | W |
| | `PwrStored` | `PwrInd` | Electrical winding loss | W |

#### `AltVolt` — Alternator output voltage

`scalar`

Alternator output voltage, in V.

#### `LdTrq` — Motor shaft torque

`scalar`

Motor shaft torque, in N·m.

## Parameters

#### Machine Configuration

**`Voltage constant, Kv` — Constant**
`.1` (default) | `scalar`

Voltage constant, in V/rad/s.

**`Field winding resistance, Rf` — Resistance**
`0.2` (default) | `scalar`

Field winding resistance, in ohm.

**`Field winding inductance, Lf` — Inductance**
`0.002` (default) | `scalar`

Field winding inductance, in H.

**`Stator winding resistance, Rs` — Resistance**
`0.01` (default) | `scalar`

Stator winding resistance, in ohm.

**`Diode voltage drop, Vd` — Voltage**
`0.7` (default) | `scalar`

Diode voltage drop, in V.

#### Voltage Regulator

**`Regulator bandwidth, Fv` — Bandwidth**
`2000` (default) | `scalar`

The regulator bandwidth, in Hz.

**`Current filter bandwidth, Fc` — Bandwidth**
`1000` (default) | `scalar`

The current filter bandwidth, in Hz.

**`Field voltage max, Vfmax` — Maximum field voltage**
`100` (default) | `scalar`

The maximum field voltage, in V.

**`Field voltage min, Vfmin` — Minimum field voltage**
`-100` (default) | `scalar`

The minimum field voltage, in V.

#### Mechanical Losses

**`Coulomb friction, Kc` — Friction**
`0` (default) | `scalar`

Coulomb friction, in N·m.

**`Viscous friction, Kb` — Friction**
`0` (default) | `scalar`

Viscous friction, in N·m/rad/s.

**`Windage, Kw` — Windage**
`0` (default) | `scalar`

Windage, in $N \cdot m/rad^2/s^2$.

## References

[1] Krause, P. C. *Analysis of Electric Machinery*. New York: McGraw-Hill, 1994.

## Extended Capabilities

**C/C++ Code Generation**
Generate C and C++ code using Simulink® Coder™.

## See Also

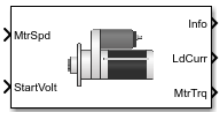
Starter

**Introduced in R2017a**

# Starter

Starter as a DC motor

**Library:** Powertrain Blockset / Energy Storage and Auxiliary Drive / Starter




## Description

The Starter block implements a starter assembly as a separately excited DC motor, permanent magnet DC motor, or series connection DC motor. The motor operates as a torque source to an internal combustion engine.

Use the Starter block:

- In an engine model with a front-end accessory drive (FEAD)
- To model engine start and stop scenarios

The Starter block supports only an angular speed input to the DC motor. A load torque input requires engine dynamics.

### Separately Excited DC Motor

In a separately excited DC motor, the field winding is connected to a separate source of DC power.

The relationship between the field winding voltage, field resistance, and field inductance is given by:

$$V_f = L_f\frac{di_f}{dt} + R_f i_f$$

The counter-electromotive force is a product of the field resistance, mutual inductance, and motor shaft angular speed:

$$EMF = L_a i_f L_{af}\omega$$

The armature voltage is given by:

$$V_a = L_a\frac{di_a}{dt} + R_a i_a + \quad EMF$$

The starter motor current load is the sum of the field winding current and armature winding current:

$$i_{load} = \quad i_f + \quad i_a$$

The starter motor shaft torque is the product of the armature current, field current, and mutual inductance:

$$T_{mech} = \quad i_a i_f L_{af}$$

### Permanent Magnet DC Motor

In a permanent magnet DC motor, the magnets establish the excitation flux, so there is no field current.

The counter-electromotive force is proportional to the motor shaft angular speed:

$$EMF = \quad K_t\omega$$

The armature voltage is given by:

$$V_a = L_a\frac{di_a}{dt} + R_a i_a + \quad EMF$$

The starter motor current load is equal to the armature winding current:

$$i_{load} = \quad i_a$$

The starter motor shaft torque is proportional to the armature winding current:

$$T_{mech} = \quad K_t i_a$$

### Series Excited DC Motor

A series excited DC motor connects the armature and field windings in series with a common DC power source.

The counter-electromotive force is a product of the field and armature initial series current, field, and armature mutual inductance and motor shaft angular speed:

$$EMF = i_{af}L_{af}\omega$$

The field and armature winding voltage is given by:

$$V_{af} = L_{ser}\frac{di_{af}}{dt} + R_{ser}i_{af} + \quad EMF$$

The starter motor current load is equal to the field and armature series current:

$$i_{load} = \quad i_{af}$$

The starter motor shaft torque is the product of the squared field and armature series current and the field and armature mutual inductance:

$$T_{mech} = \quad i_{af}^2 L_{af}$$

For motor stability, the motor shaft angular speed must be greater than the ratio of the series connected field and armature resistance to the mutual inductance:

$$\omega > \ -\frac{R_{ser}}{L_{af}}$$

### Power Accounting

For the power accounting, the block implements these equations.

| Bus Signal | | | Description | Variable | Equations |
|---|---|---|---|---|---|
| PwrInfo | PwrTrnsfrd — Power transferred between blocks<br>• Positive signals indicate flow into block<br>• Negative signals indicate flow out of block | PwrMtr | Mechanical power | $P_{mot}$ | $P_{mot} = -\omega T_{mech}$ |
| | | PwrBus | Electrical power | $P_{bus}$ | Separately excited DC motor<br>$P_{bus} = v_a i_a + v_f i_f$ |
| | | | | | PM excited DC motor<br>$P_{bus} = v_a i_a$ |
| | | | | | Series excited DC motor<br>$P_{bus} = v_{af} i_{af}$ |
| | PwrNotTrnsfrd — Power crossing the block boundary, but not transferred<br>• Positive signals indicate an input<br>• Negative signals indicate a loss | PwrLoss | Motor losses | $P_{loss}$ | $P_{loss} = -(P_{mot} + P_{bus} - P_{ind})$ |
| | PwrStored — Stored energy rate of change<br>• Positive signals indicate an increase<br>• Negative signals indicate a decrease | PwrInd | Electrical inductance | $P_{ind}$ | Separately excited DC motor<br>$P_{ind} = L_f i_f \frac{di_f}{dt} + L_a i_a \frac{di_a}{dt}$ |
| | | | | | PM excited DC motor<br>$P_{ind} = L_a i_a \frac{di_a}{dt}$ |
| | | | | | Series excited DC motor<br>$P_{ind} = L_{ser} i_{af} \frac{di_{af}}{dt}$ |

The equations use these variables.

| | |
|---|---|
| $R_a$ | Armature winding resistance |
| $L_a$ | Armature winding inductance |
| $EMF$ | Counter-electromotive force |
| $R_f$ | Field winding resistance |
| $L_f$ | Field winding inductance |
| $L_{af}$ | Field and armature mutual inductance |

| | |
|---|---|
| $i_a$ | Armature winding current |
| $i_f$ | Field winding current |
| $K_t$ | Motor torque constant |
| $\omega$ | Motor shaft angular speed |
| $V_a$ | Armature winding voltage |
| $V_f$ | Field winding voltage |
| $V_{af}$ | Field and armature winding voltage |
| $i_{af}$ | Field and armature series current |
| $R_{ser}$ | Series connected field and armature resistance |
| $L_{ser}$ | Series connected field and armature inductance |
| $i_{load}$ | Starter motor current load |
| $T_{mech}$ | Starter motor shaft torque |

## Ports

### Inputs

#### `MtrSpd` — Angular speed

`scalar`

Motor shaft angular speed, in rad/s.

#### `StartVolt` — Armature and field voltage

`scalar`

- Armature winding voltage $V_a$ and field winding voltage $V_f$, in V.
- In series excited DC motor, armature and field winding voltage $V_{af}$.

### Output

#### `Info` — Bus signal

bus

Bus signal containing these block calculations.

| Signal | | | Description | Units |
|---|---|---|---|---|
| `ArmCurr` | | | Armature winding current | A |
| `FldCurr` | | | Field winding current | A |
| `PwrInfo` | `PwrTrnsfrd` | `PwrMtr` | Mechanical power | W |
| | | `PwrBus` | Electrical power | W |
| | `PwrNotTrnsfrd` | `PwrLoss` | Motor power loss | W |
| | `PwrStored` | `PwrInd` | Electrical inductance | W |

#### `LdCurr` — Starter motor load current

`scalar`

Starter motor load current, in A.

**`MtrTrq` — Starter motor shaft torque**
`scalar`

Starter motor shaft torque, in N·m.

## Parameters

**Configuration**

**`Motor Type` — Select motor type**
`Separately Excited DC Motor` (default) | `Permanent Magnet Excited DC Motor` | `Series Connection DC Motor`

Select one of the three motor types.

**Dependencies**

The table summarizes the motor parameter dependencies.

| Motor Type | Enables Motor Parameter |
|---|---|
| `Separately Excited DC Motor` | **Armature winding resistance, Ra** |
| | **Armature winding inductance, La** |
| | **Field winding resistance Rf** |
| | **Field winding inductance, Lf** |
| | **Mutual inductance, Laf** |
| | **Initial armature and field current, Iaf** |
| `Permanent Magnet Excited DC Motor` | **Armature winding resistance, Rapm** |
| | **Armature winding inductance, Lapm** |
| | **Torque constant, Kt** |
| | **Initial armature current, Ia** |
| `Series Connection DC Motor` | **Total resistance, Rser** |
| | **Total inductance, Lser** |
| | **Initial current, Iafser** |
| | **Mutual inductance, Lafser** |

**Separately Excited DC Motor**

**`Armature winding resistance, Ra` — Resistance**
`1` (default) | `scalar`

Armature winding resistance, in ohm.

**Dependencies**

To enable this parameter, select `Separately Excited DC Motor` for the **Motor Type** parameter.

**`Armature winding inductance, La` — Inductance**
`.1` (default) | `scalar`

Armature winding inductance, in H.

**Dependencies**

To enable this parameter, select `Separately Excited DC Motor` for the **Motor Type** parameter.

**`Field winding resistance, Rf` — Resistance**
`.3` (default) | `scalar`

Field winding resistance, in ohm.

**Dependencies**

To enable this parameter, select `Separately Excited DC Motor` for the **Motor Type** parameter.

**`Field winding inductance, Lf` — Inductance**
`.2` (default) | `scalar`

Field winding inductance, in H.

**Dependencies**

To enable this parameter, select `Separately Excited DC Motor` for the **Motor Type** parameter.

**`Mutual inductance, Laf` — Inductance**
`.3` (default) | `scalar`

Mutual inductance, in H.

**Dependencies**

To enable this parameter, select `Separately Excited DC Motor` for the **Motor Type** parameter.

**`Initial armature current and field current, Iaf` — Current**
`[0 0]` (default) | `vector`

Initial armature and field current, in A.

**Dependencies**

To enable this parameter, select `Separately Excited DC Motor` for the **Motor Type** parameter.

**Permanent Magnet Excited DC Motor**

**`Armature winding resistance, Rapm` — Resistance**
`.5` (default) | `scalar`

Armature winding resistance, in ohm.

**Dependencies**

To enable this parameter, select `Permanent Magnet Excited DC Motor` for the **Motor Type** parameter.

**`Armature winding inductance, Lapm` — Inductance**
`.1` (default) | `scalar`

Armature winding inductance, in H.

#### Dependencies

To enable this parameter, select `Permanent Magnet Excited DC Motor` for the **Motor Type** parameter.

### `Torque constant, Kt` — Motor torque constant

`.1` (default) | `scalar`

Motor torque constant, in N·m/A.

#### Dependencies

To enable this parameter, select `Permanent Magnet Excited DC Motor` for the **Motor Type** parameter.

### `Initial armature current, Ia` — Current

`.1` (default) | `scalar`

Initial armature current, in A.

#### Dependencies

To enable this parameter, select `Permanent Magnet Excited DC Motor` for the **Motor Type** parameter.

#### Series Connection DC Motor

### `Total resistance, Rser` — Resistance

`.1` (default) | `scalar`

Series connected field and armature resistance, in ohm.

#### Dependencies

To enable this parameter, select `Series Excited DC Motor` for the **Motor Type** parameter.

### `Total inductance, Lser` — Inductance

`.1` (default) | `scalar`

Series connected field and armature inductance, in H.

#### Dependencies

To enable this parameter, select `Series Excited DC Motor` for the **Motor Type** parameter.

### `Initial current, Iafser` — Current

`0` (default) | `scalar`

Initial series current, in A.

#### Dependencies

To enable this parameter, select `Series Excited DC Motor` for the **Motor Type** parameter.

### `Mutual inductance, Lafser` — Inductance

`.3` (default) | `scalar`

Field and armature mutual inductance, in H.

#### Dependencies

To enable this parameter, select `Series Excited DC Motor` for the **Motor Type** parameter.

## References

[1] Krause, P. C. *Analysis of Electric Machinery*. New York: McGraw-Hill, 1994.

## Extended Capabilities

#### C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

## See Also

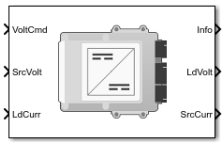
Reduced Lundell Alternator

#### Introduced in R2017a

# Bidirectional DC-DC

DC-to-DC converter that supports bidirectional boost and buck

**Library:** Powertrain Blockset / Energy Storage and Auxiliary Drive / DC-DC




## Description

The Bidirectional DC-DC block implements a DC-to-DC converter that supports bidirectional boost and buck (lower) operation. Unless the DC-to-DC conversion limits the power, the output voltage tracks the voltage command. You can specify electrical losses or measured efficiency.

Depending on your battery system configuration, the voltage might not be at a potential that is required by electrical system components such has inverters and motors. You can use the block to boost or buck the voltage. Connect the block to the battery and one of these blocks:

- Mapped Motor
- IM Controller
- Interior PM Controller
- Surface Mount PM Controller

To calculate the electrical loss during the DC-to-DC conversion, use **Parameterize losses by**.

| Parameter Option | Description |
|---|---|
| `Single efficiency measurement` | Electrical loss calculated using a constant value for conversion efficiency. |
| `Tabulated loss data` | Electrical loss calculated as a function of load current and voltage. DC-to-DC converter data sheets typically provide loss data in this format. When you use this option, provide data for all the operating quadrants in which the simulation will run. If you provide partial data, the block assumes the same loss pattern for other quadrants. The block does not extrapolate loss that is outside the range voltage and current that you provide. The block allows you to account for fixed losses that are still present for zero voltage or current. |

| Parameter Option | Description |
|---|---|
| `Tabulated efficiency data` | Electrical loss calculated using conversion efficiency that is a function of load current and voltage. When you use this option, provide data for all the operating quadrants in which the simulation will run. If you provide partial data, the block assumes the same efficiency pattern for other quadrants. The block:<br>• Assumes zero loss when either the voltage or current is zero.<br>• Uses linear interpolation to determine the loss. At lower power conditions, for calculation accuracy, provide efficiency at low voltage and low current. |

**Note** The block does not support inversion. The polarity of the input voltage matches the polarity of the output voltage.

### Theory

The Bidirectional DC-DC block uses the commanded voltage and the actual voltage to determine whether to boost or buck (lower) the voltage. You can specify a time constant for the voltage response.

| If | Then |
|---|---|
| $Volt_{cmd} > Src_{Volt}$ | Boost |
| $Volt_{cmd} < Src_{Volt}$ | Buck |

The Bidirectional DC-DC block uses a time constant-based regulator to provide a fixed output voltage that is independent of load current. Using the output voltage and current, the block determines the losses of the DC-to-DC conversion. The block uses the conversion losses to calculate the input current. The block accounts for:

- Bidirectional current flow
  - Source to load — Battery discharge
  - Load to source — Battery charge
- Rated power limits

The block provides voltage control that is power limited based on these equations. The voltage is fixed. The block does not implement a voltage drop because the load current approximates DC-to-DC conversion with a bandwidth that is greater than the load current draw.

| | |
|---|---|
| DC-to-DC converter load voltage | $LdVolt_{Cmd} = \min(Volt_{Cmd}, \frac{P_{limit}}{Ld_{Amp}}, 0)$<br>$LdVolt = LdVolt_{Cmd} \cdot \frac{1}{\tau s + 1}$ |
| Power loss for single efficiency source to load | $Pwr_{Loss} = \frac{100 - Eff}{Eff} \cdot Ld_{Volt} \cdot Ld_{Amp}$ |

| | |
|---|---|
| Power loss for single efficiency load to source | $Pwr_{Loss} = \frac{100 - Eff}{Eff} \cdot \left\lvert Ld_{Volt} \cdot Ld_{Amp} \right\rvert$ |
| Power loss for tabulated efficiency | $Prw_{Loss} = f\left(Ld_{Volt}, Ld_{Amp}\right)$ |
| Source current draw from DC-to-DC converter | $Src_{Amp} = \frac{Ld_{Pwr} + Prw_{Loss}}{Src_{Volt}}$ |
| Source power from DC-to-DC converter | $Src_{Pwr} = Src_{Amp} \cdot Src_{Volt}$ |

### Power Accounting

For the power accounting, the block implements these equations.

| Bus Signal | | | Description | Variable | Equations |
|---|---|---|---|---|---|
| `PwrInfo` | `PwrTrnsfrd` — Power transferred between blocks<br>• Positive signals indicate flow into block<br>• Negative signals indicate flow out of block | `PwrBusSrc` | Source power to DC-to-DC converter | $P_{src}$ | $P_{src} = SrcPwr$ |
| | | `PwrBusLd` | Load power from DC-to-DC converter | $P_{bus}$ | $P_{bus} = -LdVolt$ |
| | `PwrNotTrnsfrd` — Power crossing the block boundary, but not transferred<br>• Positive signals indicate an input<br>• Negative signals indicate a loss | `PwrLoss` | Converter power loss | $P_{loss}$ | $P_{loss} = PwrLoss$ |
| | `PwrStored` — Stored energy rate of change<br>• Positive signals indicate an increase<br>• Negative signals indicate a decrease | | *Not used* | | |

The equations use these variables.

| | |
|---|---|
| $Volt_{Cmd}$ | DC-to-DC converter commanded output voltage |
| $Src_{Volt}$ | Source input voltage to DC-to-DC converter |
| $Ld_{Amp}$ | Load current of DC-to-DC converter |
| $Ld_{Volt}$ | Load voltage of DC-to-DC converter |
| $Src_{Amp}$ | Source current draw from DC-to-DC converter |
| $\tau$ | Conversion time constant |
| $V_{init}$ | Initial load voltage of the DC-to-DC converter |
| $P_{limit}$ | Output power limit for DC-to-DC converter |
| $Eff$ | Input to output efficiency |
| $Src_{Pwr}$ | Source power to DC-to-DC converter |
| $Ld_{Pwr}$ | Load power from DC-to-DC converter |

| | |
|---|---|
| $Pwr_{Loss}$ | Power loss |
| $LdVolt_{Cmd}$ | Commanded load voltage of DC-to-DC converter before application of time constant |

## Ports

### Inputs

#### `VoltCmd` — Commanded voltage

`scalar`

DC-to-DC converter commanded output voltage, $Volt_{Cmd}$, in V.

#### `SrcVolt` — Input voltage

`scalar`

Source input voltage to DC-to-DC converter, $Src_{Volt}$, in V.

#### `LdCurr` — Load current

`scalar`

Load current of DC-to-DC converter, $Ld_{Amp}$, in A.

### Output

#### `Info` — Bus signal

bus

Bus signal containing these block calculations.

| Signal | | | Description | Variable | Units |
|---|---|---|---|---|---|
| `SrcPwr` | | | Source power to DC-to-DC converter | $Src_{Pwr}$ | W |
| `LdPwr` | | | Load power from DC-to-DC converter | $Ld_{Pwr}$ | W |
| `PwrLoss` | | | Power loss | $Pwr_{Loss}$ | W |
| `LdVoltCmd` | | | Commanded load voltage of DC-to-DC converter before application of time constant | $LdVolt_{Cmd}$ | V |
| `PwrInfo` | `PwrTrnsfrd` | `PwrBusSrc` | Source power to DC-to-DC converter | $P_{src}$ | W |
| | | `PwrBusLd` | Load power from DC-to-DC converter | $P_{bus}$ | W |
| | `PwrNotTrnsfrd` | `PwrLoss` | Converter power loss | $P_{loss}$ | W |
| | `PwrStored` | *Not used* | | | |

#### `LdVolt` — Load voltage

`scalar`

Load voltage of DC-to-DC converter, $Ld_{Volt}$, in V.

**`SrcCurr` — Source current**
`scalar`

Source current draw from DC-to-DC converter, $Src_{Amp}$, in A.

## Parameters

#### Electrical Control

**`Converter response time constant` — Constant**
`1/1000` (default) | `scalar`

Converter response time, $\tau$, in s.

**`Converter response initial voltage, Vinit` — Voltage**
`0` (default) | `scalar`

Initial load voltage of the DC-to-DC converter, $V_{init}$, in V.

**`Converter power limit, Plimit` — Power**
`100000` (default) | `scalar`

Initial load voltage of the DC-to-DC converter, $P_{limit}$, in W.

#### Electrical Losses

**`Parameterize losses by` — Loss calculation**
`Single efficiency measurement` (default) | `Tabulated loss data` | `Tabulated efficiency data`

This table summarizes the loss options used to calculate electrical options.

| Parameter Option | Description |
|---|---|
| `Single efficiency measurement` | Electrical loss calculated using a constant value for conversion efficiency. |
| `Tabulated loss data` | Electrical loss calculated as a function of load current and voltage. DC-to-DC converter data sheets typically provide loss data in this format. When you use this option, provide data for all the operating quadrants in which the simulation will run. If you provide partial data, the block assumes the same loss pattern for other quadrants. The block does not extrapolate loss that is outside the range voltage and current that you provide. The block allows you to account for fixed losses that are still present for zero voltage or current. |

| Parameter Option | Description |
|---|---|
| `Tabulated efficiency data` | Electrical loss calculated using conversion efficiency that is a function of load current and voltage. When you use this option, provide data for all the operating quadrants in which the simulation will run. If you provide partial data, the block assumes the same efficiency pattern for other quadrants. The block:<br>• Assumes zero loss when either the voltage or current is zero.<br>• Uses linear interpolation to determine the loss. At lower power conditions, for calculation accuracy, provide efficiency at low voltage and low current. |

**`Overall DC to DC converter efficiency, eff` — Constant**
98 (default) | `scalar`

Overall conversion efficiency, *Eff*, in %.

**Dependencies**

To enable this parameter, for **Parameterize losses by**, select `Single efficiency measurement`.

**`Vector of voltages (v) for tabulated loss, v_loss_bp` — Breakpoints**
`[0 200 400 600 800 1000]` (default) | 1-by-`M` vector

Tabulated loss breakpoints for `M` load voltages, in V.

**Dependencies**

To enable this parameter, for **Parameterize losses by**, select `Tabulated loss data`.

**`Vector of currents (i) for tabulated loss, i_loss_bp` — Breakpoints**
`[0 25 50 75 100]` (default) | 1-by-`N` vector

Tabulated loss breakpoints for `N` load currents, in A.

**Dependencies**

To enable this parameter, for **Parameterize losses by**, select `Tabulated loss data`.

**`Corresponding losses, losses_table` — 2-D lookup table**
`N`-by-`M` matrix

Electrical loss map, as a function of `N` load currents and `M` load voltages, in W.

**Dependencies**

To enable this parameter, for **Parameterize losses by**, select `Tabulated loss data`.

**`Vector of voltages (v) for tabulated efficiency, v_eff_bp` — Breakpoints**
`[200 400 600 800 1000]` (default) | 1-by-`M` vector

Tabulated efficiency breakpoints for `M` load voltages, in V.

**Dependencies**

To enable this parameter, for **Parameterize losses by**, select `Tabulated efficiency data`.

**`Vector of currents (i) for tabulated efficiency, i_eff_bp` — Breakpoints**
`[25 50 75 100]` (default) | 1-by-N vector

Tabulated efficiency breakpoints for `N` load currents, in A.

**Dependencies**

To enable this parameter, for **Parameterize losses by**, select `Tabulated efficiency data`.

**`Corresponding efficiency, efficiency_table` — 2-D lookup table**
N-by-M matrix

Electrical efficiency map, as a function of `N` load currents and `M` load voltages, in %.

**Dependencies**

To enable this parameter, for **Parameterize losses by**, select `Tabulated efficiency data`.

## Extended Capabilities

**C/C++ Code Generation**
Generate C and C++ code using Simulink® Coder™.

## See Also

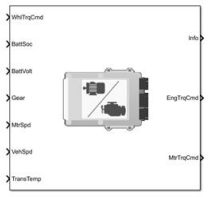
Equivalent Circuit Battery | Estimation Equivalent Circuit Battery

**Topics**
Battery Modeling

**Introduced in R2017b**

# Equivalent Consumption Minimization Strategy

Energy management controller for P0–P4 hybrid electric vehicles
**Library:** Powertrain Blockset / Propulsion / Supervisory Controllers

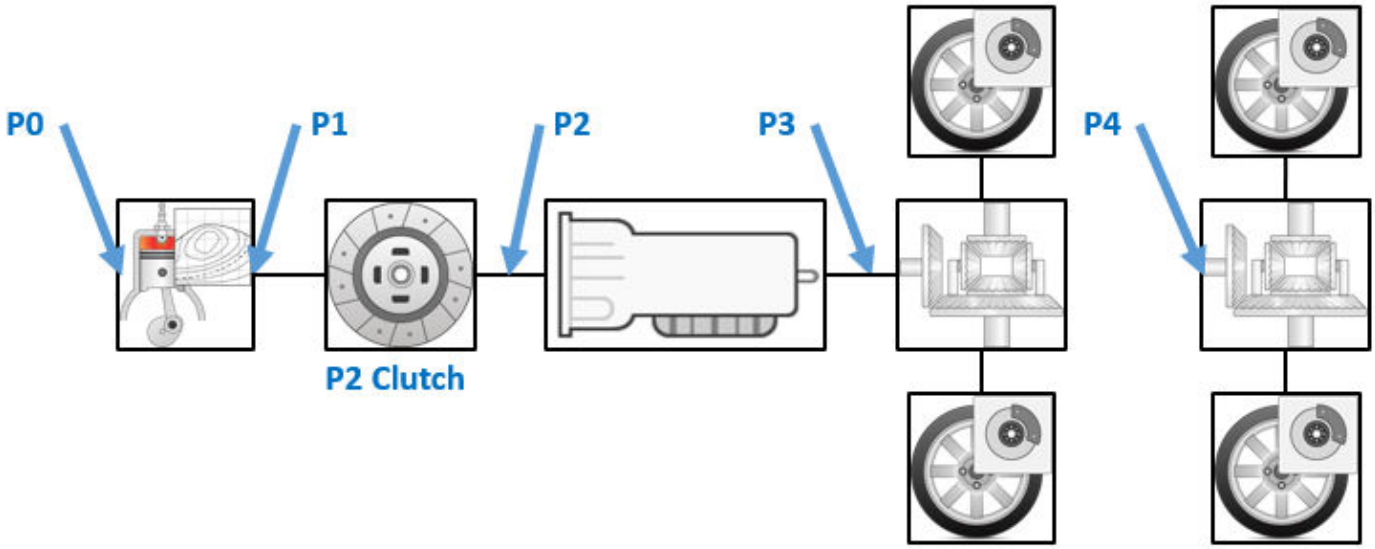


## Description

Use the Equivalent Consumption Minimization Strategy (ECMS) block to control the energy management of hybrid electric vehicles (HEVs). The block optimizes the torque split between the engine and motor to minimize energy consumption while maintaining the battery state of charge (SOC).

The HEV P0, P1, P2, P3, and P4 reference applications use the Equivalent Consumption Minimization Strategy block for hybrid control.

Use the **Motor location** parameter to specify the HEV motor location.




Use the **ECMS method** parameter to implement either an adaptive or non-adaptive ECMS method. The HEV architectures are charge-sustaining, meaning the battery SOC must remain in a specified range because there is no plugin capability to recharge the battery. The battery is an energy buffer, and all energy comes from the fuel if the change in SOC is minimized over a drive cycle. To sustain the charge over a specified drive cycle, the block implements either of these ECMS methods.

| ECMS Method | Description |
| --- | --- |
| `Non-adaptive` (default) | The block uses a constant ECMS equivalence factor.<br>• Use this method to determine the best fuel economy over a drive cycle.<br>  • If you change the drive cycle or HEV architecture, retune the **ECMS weighting factor** to maintain the ending SOC.<br>• By default, the block uses a single constant. |
| `Adaptive` | The block adjusts an ECMS equivalence factor by using the output of a PI controller.<br>• Use this method to maintain the SOC and minimize the delta SOC over many drive cycles. The block:<br>  • Tunes the PI controller gains.<br>  • Sustains the SOC.<br>• The PI controller minimizes the error between the target SOC and current SOC. |

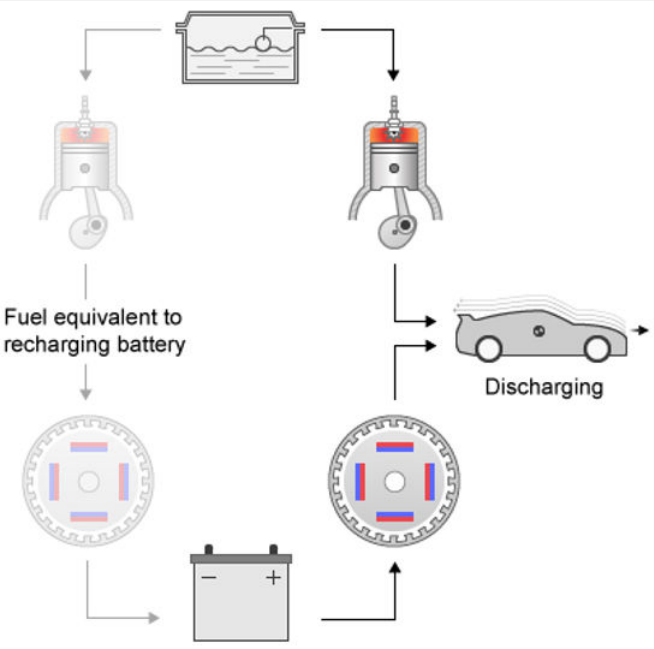
### ECMS Control Algorithm

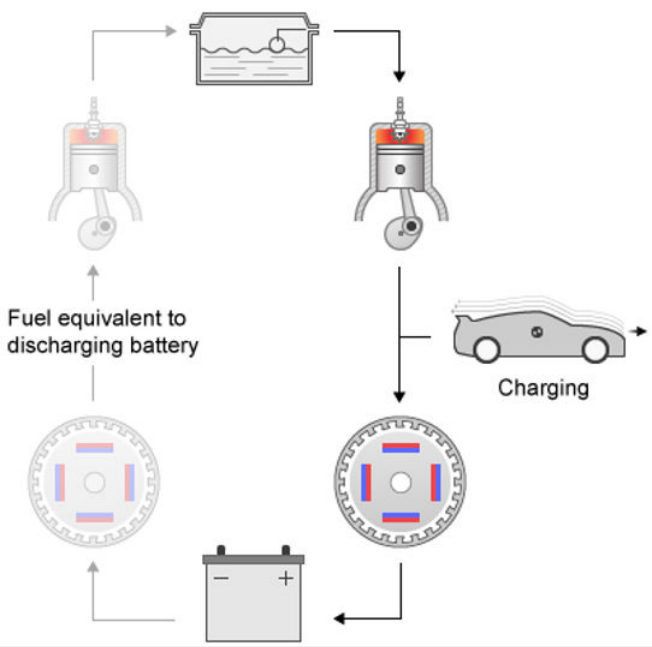
The block implements a dynamic supervisory controller that determines the engine torque, motor torque, starter, clutch, and brake pressure commands. Specifically, the block:

- Converts the driver accelerator pedal signal to a wheel torque request. To calculate the total powertrain torque at the wheels, the algorithm uses the maximum engine torque and motor torque curves and the transmission and differential gear ratios.
- Converts the driver brake pedal signal to a brake pressure request. The algorithm multiplies the brake pedal signal by a maximum brake pressure.
- Implements a regenerative braking algorithm for the traction motor to recover the maximum amount of kinetic energy from the vehicle.

  The block implements an ECMS algorithm[2] that optimizes the torque split between the engine and motor to minimize energy consumption while maintaining the battery SOC. Specifically, the ECMS:

  - Assigns a cost to electrical energy, so that using stored electrical energy is equal to consuming fuel energy.

| Battery Mode | Equivalent Electrical Energy | Description |
|---|---|---|
| Discharging | Positive | Battery discharges stored electrical energy when the electric machine is in use.<br>  |

| Battery Mode | Equivalent Electrical Energy | Description |
|---|---|---|
| Charging | Negative | Battery stores electrical energy from either the:<br>• Engine and electric machine acting as a generator<br>• Electric machine acting as a generator during regenerative braking<br>  |

- Is an instantaneous minimization method that the software solves at every controller time step. To implement the strategy, the ECMS selects the optimal motor and engine torque in the optimization strategy to minimize the equivalent energy consumption.
- Implements either an adaptive or non-adaptive ECMS method.

## Ports

### Input

#### `WhlTrqCmd` — Wheel torque command

`scalar`

Wheel torque command.

Data Types: `double`

#### `BattSoc` — Battery state of charge

`scalar`

Battery state of charge.

Data Types: `double`

**`BattVolt` — Battery voltage**
`scalar`

Battery voltage.

Data Types: `double`

**`Gear` — Transmission gear**
`scalar`

Transmission gear.

Data Types: `double`

**`MtrSpd` — Motor speed**
`scalar`

Motor speed.

Data Types: `double`

**`VehSpd` — Vehicle speed**
`scalar`

Vehicle speed, in m/s.

Data Types: `double`

**`TransTemp` — Transmission temperature**
`scalar`

Transmission temperature, in K.

Data Types: `double`

**Output**

**`Info` — Block data**
bus

Block data, returned as a bus signal that contains these block values.

| Signal | Description | Units |
|---|---|---|
| `EngTrqCmd` | Engine torque command | N·m |
| `MtrTrqCmd` | Motor torque command | N·m |
| `EquivFctr` | Equivalence factor | NA |
| `MinHamil` | Minimum Hamiltonian | kW |

**`EngTrqCmd` — Engine torque command**
`scalar`

Engine torque command, in N·m.

Data Types: `double`

**`MtrTrqCmd` — Motor torque command**
`scalar`

Motor torque command, in N·m.

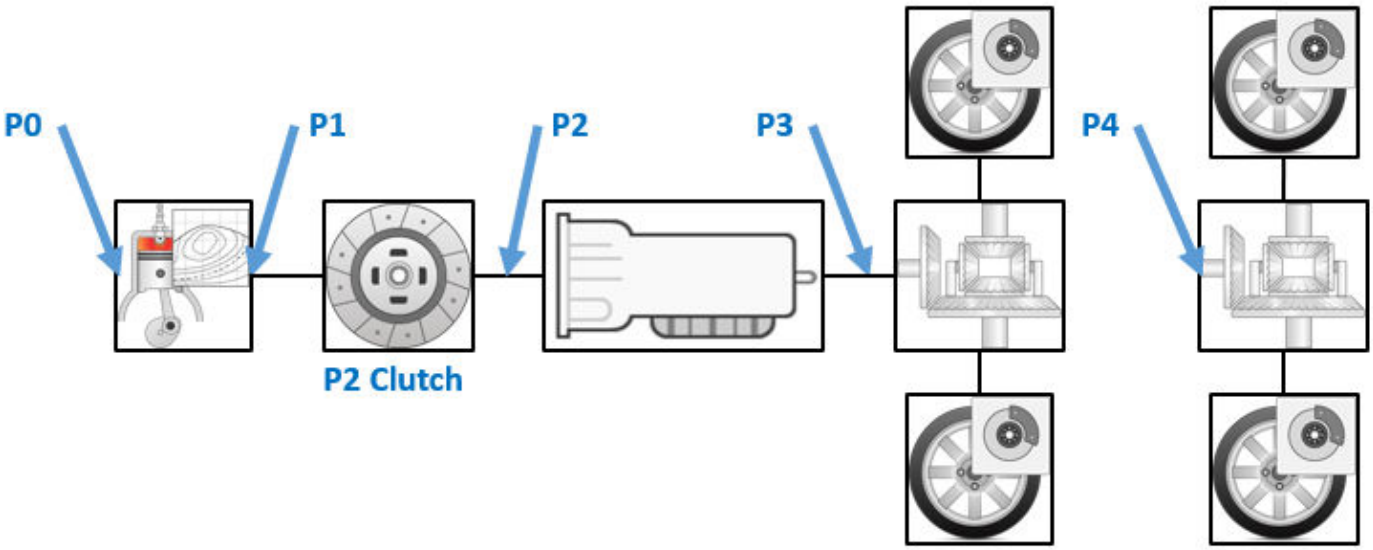
Data Types: `double`

## Parameters

### Block Options

**`Motor location` — Location of motor**
`P0` (default) | `P1` | `P2` | `P3` | `P4`

Specify the HEV motor location.




**`ECMS method` — ECMS method**
`Non-adaptive` (default) | `Adaptive`

Use the **ECMS method** parameter to implement either an adaptive or non-adaptive ECMS method. The HEV architectures are charge-sustaining, meaning the battery SOC must remain in a specified range because there is no plugin capability to recharge the battery. The battery is an energy buffer, and all energy comes from the fuel if the change in SOC is minimized over a drive cycle. To sustain the charge over a specified drive cycle, the block implements either of these ECMS methods.

| ECMS Method | Description |
|---|---|
| `Non-adaptive` (default) | The block uses a constant ECMS equivalence factor.<br>• Use this method to determine the best fuel economy over a drive cycle.<br>  • If you change the drive cycle or HEV architecture, retune the **ECMS weighting factor** to maintain the ending SOC.<br>• By default, the block uses a single constant. |

| ECMS Method | Description |
|---|---|
| `Adaptive` | The block adjusts an ECMS equivalence factor by using the output of a PI controller.<br>• Use this method to maintain the SOC and minimize the delta SOC over many drive cycles. The block:<br>  • Tunes the PI controller gains.<br>  • Sustains the SOC.<br>• The PI controller minimizes the error between the target SOC and current SOC. |

#### Differential

**`Differential gear ratio, N_diff` — Differential gear ratio**
`3.32` (default) | `scalar`

Differential gear ratio. No dimension.

Data Types: `double`

**`Differential efficiency factor, eta_diff` — Differential efficiency factor**
`0.98` (default) | `scalar`

Differential efficiency factor. No dimension.

Data Types: `double`

**`Loaded wheel radius, Re` — Loaded wheel radius**
`0.327` (default) | `scalar`

Loaded wheel radius, in m.

Data Types: `double`

#### Transmission

**`Transmission efficiency factors` — Transmission efficiency factors**
`Gear, input torque, input speed, and temperature` (default) | `Gear only`

Transmission efficiency factors.

Data Types: `double`

**`Transmission gear number vector, G_trans` — Transmission gear number vector**
`[0 1 2 3 4 5 6]` (default) | `vector`

Transmission gear number vector. No dimension.

Data Types: `double`

**`Transmission gear ratio vector, N_trans` — Transmission gear ratio vector**
`[1 4.212 2.637 1.8 1.386 1 0.772]` (default) | `vector`

Transmission gear ratio vector. No dimension.

Data Types: `double`

**`Transmission efficiency vector, eta_trans` — Transmission efficiency vector**
`[1 1 1 1 1 1 1]` (default) | `vector`

Transmission efficiency vector. No dimension.

**Dependencies**

To enable this parameter, set **Transmission efficiency factors** to `Gear only`.

Data Types: `double`

**`Transmission efficiency torque breakpoints, Trq_trans_bpts` — Transmission efficiency torque breakpoints**
`[25 50 75 100 150 200 250]` (default) | `vector`

Transmission efficiency torque breakpoints, in N·m.

**Dependencies**

To enable this parameter, set **Transmission efficiency factors** to `Gear, input torque, input speed, and temperature`.

Data Types: `double`

**`Transmission efficiency speed breakpoints, omega_trans_bpts` — Transmission efficiency speed breakpoints**
`[500.383141080919 749.619781962827 1002.676141478941 1250.957852702297 1499.239563925654 1747.521275149011 1995.802986372368 2501.915705404595 2998.479127851308 4001.155269330249 5003.83141080919]` (default) | `vector`

Transmission efficiency speed breakpoints, in rad/s.

**Dependencies**

To enable this parameter, set **Transmission efficiency factors** to `Gear, input torque, input speed, and temperature`.

Data Types: `double`

**`Transmission efficiency temperature breakpoints, Temp_trans_bpts` — Transmission efficiency temperature breakpoints**
`[313 358]` (default) | `vector`

Transmission efficiency temperature breakpoints, in K.

**Dependencies**

To enable this parameter, set **Transmission efficiency factors** to `Gear, input torque, input speed, and temperature`.

Data Types: `double`

**`Transmission efficiency vector, eta_trans_tbl` — Transmission efficiency vector**
`array`

Transmission efficiency vector. No dimension.

#### Dependencies

To enable this parameter, set **Transmission efficiency factors** to `Gear, input torque, input speed, and temperature`.

Data Types: `double`

#### Engine

**`Speed breakpoints, f_tbrake_n_bpt` — Speed breakpoints**
`[0 750 1053.57142857143 1357.14285714286 1660.71428571429 1964.28571428571 2267.85714285714 2571.42857142857 2875 3178.57142857143 3482.14285714286 3785.71428571429 4089.28571428571 4392.85714285714 4696.42857142857 5000]` (default) | `vector`

Speed breakpoints, in rpm.

Data Types: `double`

**`Commanded torque breakpoints, f_tbrake_t_bpt` — Commanded torque breakpoints**
`[0 15 26.4285714285714 37.8571428571429 49.2857142857143 60.7142857142857 72.1428571428571 83.5714285714286 95 106.428571428571 117.857142857143 129.285714285714 140.714285714286 152.142857142857 163.571428571429 175]` (default) | `vector`

Commanded torque breakpoints, in N·m.

Data Types: `double`

**`Brake torque map, f_tbrake` — Brake torque map**
`array`

Brake torque map, in N·m.

Data Types: `double`

**`Minimum engine torque command table, f_tbrake_min` — Minimum engine torque command table**
`vector`

Minimum engine torque command table, in N·m.

Data Types: `double`

**`Fuel flow map, f_fuel` — Fuel flow map**
`array`

Fuel flow map, in kg/s.

Data Types: `double`

**`Minimum engine torque command, HEVEngTrq_min` — Minimum engine torque command**
`16.18610438796213` (default) | `scalar`

Minimum engine torque command, in N·m.

Data Types: `double`

**`Fuel lower heating value, LHV` — Fuel lower heating value**
`46000000` (default) | `scalar`

Fuel lower heating value, in J/kg.

Data Types: `double`

**`Accessory power, AccPwrTbl` — Accessory power**
`[0 0]` (default) | `vector`

Accessory power. No dimension.

Data Types: `double`

**`Accessory power speed breakpoints, AccSpdBpts` — Accessory power speed breakpoints**
`[300 750]` (default) | `vector`

Accessory power speed breakpoints. No dimension.

Data Types: `double`

**`Engine idle speed, N_idle` — Engine idle speed**
`750` (default) | `scalar`

Engine idle speed, in rpm.

Data Types: `double`

#### Battery

**`Battery state-of-charge breakpoints, SOC_bpt` — Battery state-of-charge breakpoints**
`[0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1]` (default) | `vector`

Battery state-of-charge breakpoints. No dimension.

Data Types: `double`

**`Battery charge limit table, ChrgLmt` — Battery charge limit table**
`[1 1 1 1 1 1 1 0.9 0.7 0.5 0]` (default) | `vector`

Battery charge limit table. No dimension.

Data Types: `double`

**`Battery discharge limit table, DischrgLmt` — Battery discharge limit table**
`[0 0.5 0.7 0.9 1 1 1 1 1 1 1]` (default) | `vector`

Battery discharge limit table. No dimension.

Data Types: `double`

**`Maximum battery current, BattCurrMax` — Maximum battery current**
`150` (default) | `scalar`

Maximum battery current, in A.

Data Types: `double`

**`DC/DC converter efficiency, eta_dcdc` — DC/DC converter efficiency**
`1` (default) | `scalar`

DC/DC converter efficiency. No dimension.

Data Types: `double`

**`Maximum battery charge power, BattChrgPwrMax` — Maximum battery charge power**
`-30000` (default) | `scalar`

Maximum battery charge power, in W.

Data Types: `double`

**`Maximum battery discharge power, BattDischrgPwrMax` — Maximum battery discharge power**
`46000` (default) | `scalar`

Maximum battery discharge power, in W.

Data Types: `double`

**Motor**

**`Motor maximum torque table, f_tmtr_max` — Motor maximum torque table**
`vector`

Motor maximum torque table, in N·m.

Data Types: `double`

**`Motor speed breakpoints, f_mtr_w_bpt` — Motor speed breakpoints**
`vector`

Motor speed breakpoints, in rpm.

Data Types: `double`

**`Motor torque breakpoints, f_mtr_t_bpt` — Motor torque breakpoints**
`vector`

Motor torque breakpoints, in N·m.

Data Types: `double`

**`Motor efficiency map, f_mtr_eta` — Motor efficiency map**
`array`

Motor efficiency map. No dimension.

Data Types: `double`

**`Number of motor torque calculation points, Ngrid` — Number of motor torque calculation points**
`200` (default) | `scalar`

Number of motor torque calculation points. No dimension.

Data Types: `double`

**`P0 belt ratio, N_P0` — P0 belt ratio**
`3` (default) | `scalar`

P0 belt ratio. No dimension.

**Dependencies**

To enable this parameter, set **Motor location** to `P0`.

Data Types: `double`

**Energy Management**

**`ECMS weighting factor, ECMS_s` — ECMS weighting factor**
`3.385` (default) | `scalar`

ECMS weighting factor. No dimension.

Data Types: `double`

**`Penalty factor power, PenaltyFctrPwr` — Penalty factor power**
`3` (default) | `scalar`

Penalty factor power. No dimension.

Data Types: `double`

**`Adaptive ECMS proportional gain, ECMS_Kp` — Adaptive ECMS proportional gain**
`0` (default) | `scalar`

Adaptive ECMS proportional gain. No dimension.

**Dependencies**

To enable this parameter, set **ECMS method** to `Adaptive`.

Data Types: `double`

**`Adaptive ECMS integral gain, ECMS_Ki` — Adaptive ECMS integral gain**
`0` (default) | `scalar`

Adaptive ECMS integral gain. No dimension.

**Dependencies**

To enable this parameter, set **ECMS method** to `Adaptive`.

Data Types: `double`

**`Constraint penalty factor, PenaltyFctr` — Constraint penalty factor**
`10000000` (default) | `scalar`

Constraint penalty factor. No dimension.

Data Types: `double`

**`Target battery state-of-charge, SOCTrgt` — Target battery state-of-charge**
`60` (default) | `scalar`

Target battery state-of-charge. No dimension.

Data Types: `double`

**`Minimum battery state-of-charge, SOCmin` — Minimum battery state-of-charge**
`40` (default) | `scalar`

Minimum battery state-of-charge. No dimension.

Data Types: `double`

**`Maximum battery state-of-charge, SOCmax` — Maximum battery state-of-charge**
`80` (default) | `scalar`

Maximum battery state-of-charge. No dimension.

Data Types: `double`

**Acknowledgments**

MathWorks® would like to acknowledge the contribution of Dr. Simona Onori to the ECMS optimal control algorithm implemented in this block. Dr. Onori is a Professor of Energy Resources Engineering at Stanford University. Her research interests include electrochemical modeling, estimation and optimization of energy storage devices for automotive and grid-level applications, hybrid and electric vehicles modeling and control, PDE modeling, and model-order reduction and estimation of emission mitigation systems. She is a senior member of IEEE.

## References

[1] Balazs, A., Morra, E., and Pischinger, S., *Optimization of Electrified Powertrains for City Cars*. SAE Technical Paper 2011-01-2451. Warrendale, PA: SAE International Journal of Alternative Powertrains, 2012.

[2] Onori, S., Serrao, L., and Rizzoni, G., *Hybrid Electric Vehicles Energy Management Systems*. New York: Springer, 2016.

## Extended Capabilities

**C/C++ Code Generation**
Generate C and C++ code using Simulink® Coder™.

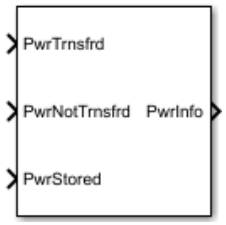
## See Also

**Topics**
“Hybrid and Electric Vehicle Reference Application Projects”

**Introduced in R2020b**

# Power Accounting Bus Creator

Create power information bus
**Library:** Powertrain Blockset / Utilities / Power Accounting




## Description

Creates a power information bus for reporting system power and energy consumption. You can associate the block to a parent system, select types of power signals to track, and add signal descriptions. If you want to generate a power and energy report, you must use this block to log the power signals in your plant model blocks. The Powertrain Blockset plant blocks use the Power Accounting Bus Creator to log the power signals. The documentation for each block includes information about the logged power bus signals.

The system-level power and energy accounting satisfies the conservation of energy.

$$\sum P_{trans} + \sum P_{nottrans} = \sum P_{store}$$

To add the Power Accounting Bus Creator to your plant block, follow these steps:

1. Add the Power Accounting Bus Creator block to your block.
2. Select the types of power signals that you want to log. See “Power Signals” on page 3-53.
3. Associate the Power Accounting Bus Creator with a parent subsystem. See “Block Association” on page 3-54.
4. Connect the power signals to the Power Accounting Bus Creator.
   - Follow the sign convention.
   - To ensure that your plant block conserves energy, include all power associated with the block.
5. In the Power Accounting Bus Creator:
   - On the **Transferred** power tab, specify these parameters:
     - **Associated Port**
     - **Description**
   - On the **Not Transferred** power tab, specify the **Description** parameter:
6. In the plant block, connect the transferred power signals to the Power Accounting Bus Creator ports that are specified with the **Associated Port** parameter.

### Power Signals

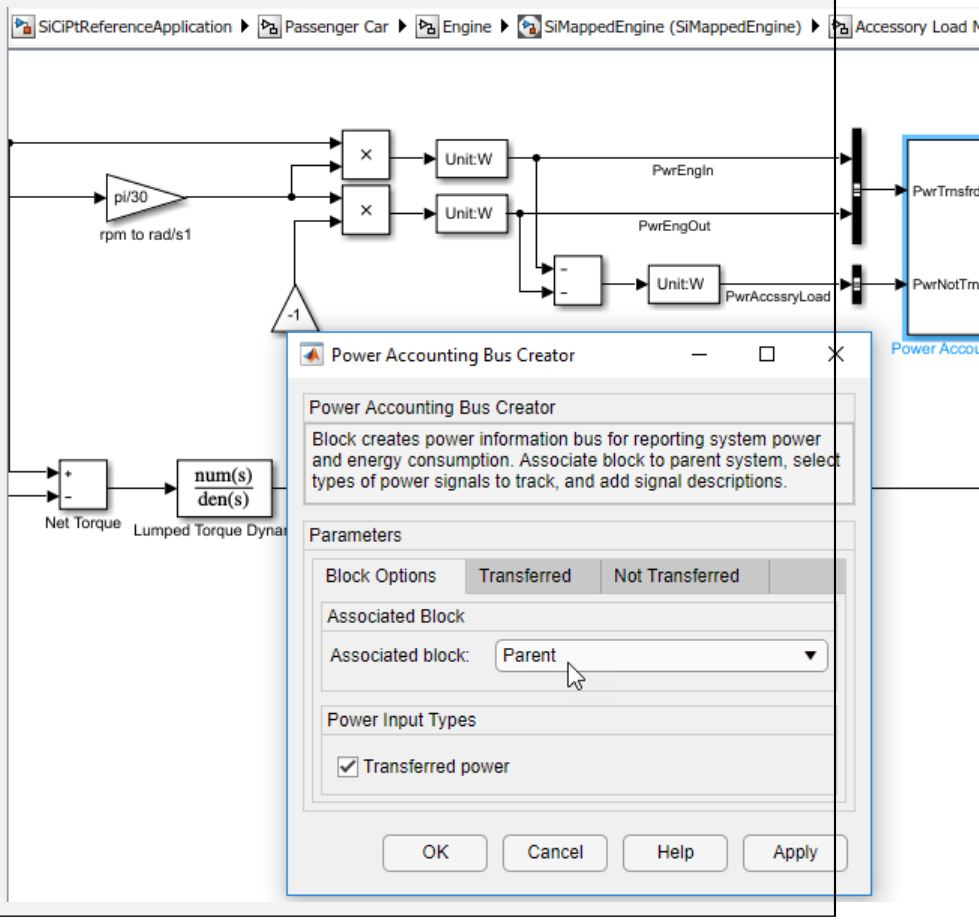
The Power Accounting Bus Creator sorts the signals into three power types.

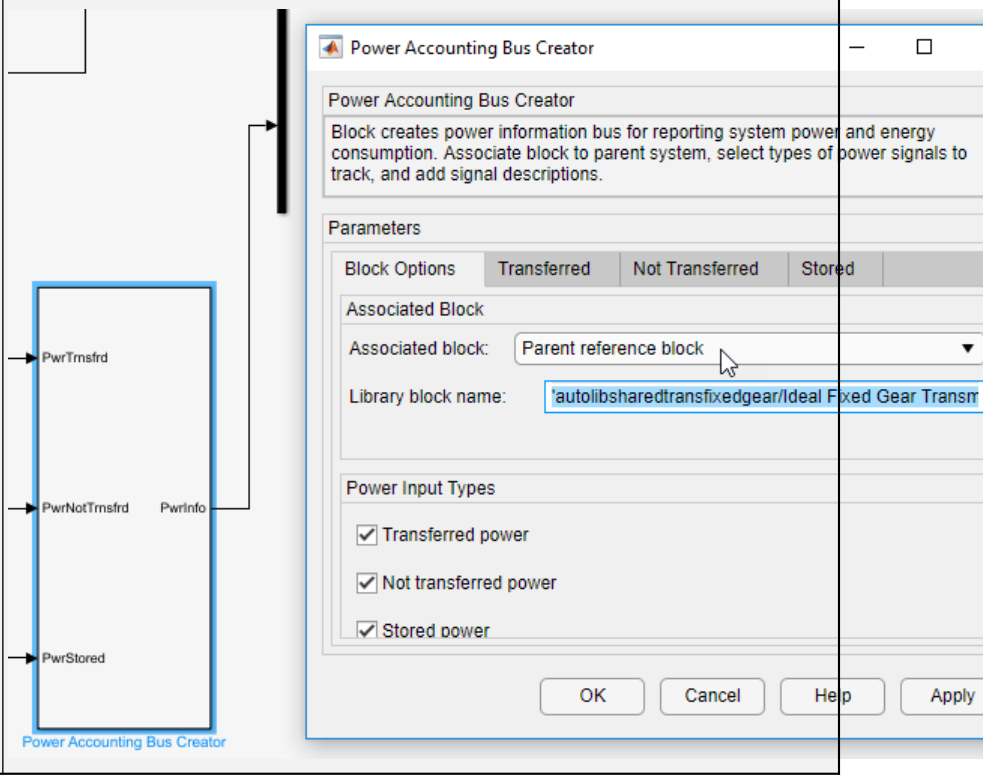
| Power Type | | Description | Examples |
|---|---|---|---|
| $P_{trans}$ | Transferred | Power transferred between blocks:<br>• Positive signals indicate flow into block<br>• Negative signals indicate flow out of block | • Crankshaft power transferred from mapped engine to transmission.<br>• Road load power transferred from wheel to vehicle.<br>• Rate of heat flow transferred from throttle to manifold volume. |
| $P_{nottrans}$ | Not transferred | Power crossing the block boundary, but not transferred:<br>• Positive signals indicate an input<br>• Negative signals indicate a loss | • Rate of heat transfer with the environment.<br>  • From environment is an input (positive signal)<br>  • To environment is a loss (negative signal)<br>• Flow boundary with the environment.<br>  • From environment is an input (positive signal)<br>  • To environment is a loss (negative signal)<br>• Mapped engine fuel flow. |
| $P_{store}$ | Stored | Stored energy rate of change:<br>• Positive signals indicate an increase<br>• Negative signals indicate a decrease | Energy rate of change:<br>• Battery storage<br>• Kinetic energy in drivetrain components<br>• Vehicle potential energy<br>• Vehicle velocity |

### Block Association

When you add the Power Accounting Bus Creator to your plant block, you associate the signals to a parent block. There are two association methods.

| Method | Description | Example |
|---|---|---|
| `Parent` | Power Accounting Bus Creator associates the power bus signals with the parent block. | In the conventional vehicle reference application, navigate to the `Passenger Car > Engine > SiMappedEngine > Accessory Load Model` plant subsystem. Open the Power Accounting Bus Creator.<br><br>The **Associated block** parameter is set to `Parent`, so the Power Accounting Bus Creator associates the power signals with the `Accessory Load Model` plant subsystem. |



| Method | Description | Example |
|---|---|---|
| `Parent reference block` | Power Accounting Bus Creator associates the power bus signals with a reference block.<br><br>Use the **Library block name** parameter to specify the block. | In the Ideal Fixed Gear Transmission block, navigate to the `Bus Creation` subsystem. Open the Power Accounting Bus Creator.<br><br>The Power Accounting Bus Creator block uses these parameter settings to associate the power signals to the Ideal Fixed Gear Transmission block.<br>• **Associated block** parameter is set to `Parent reference block`.<br>• **Library block name** parameter is set to `'autolibsharedtransfixedgear/Ideal Fixed Gear Transmission'`.<br>  |

## Ports

### Input

#### `PwrTrnsfrd` — Power transferred between blocks

bus

`PwrTrnsfrd` — Power transferred between blocks

- Positive signals indicate flow into block
- Negative signals indicate flow out of block

#### Dependencies

To create this input port, select **Transferred power**.

### `PwrNotTrnsfrd` — Power crossing block boundary, not transferred

`bus`

`PwrNotTrnsfrd` — Power crossing the block boundary, but not transferred

- Positive signals indicate an input
- Negative signals indicate a loss

#### Dependencies

To create this input port, select **Not transferred power**.

### `PwrStored` — Stored energy rate of change

`bus`

`PwrStored` — Stored energy rate of change

- Positive signals indicate an increase
- Negative signals indicate a decrease

#### Dependencies

To create this input port, select **Stored power**.

#### Output

### `PwrInfo` — Power information bus

`bus`

Power information bus

## Parameters

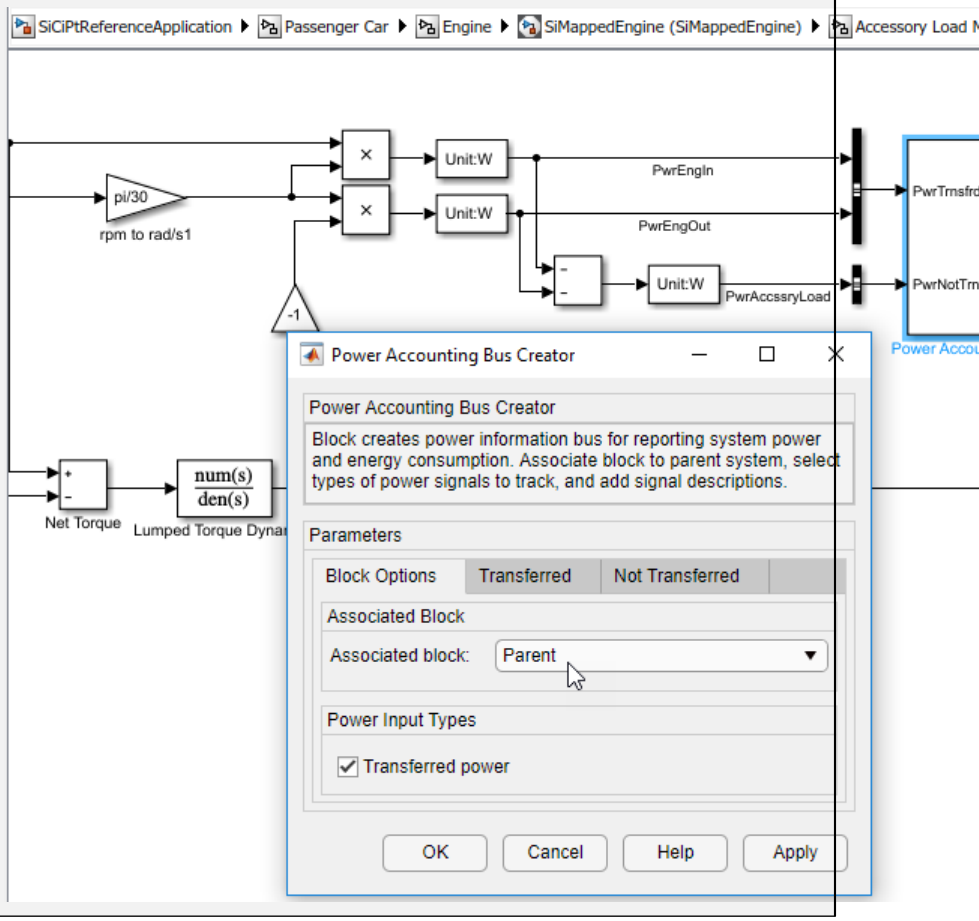
#### Block Options

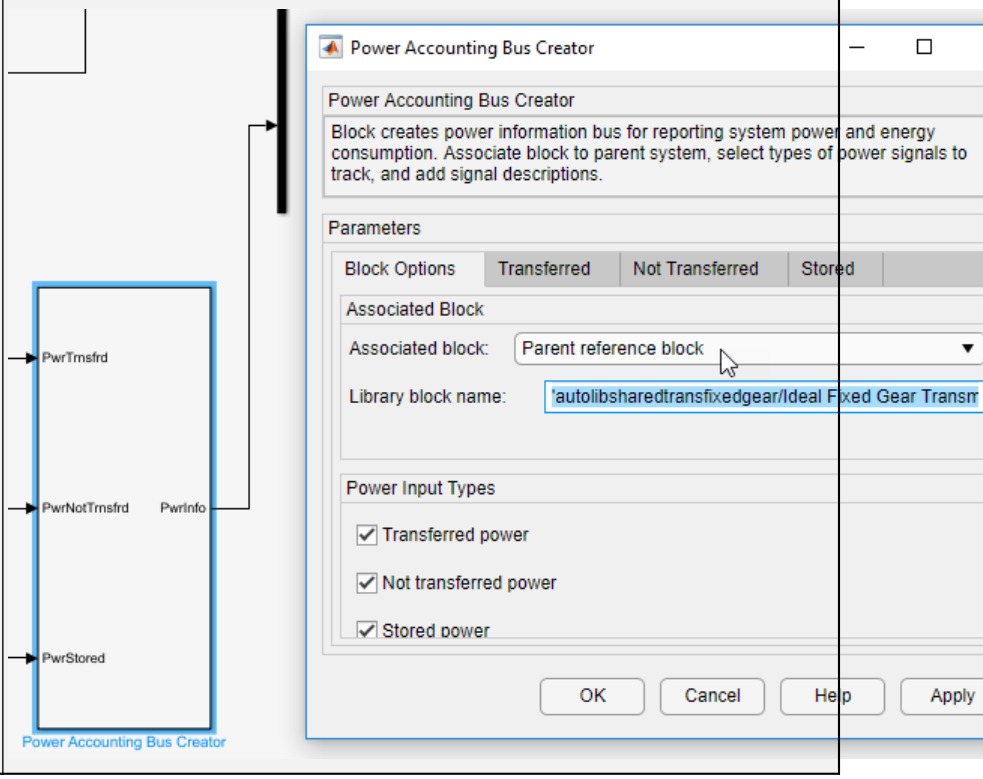
### `Associated block` — Associated block

`Parent` (default) | `Parent reference block`

When you add the Power Accounting Bus Creator to your plant block, you associate the signals to a parent block. There are two association methods.

| Method | Description | Example |
|---|---|---|
| `Parent` | Power Accounting Bus Creator associates the power bus signals with the parent block. | In the conventional vehicle reference application, navigate to the `Passenger Car > Engine > SiMappedEngine > Accessory Load Model` plant subsystem. Open the Power Accounting Bus Creator.<br><br>The **Associated block** parameter is set to `Parent`, so the Power Accounting Bus Creator associates the power signals with the `Accessory Load Model` plant subsystem. |



| Method | Description | Example |
|---|---|---|
| `Parent reference block` | Power Accounting Bus Creator associates the power bus signals with a reference block.<br><br>Use the **Library block name** parameter to specify the block. | In the Ideal Fixed Gear Transmission block, navigate to the `Bus Creation` subsystem. Open the Power Accounting Bus Creator.<br><br>The Power Accounting Bus Creator block uses these parameter settings to associate the power signals to the Ideal Fixed Gear Transmission block.<br>• **Associated block** parameter is set to `Parent reference block`.<br>• **Library block name** parameter is set to `'autolibsharedtransfixedgear/Ideal Fixed Gear Transmission'`.<br>  |

### `Library block name` — Block name

`Block name`

#### Dependencies

To create this parameter, set **Associated block** to `Parent reference block`.

#### Power Input Types

### `Transferred power` — Power transferred between blocks

`on` (default) | `off`

Power transferred between blocks.

#### Dependencies

Selecting this parameter creates the:

- `PwrTrnsfrd` input port
- **Transferred** parameters

### `Not transferred power` — Power crossing block boundary

`on` (default) | `off`

Power crossing block boundary, but not transferred.

#### Dependencies

Selecting this parameter creates the:

- `PwrNotTrnsfrd` input port
- **Not Transferred** parameters

### `Stored power` — Stored energy rate of change

`on` (default) | `off`

Stored energy rate of change.

#### Dependencies

Selecting this parameter creates the:

- `PwrStored` input port
- **Stored** parameters

#### Transferred

### `Signal name` — Name of signal

`char`

Signal name.

For example, this table summarizes the Power Accounting Bus Creator parameter **Transferred** parameter values for the listed blocks.

| Block | Power Accounting Bus Creator Parameter Values | | |
|---|---|---|---|
| | **Signal Name** | **Associated Port** | **Description** |
| Ideal Fixed Gear Transmission | `PwrTrnsfrd.PwrDiffrntl` | `{'DiffTrq','DiffSpd'}` | `Differential` |
| | `PwrTrnsfrd.PwrEng` | `{'EngTrq','EngSpd'}` | `Engine` |
| Gearbox | `PwrTrnsfrd.PwrBase` | `{{'BTrq','BSpd'}'B'}` | `Base input` |
| | `PwrTrnsfrd.PwrFlwr` | `{{'FTrq','FSpd'}'F'}` | `Follower output` |

| Block | Power Accounting Bus Creator Parameter Values | | |
|---|---|---|---|
| | **Signal Name** | **Associated Port** | **Description** |
| Boost Drive Shaft | PwrTrnsfrd.PwrCmpsr | 'Cmpsr' | Compressor |
| | PwrTrnsfrd.PwrExt | 'ExtTrq' | External |
| | PwrTrnsfrd.Turb | 'Turb' | Turbine |

**`Associated Port` — Name of ports that transfer power**
`{'PortA','PortB','PortC'}`

Name of ports that transfer power.

For example, this table summarizes the Power Accounting Bus Creator parameter **Transferred** parameter values for the listed blocks.

| Block | Power Accounting Bus Creator Parameter Values | | |
|---|---|---|---|
| | **Signal Name** | **Associated Port** | **Description** |
| Ideal Fixed Gear Transmission | PwrTrnsfrd.PwrDiffrntl | {'DiffTrq','DiffSpd'} | Differential |
| | PwrTrnsfrd.PwrEng | {'EngTrq','EngSpd'} | Engine |
| Gearbox | PwrTrnsfrd.PwrBase | {{'BTrq','BSpd'}'B'} | Base input |
| | PwrTrnsfrd.PwrFlwr | {{'FTrq','FSpd'}'F'} | Follower output |
| Boost Drive Shaft | PwrTrnsfrd.PwrCmpsr | 'Cmpsr' | Compressor |
| | PwrTrnsfrd.PwrExt | 'ExtTrq' | External |
| | PwrTrnsfrd.Turb | 'Turb' | Turbine |

**`Description` — Signal description**
`char`

Signal description.

For example, this table summarizes the Power Accounting Bus Creator parameter **Transferred** parameter values for the listed blocks.

| Block | Power Accounting Bus Creator Parameter Values | | |
|---|---|---|---|
| | **Signal Name** | **Associated Port** | **Description** |
| Ideal Fixed Gear Transmission | PwrTrnsfrd.PwrDiffrntl | {'DiffTrq','DiffSpd'} | Differential |
| | PwrTrnsfrd.PwrEng | {'EngTrq','EngSpd'} | Engine |
| Gearbox | PwrTrnsfrd.PwrBase | {{'BTrq','BSpd'}'B'} | Base input |
| | PwrTrnsfrd.PwrFlwr | {{'FTrq','FSpd'}'F'} | Follower output |

| Block | Power Accounting Bus Creator Parameter Values | | |
|---|---|---|---|
| | Signal Name | Associated Port | Description |
| Boost Drive Shaft | PwrTrnsfrd.PwrCmpsr | 'Cmpsr' | Compressor |
| | PwrTrnsfrd.PwrExt | 'ExtTrq' | External |
| | PwrTrnsfrd.Turb | 'Turb' | Turbine |

**Not Transferred**

**`Signal name` — Name of signal**
`char`

Signal name.

For example, this table summarizes the Power Accounting Bus Creator parameter **Not Transferred** parameter values for the listed blocks.

| Block | Power Accounting Bus Creator Parameter Values | |
|---|---|---|
| | Signal Name | Description |
| Ideal Fixed Gear Transmission | PwrNotTrnsfrd.PwrDampLoss | Damping loss |
| | PwrNotTrnsfrd.PwrEffLoss | Efficiency loss |
| Gearbox | PwrNotTrnsfrd.PwrDampLoss | Damping loss |
| | PwrNotTrnsfrd.PwrMechLoss | Mechanical loss |
| Boost Drive Shaft | PwrNotTrnsfrd.PwrMechLoss | Mechanical loss |

**`Description` — Signal description**
`char`

Signal description.

For example, this table summarizes the Power Accounting Bus Creator parameter **Not Transferred** parameter values for the listed blocks.

| Block | Power Accounting Bus Creator Parameter Values | |
|---|---|---|
| | Signal Name | Description |
| Ideal Fixed Gear Transmission | PwrNotTrnsfrd.PwrDampLoss | Damping loss |
| | PwrNotTrnsfrd.PwrEffLoss | Efficiency loss |
| Gearbox | PwrNotTrnsfrd.PwrDampLoss | Damping loss |
| | PwrNotTrnsfrd.PwrMechLoss | Mechanical loss |
| Boost Drive Shaft | PwrNotTrnsfrd.PwrMechLoss | Mechanical loss |

**Stored**

**`Signal name` — Name of signal**
`char`

Signal name.

For example, this table summarizes the Power Accounting Bus Creator parameter **Stored** parameter values for the listed blocks.

| Block | Power Accounting Bus Creator Parameter Values | |
|---|---|---|
| | **Signal Name** | **Description** |
| Ideal Fixed Gear Transmission | `PwrStored.PwrStoredTrans` | `Rotational` |
| Control Volume System | `PwrStored.PwrHeatStored` | `Stored heat` |
| Datasheet Battery | `PwrStored.PwrStoredBatt` | `Battery stored` |

**`Description` — Signal description**
`char`

Signal description.

For example, this table summarizes the Power Accounting Bus Creator parameter **Stored** parameter values for the listed blocks.

| Block | Power Accounting Bus Creator Parameter Values | |
|---|---|---|
| | **Signal Name** | **Description** |
| Ideal Fixed Gear Transmission | `PwrStored.PwrStoredTrans` | `Rotational` |
| Control Volume System | `PwrStored.PwrHeatStored` | `Stored heat` |
| Datasheet Battery | `PwrStored.PwrStoredBatt` | `Battery stored` |

## Extended Capabilities

**C/C++ Code Generation**
Generate C and C++ code using Simulink® Coder™.

## See Also
`autoblks.pwr.PlantInfo`

**Topics**
“Conventional Vehicle Powertrain Efficiency”
“Analyze Power and Energy”

**Introduced in R2019a**

# Propulsion Blocks

# Boost Drive Shaft

Boost drive shaft speed

**Library:** Powertrain Blockset / Propulsion / Combustion Engine Components / Boost

## Description

The Boost Drive Shaft block uses the compressor, turbine, and external torques to calculate the drive shaft speed. Use the block to model turbochargers and superchargers in an engine model.

You can specify these configurations:

- Turbocharger — Connect the compressor to the turbine
  - Two-way ports for turbine and compressor connections
  - Option to add an externally applied input torque
- Compressor only — Connect the drive shaft to the compressor
  - Two-way port for compressor connection
  - Externally applied input torque
- Turbine only — Connect the drive shaft to the turbine
  - Two-way port for turbine connection
  - Externally applied load torque

For the `Turbine only` and `Turbocharger` configurations, the block modifies the turbine torque with a mechanical efficiency.

### Equations

The Boost Drive Shaft block applies Newton's Second Law for Rotation. Positive torques cause the drive shaft to accelerate. Negative torques impose a load and decelerate the drive shaft.

The block also calculates the power loss due to mechanical inefficiency.

| Calculation | Equations |
|---|---|
| Shaft dynamics | $\frac{d\omega}{dt} = \frac{1}{J_{shaft}}\left(\eta_{mech}\tau_{turb} + \tau_{comp} + \tau_{ext}\right)$ with initial speed $\omega_0$ |
| Speed constraint | $\omega_{min} \le \omega \le \omega_{max}$ |
| Power loss | $\dot{W}_{loss} = \omega\tau_{turb}(1 - \eta_{mech})$ |

### Power Accounting

For the power accounting, the block implements these equations.

| Bus Signal | | | Description | Equations |
|---|---|---|---|---|
| PwrInfo | PwrTrnsfrd — Power transferred between blocks<br>• Positive signals indicate flow into block<br>• Negative signals indicate flow out of block | PwrCmpsr | Shaft power from compressor | $\tau_{comp}\omega$ |
| | | PwrTurb | Shaft power from turbine | $\tau_{turb}\omega$ |
| | | PwrExt | Externally applied power | $\tau_{ext}\omega$ |
| | PwrNotTrnsfrd — Power crossing the block boundary, but not transferred<br>• Positive signals indicate an input<br>• Negative signals indicate a loss | PwrMechLoss | Mechanical power loss | $-\dot{W}_{turb}$ |
| | PwrStored — Stored energy rate of change<br>• Positive signals indicate an increase<br>• Negative signals indicate a decrease | PwrStoredDriveshft | Rate change in rotational kinetic energy | $\left(\eta_{mech}\tau_{turb} + \tau_{comp} + \tau_{ext}\right)\omega$ |

The equations use these variables.

| | |
|---|---|
| $\omega$ | Shaft speed |
| $\omega_0$ | Initial drive shaft speed |
| $\omega_{min}$ | Minimum drive shaft speed |
| $\omega_{max}$ | Maximum drive shaft speed |
| $J_{shaft}$ | Shaft inertia |
| $\eta_{max}$ | Mechanical efficiency of turbine |
| $\tau_{comp}$ | Compressor torque |
| $\tau_{turb}$ | Turbine torque |
| $\tau_{ext}$ | Externally applied torque. |
| $\dot{W}_{loss}$ | Power loss due to mechanical inefficiency |

## Ports

### Input

#### Cmprs — Compressor torque

two-way connector port

Compressor torque, $\tau_{comp}$, in N·m.

#### Dependencies

To create this port, for the **Configuration** parameter, select `Turbocharger` or `Compressor only`.

### `Turb` — Turbine torque

two-way connector port

Turbine torque, $\tau_{turb}$, in N·m.

#### Dependencies

To create this port, for the **Configuration** parameter, select `Turbocharger` or `Turbine only`.

### `ExtTrq` — Externally applied torque

`scalar`

Externally applied torque, $\tau_{ext}$, in N·m.

#### Dependencies

For turbocharger configurations, to create this port, set **Additional torque input** to `External torque input`.

## Output

### `Info` — Bus signal

bus

Bus signal containing these block calculations.

| Signal | | | Description | Units |
|---|---|---|---|---|
| `DriveshftSpd` | | | Shaft speed | rad/s |
| `MechPwrLoss` | | | Mechanical power loss | W |
| `ExtTrq` | | | Applied external torque | N·m |
| `PwrInfo` | `PwrTrnsfrd` | `PwrCmpsr` | Shaft power from compressor | W |
| | | `PwrTurb` | Shaft power from turbine | W |
| | | `PwrExt` | Externally applied power | W |
| | `PwrNotTrnsfrd` | `PwrMechLoss` | Mechanical power loss | W |
| | `PwrStored` | `PwrStoredDriveshft` | Rate change in rotational kinetic energy | W |

### `Cmprs` — Compressor speed

two-way connector port

Compressor speed, $\omega$, in rad/s.

#### Dependencies

To create this port, for the **Configuration** parameter, select `Turbocharger` or `Compressor only`.

### `Turb` — Turbine speed

two-way connector port

Turbine speed, $\omega$, in N·m.

#### Dependencies

To create this port, for the **Configuration** parameter, select `Turbocharger` or `Turbine only`.

## Parameters

#### Block Options

**`Configuration` — Specify configuration**
`Turbocharger` (default) | `Turbine only` | `Compressor only`

#### Dependencies

- Selecting `Turbocharger` or `Compressor only` creates the `Cmprs` port.
- Selecting `Turbocharger` or `Turbine only` creates the `Turb` port.

**`Additional torque input` — Specify external torque input**
`External torque input` (default) | `No external torque`

#### Dependencies

- To enable this parameter, select a `Turbocharger` configuration.
- To create the `Trq` port, select `External torque input`.

**`Shaft inertia, J_shaft` — Inertia**
`1.55e-5` (default) | `scalar`

Shaft inertia, $J_{shaft}$, in kg·m^2.

**`Initial shaft speed, w_0` — Speed**
`1000` (default) | `scalar`

Initial drive shaft speed, $\omega_0$, in rad/s.

**`Min shaft speed, w_min` — Speed**
`100` (default) | `scalar`

Minimum drive shaft speed, $\omega_{min}$, in rad/s.

**`Max shaft speed, w_max` — Speed**
`20000` (default) | `scalar`

Maximum drive shaft speed, $\omega_{max}$, in rad/s.

**`Turbine mechanical efficiency, eta_mech` — Efficiency**
`0.95` (default) | `scalar`

Mechanical efficiency of turbine $\eta_{max}$.

#### Dependencies

To enable this parameter, select the `Turbocharger` or `Turbine only` configuration.

## Extended Capabilities

### C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

## See Also

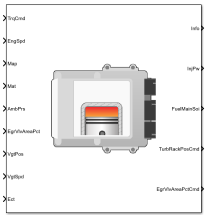
Compressor | Turbine

**Introduced in R2017a**

# CI Controller

Compression-ignition controller that includes air mass flow, torque, and EGR estimation

**Library:** Powertrain Blockset / Propulsion / Combustion Engine Controllers



## Description

The CI Controller block implements a compression-ignition (CI) controller with air mass flow, torque, exhaust gas recirculation (EGR) flow, exhaust back-pressure, and exhaust gas temperature estimation. You can use the CI Controller block in engine control design or performance, fuel economy, and emission tradeoff studies. The core engine block requires the commands that are output from the CI Controller block.

The block uses the commanded torque and measured engine speed to determine these open-loop actuator commands:

- Injector pulse-width
- Fuel injection timing
- Variable geometry turbocharger (VGT) rack position
- EGR valve area percent

The CI Controller block has two subsystems:

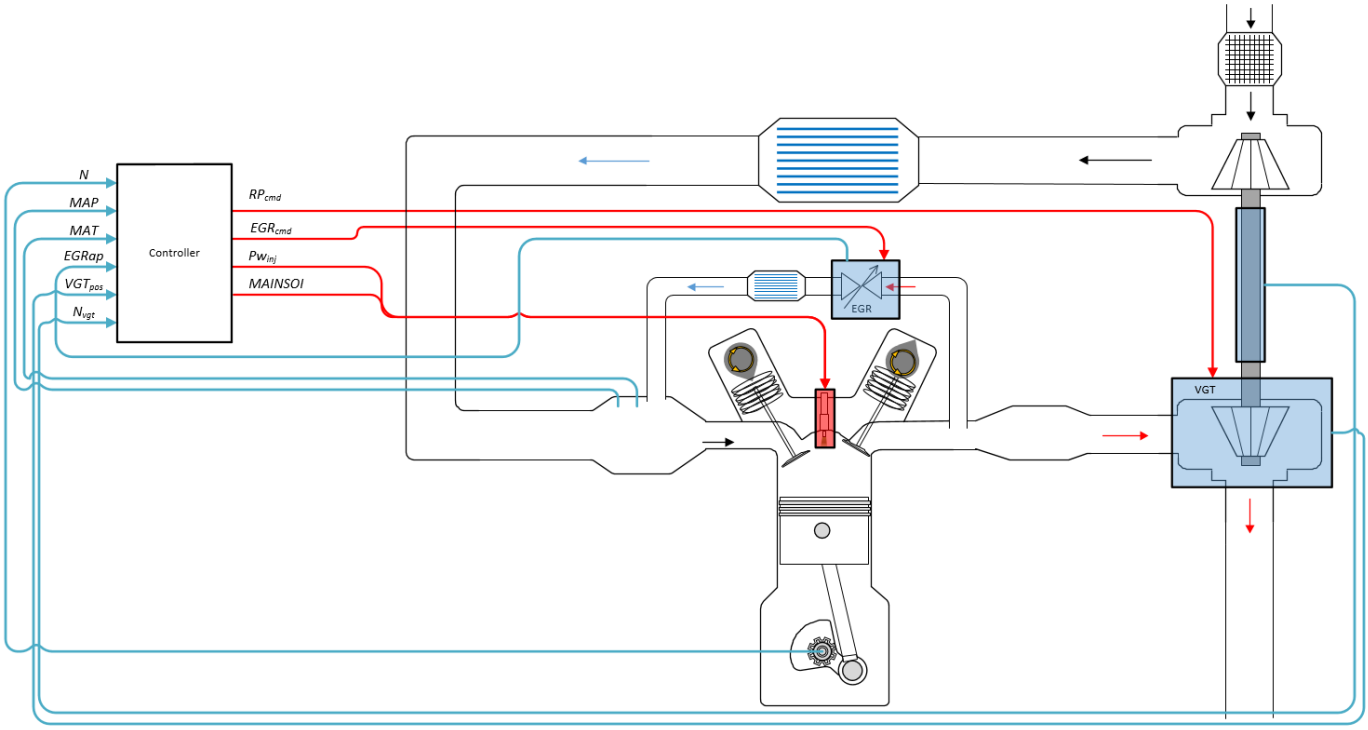
- The `Controller` subsystem — Determines the commands based on tables that are functions of commanded torque and measured engine speed.

  | Based On | Determines Commands for |
  |---|---|
  | Commanded torque<br><br>Measured engine speed | Injector pulse-width<br><br>Fuel injection timing<br><br>VGT rack position<br><br>EGR valve area percent |

- The `Estimator` subsystem — Determines estimates based on these engine attributes.

| Based On | Estimates |
|---|---|
| Measured engine speed<br>Fuel injection timing<br>Cycle average intake manifold pressure and temperature<br>Fuel injector pulse-width<br>Absolute ambient pressure<br>EGR valve area percent<br>VGT rack position<br>VGT speed | Air mass flow<br>Torque<br>Exhaust gas temperature<br>Exhaust gas back-pressure<br>EGR valve gas mass flow |

The figure illustrates the signal flow.




The figure uses these variables.

| | |
|---|---|
| $N$ | Engine speed |
| $MAP$ | Cycle average intake manifold absolute pressure |
| $MAT$ | Cycle average intake manifold gas absolute temperature |
| $EGRap$, $EGR_{cmd}$ | EGR valve area percent and EGR valve area percent command, respectively |
| $VGT_{pos}$ | VGT rack position |

| | |
|---|---|
| $N_{vgt}$ | Corrected turbocharger speed |
| $RP_{cmd}$ | VGT rack position command |
| $Pw_{inj}$ | Fuel injector pulse-width |
| *MAINSOI* | Start of injection timing for main fuel injection pulse |

The Model-Based Calibration Toolbox™ was used to develop the tables that are available with the Powertrain Blockset.

### Controller

The controller governs the combustion process by commanding VGT rack position, EGR valve area percent, fuel injection timing, and injector pulse-width. Feedforward lookup tables, which are functions of measured engine speed and commanded torque, determine the control commands.

#### Air

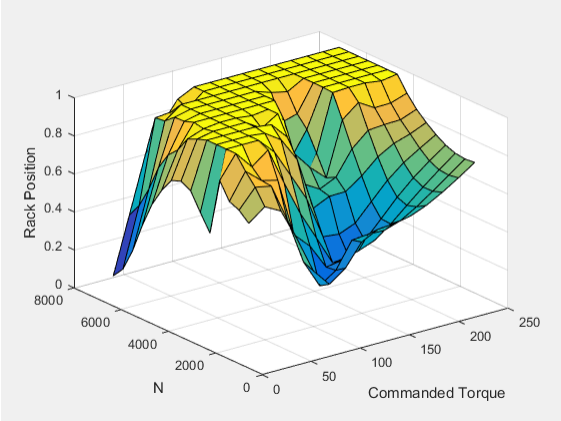
The controller commands the EGR valve area percent and VGT rack position. Changing the VGT rack position modifies the turbine flow characteristics. At low-requested torques, the rack position can reduce the exhaust back pressure, resulting in a low turbocharger speed and boost pressure. When the commanded fuel requires additional air mass flow, the rack position is set to close the turbocharger vanes, increasing the turbocharger speed and intake manifold boost pressure.

The variable geometry turbocharger (VGT) rack position lookup table is a function of commanded torque and engine speed

$$RP_{cmd} = f_{RPcmd}(Trq_{cmd}, N)$$

where:

- $RP_{cmd}$ is VGT rack position command, in percent.
- $Trq_{cmd}$ is commanded engine torque, in N·m.
- $N$ is engine speed, in rpm.

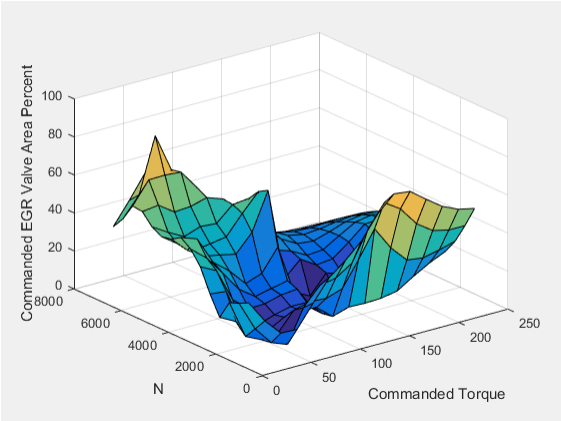



The commanded exhaust gas recirculation (EGR) valve area percent lookup table is a function of commanded torque and engine speed

$$EGR_{cmd} = f_{EGRcmd}(Trq_{cmd}, N)$$

where:

- $EGR_{cmd}$ is commanded EGR valve area percent, in percent.
- $Trq_{cmd}$ is commanded engine torque, in N·m.
- $N$ is engine speed, in rpm.




**Fuel**

To initiate combustion, a CI engine injects fuel directly into the combustion chamber. After the injection, the fuel spontaneously ignites, increasing cylinder pressure. The total mass of the injected fuel and main injection timing determines the torque production.

Assuming constant fuel rail pressure, the CI controller commands the injector pulse-width based on the total requested fuel mass:

$$Pw_{inj} = \frac{F_{cmd,\ tot}}{S_{inj}}$$

The equation uses these variables.

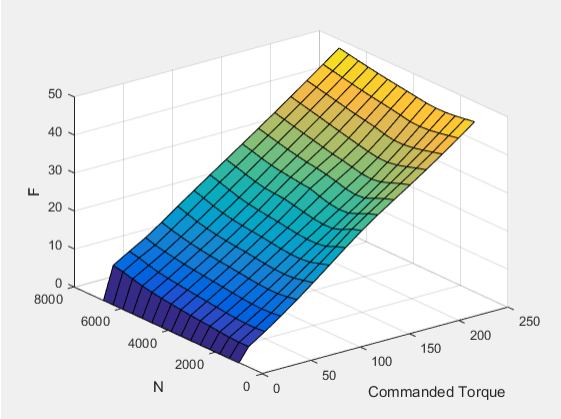
| | |
|---|---|
| $Pw_{inj}$ | Fuel injector pulse-width |
| $S_{inj}$ | Fuel injector slope |
| $F_{cmd,tot}$ | Commanded total fuel mass per injection |
| $MAINSOI$ | Main start-of-injection timing |
| $N$ | Engine speed |

The commanded total fuel mass per injection table is a function of the torque command and engine speed

$$F_{cmd,\,tot} = f_{Fcmd,\,tot}(Trq_{cmd}, N)$$

where:

- $F_{cmd,tot} = F$ is commanded total fuel mass per injection, in mg per cylinder.
- $Trq_{cmd}$ is commanded engine torque, in N·m.
- $N$ is engine speed, in rpm.

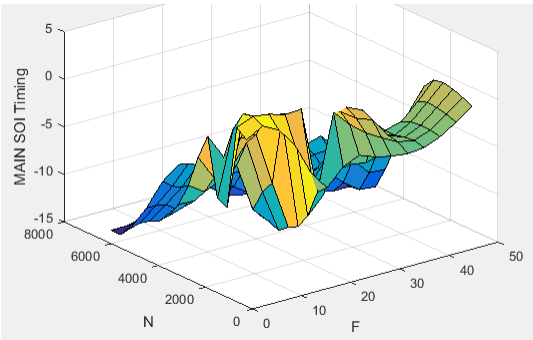



The main start-of-injection (SOI) timing lookup table is a function of commanded fuel mass and engine speed

$$MAINSOI = f(F_{cmd,\,tot}, N)$$

where:

- *MAINSOI* is the main start-of-injection timing, in degrees crank angle after top dead center (degATDC).
- $F_{cmd,tot} = F$ is commanded fuel mass, in mg per injection.
- $N$ is engine speed, in rpm.




**Idle Speed**

When the commanded torque is below a threshold value, the idle speed controller regulates the engine speed.

| If | Idle Speed Controller |
|---|---|
| $Trq_{cmd,input} < Trq_{idlecmd,enable}$ | Enabled |
| $Trq_{idlecmd,enable} \le Trq_{cmd,input}$ | Not enabled |

The idle speed controller uses a discrete PI controller to regulate the target idle speed by commanding a torque.

The PI controller uses this transfer function:

$$C_{idle}(z) = K_{p,\,idle} + K_{i,\,idle}\frac{t_s}{z-1}$$

The idle speed commanded torque must be less than the maximum commanded torque:

$0 \le Trq_{idlecomd} \le Trq_{idlecmd,max}$

Idle speed control is active under these conditions. If the commanded input torque drops below the threshold for enabling the idle speed controller ($Trq_{cmd,input} < Trq_{idlecmd,enable}$), the commanded engine torque is given by:

$Trq_{cmd} = \max(Trq_{cmd,input}, Trq_{idlecmd})$.

The equations use these variables.

| | |
|---|---|
| $Trq_{cmd}$ | Commanded engine torque |
| $Trq_{cmd,input}$ | Input commanded engine torque |
| $Trq_{idlecmd,enable}$ | Threshold for enabling idle speed controller |
| $Trq_{idlecmd}$ | Idle speed controller commanded torque |
| $Trq_{idlecmd,max}$ | Maximum commanded torque |
| $N_{idle}$ | Base idle speed |
| $K_{p,idle}$ | Idle speed controller proportional gain |
| $K_{i,idle}$ | Idle speed controller integral gain |

#### Speed Limiter

To prevent over revving the engine, the block implements an engine speed limit controller that limits the engine speed to the value specified by the **Rev-limiter speed threshold** parameter on the **Controls** > **Idle Speed** tab.

If the engine speed, $N$, exceeds the engine speed limit, $N_{lim}$, the block sets the commanded engine torque to 0.

To smoothly transition the torque command to 0 as the engine speed approaches the speed limit, the block implements a lookup table multiplier. The lookup table multiplies the torque command by a value that ranges from 0 (engine speed exceeds limit) to 1 (engine speed does not exceed the limit).

### Estimator

Using the CI Core Engine block, the CI Controller block estimates the air mass flow rate, EGR valve mass flow, exhaust back-pressure, engine torque, AFR, and exhaust temperature from sensor feedback. The `Info` port provides the estimated values, but block does not use them to determine the open-loop engine actuator commands.

#### Air Mass Flow

To calculate the air mass flow, the compression-ignition (CI) engine uses the “CI Engine Speed-Density Air Mass Flow Model”. The speed-density model uses the speed-density equation to calculate the engine air mass flow, relating the engine intake port mass flow to the intake manifold pressure, intake manifold temperature, and engine speed.

#### EGR Valve Mass Flow

To calculate the estimated exhaust gas recirculation (EGR) valve mass flow, the block calculates the EGR flow that would occur at standard temperature and pressure conditions, and then corrects the

flow to actual temperature and pressure conditions. The block EGR calculation uses estimated exhaust back-pressure, estimated exhaust temperature, standard temperature, and standard pressure.

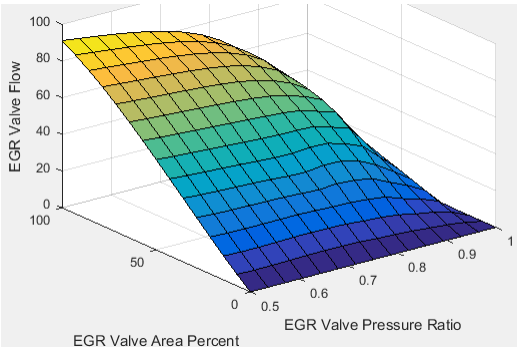
$$\dot{m}_{egr,\,est} = \dot{m}_{egr,\,std}\frac{P_{exh,\,est}}{P_{std}}\sqrt{\frac{T_{std}}{T_{exh,\,est}}}$$

- The standard exhaust gas recirculation (EGR) mass flow is a lookup table that is a function of the standard flow pressure ratio and EGR valve flow area

  $$\dot{m}_{egr,\,std} = f\left(\frac{MAP}{P_{exh,\,est}}, EGRap\right)$$

  where:

  - $\dot{m}_{egr,\,std}$ is the standard EGR valve mass flow, in g/s.
  - $P_{exh,est}$ is the estimated exhaust back-pressure, in Pa.
  - $MAP$ is the cycle average intake manifold absolute pressure, in Pa.
  - $EGRap$ is the measured EGR valve area, in percent.




The equations use these variables.

| | |
|---|---|
| $\dot{m}_{egr,\,est}$ | Estimated EGR valve mass flow |
| $\dot{m}_{egr,\,std}$ | Standard EGR valve mass flow |
| $P_{std}$ | Standard pressure |
| $T_{std}$ | Standard temperature |
| $T_{exh,est}$ | Estimated exhaust manifold gas temperature |
| $MAP$ | Measured cycle average intake manifold absolute pressure |
| $P_{exh,est}$ | Estimated exhaust back-pressure |
| $P_{Amb}$ | Absolute ambient pressure |
| $EGRap$ | Measured EGR valve area percent |

#### Exhaust Back-Pressure

To estimate the EGR valve mass flow, the block requires an estimate of the exhaust back-pressure. To estimate the exhaust back-pressure, the block uses the ambient pressure and the turbocharger pressure ratio.

$$P_{exh,\,est} = P_{Amb} Pr_{turbo}$$

For the turbocharger pressure ration calculation, the block uses two lookup tables. The first lookup table determines the approximate turbocharger pressure ratio as a function of turbocharger mass flow and corrected turbocharger speed. Using a second lookup table, the block corrects the approximate turbocharger pressure ratio for VGT rack position.

$$Pr_{turbo} = f(\dot{m}_{airstd}, N_{vgtcorr}) f(VGT_{pos})$$

where:

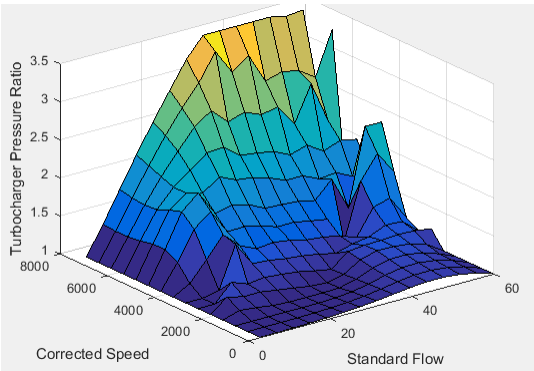
$$N_{vgtcorr} = \frac{N_{vgt}}{\sqrt{T_{exh,\,est}}}$$

The equations use these variables.

| | |
|---|---|
| $\dot{m}_{egr,\,est}$ | Estimated EGR valve mass flow |
| $\dot{m}_{egr,\,std}$ | Standard EGR valve mass flow |
| $\dot{m}_{port,\,est}$ | Estimated intake port mass flow rate |
| $\dot{m}_{airstd}$ | Standard air mass flow |
| $EGRap$ | Measured EGR valve area |
| $MAP$ | Measured cycle average intake manifold absolute pressure |
| $MAT$ | Measured cycle average intake manifold gas absolute temperature |
| $P_{std}$ | Standard pressure |
| $T_{std}$ | Standard temperature |
| $T_{exh,est}$ | Estimated exhaust manifold gas temperature |
| $Pr_{vgtcorr}$ | Turbocharger pressure ratio correction for VGT rack position |
| $Pr_{turbo}$ | Turbocharger pressure ratio |
| $P_{exh,est}$ | Estimated exhaust back-pressure |
| $P_{Amb}$ | Absolute ambient pressure |
| $N_{vgtcorr}$ | Corrected turbocharger speed |
| $VGT_{pos}$ | Measured VGT rack position |

The exhaust-back pressure calculation uses these lookup tables:

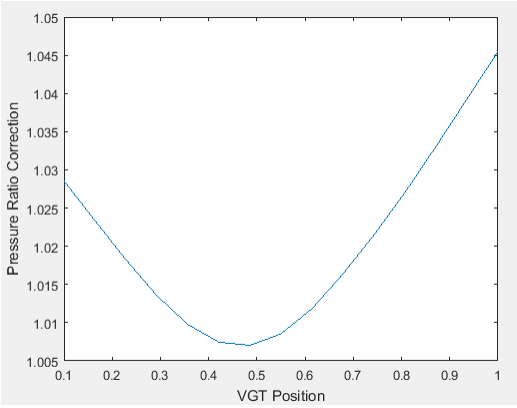
- The turbocharger pressure ratio, corrected for variable geometry turbocharger (VGT) speed, is a lookup table that is a function of the standard air mass flow and corrected turbocharger speed, $Pr_{turbo} = f(\dot{m}_{airstd}, N_{vgtcorr})$, where:
  - $Pr_{turbo}$ is the turbocharger pressure ratio, corrected for VGT speed.
  - $\dot{m}_{airstd}$ is the standard air mass flow, in g/s.
  - $N_{vgtcorr}$ is the corrected turbocharger speed, in rpm/K^(1/2).




To calculate the standard air mass flow through the turbocharger, the block uses conservation of mass, the estimated intake port, and EGR mass flows (from the last estimated calculation). The calculation assumes negligible exhaust manifold filling dynamics.

$$\dot{m}_{airstd} = (\dot{m}_{port,est} - \dot{m}_{egr,est})\frac{P_{std}}{MAP}\sqrt{\frac{MAT}{T_{std}}}$$

- The variable geometry turbocharger pressure ratio correction is a function of the rack position, $Pr_{vgtcorr} = f(VGT_{pos})$, where:
  - $Pr_{vgtcorr}$ is the turbocharger pressure ratio correction.
  - $VGT_{pos}$ is the variable geometry turbocharger (VGT) rack position.




**Engine Torque**

To calculate the engine torque, you can configure the block to use either of these torque models.

| Brake Torque Model | Description |
|---|---|
| “CI Engine Torque Structure Model” | The CI core engine torque structure model determines the engine torque by reducing the maximum engine torque potential as these engine conditions vary from nominal:<br>• Start of injection (SOI) timing<br>• Exhaust back-pressure<br>• Burned fuel mass<br>• Intake manifold gas pressure, temperature, and oxygen percentage<br>• Fuel rail pressure<br>To account for the effect of post-inject fuel on torque, the model uses a calibrated torque offset table. |
| “CI Engine Simple Torque Model” | For the simple engine torque calculation, the CI engine uses a torque lookup table map that is a function of engine speed and injected fuel mass. |

#### Exhaust Temperature

The exhaust temperature calculation depends on the torque model. For both torque models, the block implements lookup tables.

| Torque Model | Description | Equations |
|---|---|---|
| `Simple Torque Lookup` | Exhaust temperature lookup table is a function of the injected fuel mass and engine speed. | $T_{exh} = f_{Texh}(F, N)$ |
| `Torque Structure` | The nominal exhaust temperature, $Texh_{nom}$, is a product of these exhaust temperature efficiencies:<br>• SOI timing<br>• Intake manifold gas pressure<br>• Intake manifold gas temperature<br>• Intake manifold gas oxygen percentage<br>• Fuel rail pressure<br>• Optimal temperature<br>The exhaust temperature, $Texh_{nom}$, is offset by a post temperature effect, $\Delta T_{post}$, that accounts for post and late injections during the expansion and exhaust strokes. | $T_{exhnom} = SOI_{exhteff} MAP_{exhteff} MAT_{exhteff} O2p_{exhteff} FU$<br>$T_{exh} = T_{exhnom} + \Delta T_{post}$<br>$SOI_{exhteff} = f_{SOI_{exhteff}}(\Delta SOI, N)$<br>$MAP_{exhteff} = f_{MAP_{exhteff}}(MAP_{ratio}, \lambda)$<br>$MAT_{exhteff} = f_{MAT_{exhteff}}(\Delta MAT, N)$<br>$O2p_{exhteff} = f_{O2p_{exhteff}}(\Delta O2p, N)$<br>$Texh_{opt} = f_{Texh}(F, N)$ |

The equations use these variables.

| | |
|---|---|
| $F$ | Compression stroke injected fuel mass |
| $N$ | Engine speed |
| $Texh$ | Exhaust manifold gas temperature |
| $Texh_{opt}$ | Optimal exhaust manifold gas temperature |
| $\Delta T_{post}$ | Post injection temperature effect |
| $Texh_{nom}$ | Nominal exhaust temperature |
| $SOI_{exhteff}$ | Main SOI exhaust temperature efficiency multiplier |
| $\Delta SOI$ | Main SOI timing relative to optimal timing |
| $MAP_{exheff}$ | Intake manifold gas pressure exhaust temperature efficiency multiplier |
| $MAP_{ratio}$ | Intake manifold gas pressure ratio relative to optimal pressure ratio |
| $\lambda$ | Intake manifold gas lambda |
| $MAT_{exheff}$ | Intake manifold gas temperature exhaust temperature efficiency multiplier |
| $\Delta MAT$ | Intake manifold gas temperature relative to optimal temperature |
| $O2P_{exheff}$ | Intake manifold gas oxygen exhaust temperature efficiency multiplier |
| $\Delta O2P$ | Intake gas oxygen percent relative to optimal |
| $FUELP_{exheff}$ | Fuel rail pressure exhaust temperature efficiency multiplier |
| $\Delta FUELP$ | Fuel rail pressure relative to optimal |

#### Air-Fuel Ratio

The measured engine speed and fuel injector pulse-width determine the commanded fuel mass flow rate:

$$\dot{m}_{fuel,cmd} = \frac{NS_{inj}Pw_{inj}N_{cyl}}{Cps\left(\frac{60s}{min}\right)\left(\frac{1000mg}{g}\right)}$$

The commanded total fuel mass flow and estimated port mass flow rates determine the estimated AFR:

$$AFR_{est} = \frac{\dot{m}_{port,est}}{\dot{m}_{fuel,cmd}}$$

The equations use these variables.

| | |
|---|---|
| $Pw_{inj}$ | Fuel injector pulse-width |
| $AFR_{est}$ | Estimated air-fuel ratio |
| $\dot{m}_{fuel,cmd}$ | Commanded fuel mass flow rate |
| $S_{inj}$ | Fuel injector slope |
| $N$ | Engine speed |
| $N_{cyl}$ | Number of engine cylinders |
| $Cps$ | Crankshaft revolutions per power stroke, rev/stroke |
| $\dot{m}_{port,est}$ | Total estimated engine air mass flow at intake ports |

## Ports

### Input

**`TrqCmd` — Commanded engine torque**
scalar

Commanded engine torque, $Trq_{cmd,input}$, in N·m.

**`EngSpd` — Measured engine speed**
scalar

Measured engine speed, $N$, in rpm.

**`Map` — Measured intake manifold absolute pressure**
scalar

Measured intake manifold absolute pressure, $MAP$, in Pa.

**`Mat` — Measured intake manifold absolute temperature**
scalar

Measured intake manifold absolute temperature, $MAT$, in K.

**`AmbPrs` — Ambient pressure**
scalar

Absolute ambient pressure, $P_{Amb}$, in Pa.

**`EgrVlvAreaPct` — EGR valve area percent**
scalar

Measured EGR valve area percent, $EGRap$, in %.

**`VgtPos` — VGT speed**
scalar

Measured VGT rack position, $VGT_{pos}$.

**`VgtSpd` — VGT speed**
scalar

Measured VGT speed, $N_{vgt}$, in rpm.

**`Ect` — Engine cooling temperature**
scalar

Engine cooling temperature, $T_{coolant}$, in K.

### Output

**`Info` — Bus signal**
bus

Bus signal containing these block calculations.

| Signal | Description | Variable | Units |
|---|---|---|---|
| `InjPw` | Fuel injector pulse-width | $Pw_{inj}$ | ms |
| `EgrVlvAreaPctCmd` | EGR valve area percent command | $EGR_{cmd}$ | % |
| `TurbRackPosCmd` | VGT rack position command | $RP_{cmd}$ | N/A |
| `TrqCmd` | Engine torque | $Trq_{cmd}$ | N·m |
| `FuelMassTotCmd` | Commanded total fuel mass per injection | $F_{cmd,tot}$ | mg |
| `FuelMainSoi` | Main start-of-injection timing | $MAINSOI$ | degATDC |
| `FuelMassFlwCmd` | Commanded fuel mass flow rate | $\dot{m}_{fuel,\,cmd}$ | kg/s |
| `EstIntkPortMassFlw` | Estimated port mass flow rate | $\dot{m}_{port,\,est}$ | kg/s |
| `EstEngTrq` | Estimated engine torque | $Trq_{est}$ | N·m |
| `EstExhManGasTemp` | Estimated exhaust manifold gas temperature | $T_{exh,est}$ | K |
| `EstExhPrs` | Estimated exhaust back-pressure | $Pex$ | Pa |
| `EstEGRFlow` | `EstEGRFlow` | `EstEGRFlow` | `EstEGRFlow` |
| `EstAfr` | Estimated air-fuel ratio | $AFR_{est}$ | N/A |
| `EngRevLimAct` | Flag that indicates if rev-limiter control is active | N/A | N/A |

**`InjPw` — Fuel injector pulse-width**
`scalar`

Fuel injector pulse-width, $Pw_{inj}$, in ms.

**`FuelMainSoi` — Fuel main injecting timing**
`scalar`

Main start-of-injection timing, *MAINSOI*, in degrees crank angle after top dead center (degATDC).

**`TurbRackPosCmd` — Rack position**
`scalar`

VGT rack position command, $RP_{cmd}$.

**`EgrVlvAreaPctCmd` — Intake cam phaser angle command**
`scalar`

EGR valve area percent command, $EGR_{cmd}$.

## Parameters

### Controls

#### Air - EGR

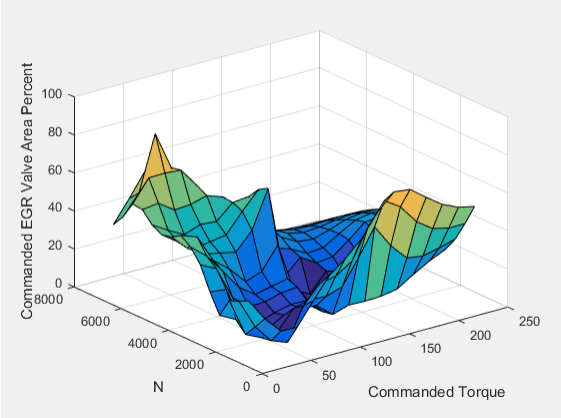
**`EGR valve area percent, f_egrcmd` — Lookup table**
`array`

The commanded exhaust gas recirculation (EGR) valve area percent lookup table is a function of commanded torque and engine speed

$$EGR_{cmd} = f_{EGRcmd}(Trq_{cmd}, N)$$

where:

- $EGR_{cmd}$ is commanded EGR valve area percent, in percent.
- $Trq_{cmd}$ is commanded engine torque, in N·m.
- $N$ is engine speed, in rpm.




**`Commanded torque breakpoints, f_egr_tq_bpt` — Breakpoints**
`[10 26.43 42.86 59.29 75.71 92.14 108.6 125 141.4 157.9 174.3 190.7 207.1 223.6 240]` (default) | `vector`

Commanded torque breakpoints, in N·m.

**`Speed breakpoints, f_egr_n_bpt` — Breakpoints**
`[1000 1411 1821 2232 2643 3054 3464 3875 4286 4696 5107 5518 5929 6339 6750]` (default) | `vector`

Speed breakpoints, in rpm.

**Air - VGR**

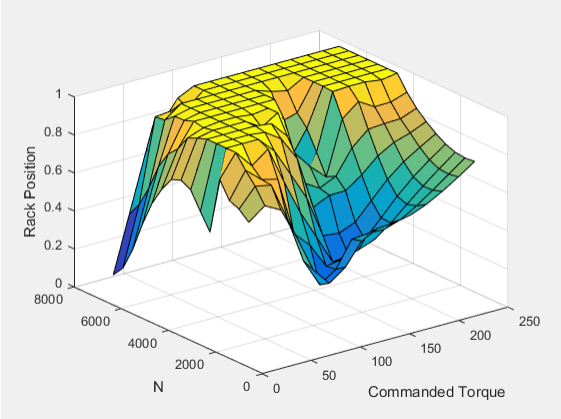
**`VGT rack position table, f_rpcmd` — Lookup table**
`array`

The variable geometry turbocharger (VGT) rack position lookup table is a function of commanded torque and engine speed

$$RP_{cmd} = f_{RPcmd}(Trq_{cmd}, N)$$

where:

- $RP_{cmd}$ is VGT rack position command, in percent.
- $Trq_{cmd}$ is commanded engine torque, in N·m.
- $N$ is engine speed, in rpm.




**`Commanded torque breakpoints, f_rp_tq_bpt` — Breakpoints**
`[10 26.43 42.86 59.29 75.71 92.14 108.6 125 141.4 157.9 174.3 190.7 207.1 223.6 240]` (default) | `vector`

Breakpoints, in N·m.

**`Speed breakpoints, f_rp_n_bpt` — Breakpoints**
`[1000 1411 1821 2232 2643 3054 3464 3875 4286 4696 5107 5518 5929 6339 6750]` (default) | `vector`

Breakpoints, in rpm.

#### Fuel

**`Injector slope, Sinj` — Slope**
`6.452` (default) | `scalar`

Fuel injector slope, $S_{inj}$, in mg/ms.

**`Stoichiometric air-fuel ratio, afr_stoich` — Ratio**
`14.6` (default) | `scalar`

Stoichiometric air-fuel ratio, $AFR_{stoich}$.

**`Fuel lower heating value, fuel_lhv` — Heat**
`42e6` (default) | `scalar`

Fuel lower heating value, in J/kg.

**`Fuel mass per injection table, f_fcmd_tot` — Lookup table**
`array`

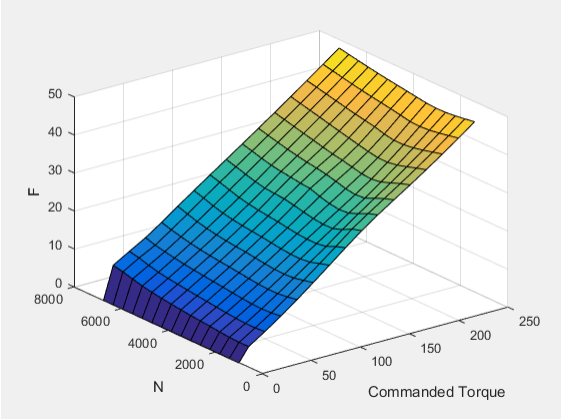
The commanded total fuel mass per injection table is a function of the torque command and engine speed

$$F_{cmd,tot} = f_{Fcmd,tot}(Trq_{cmd}, N)$$

where:

- $F_{cmd,tot} = F$ is commanded total fuel mass per injection, in mg per cylinder.

- $Trq_{cmd}$ is commanded engine torque, in N·m.
- $N$ is engine speed, in rpm.



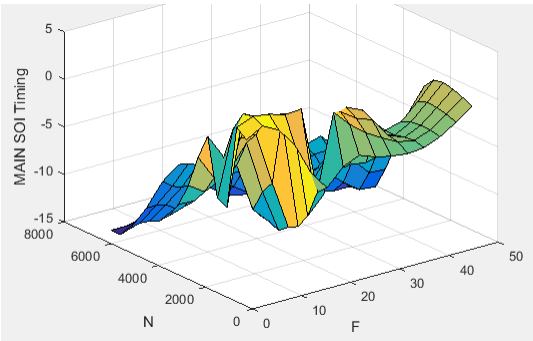

**Fuel main injection timing table, f_main_soi — Lookup table**
array

The main start-of-injection (SOI) timing lookup table is a function of commanded fuel mass and engine speed

$$MAINSOI = f(F_{cmd,tot}, N)$$

where:

- $MAINSOI$ is the main start-of-injection timing, in degrees crank angle after top dead center (degATDC).
- $F_{cmd,tot} = F$ is commanded fuel mass, in mg per injection.
- $N$ is engine speed, in rpm.




**Fuel main injection timing fuel breakpoints, f_main_soi_f_bpt — Breakpoints**
vector

Fuel main injection timing fuel breakpoints, in mg per injection.

**Fuel main injection timing speed breakpoints, f_main_soi_n_bpt — Breakpoints**
[1000,1410.71428571429,1821.42857142857,2232.14285714286,2642.85714285714,3053.57142857143,3464.28571428571,3875,4285.71428571429,4696.42857142857,5107.14

285714286,5517.85714285714,5928.57142857143,6339.28571428572,6750] (default) | vector

Fuel main injection timing speed breakpoints, in rpm.

**Commanded torque breakpoints, f_f_tot_tq_bpt — Breakpoints**
[0 10 26.43 42.86 59.29 75.71 92.14 108.6 125 141.4 157.9 174.3 190.7 207.1 223.6 240] (default) | vector

Commanded torque breakpoints, in N·m.

**Speed breakpoints, f_f_tot_n_bpt — Breakpoints**
[1000 1411 1821 2232 2643 3054 3464 3875 4286 4696 5107 5518 5929 6339 6750] (default) | vector

Speed breakpoints, in rpm.

#### Idle Speed

**Base idle speed, N_idle — Speed**
750 (default) | scalar

Base idle speed, $N_{idle}$, in rpm.

**Enable torque command limit, Trq_idlecmd_enable — Torque**
1 (default) | scalar

Torque to enable the idle speed controller, $Trq_{idlecmd,enable}$, in N·m.

**Maximum torque command, Trq_idlecmd_max — Torque**
50 (default) | scalar

Maximum idle controller commanded torque, $Trq_{idlecmd,max}$, in N·m.

**Proportional gain, Kp_idle — PI Controller**
0.05 (default) | scalar

Proportional gain for idle speed control, $K_{p,idle}$, in N·m/rpm.

**Integral gain, Ki_idle — PI Controller**
0.2 (default) | scalar

Integral gain for idle speed control, $K_{i,idle}$, in N·m/(rpm·s).

**Rev-limiter speed threshold — Engine speed limit**
scalar

Engine speed limit, $N_{lim}$, in rpm.

If the engine speed, $N$, exceeds the engine speed limit, $N_{lim}$, the block sets the commanded engine torque to 0.

To smoothly transition the torque command to 0 as the engine speed approaches the speed limit, the block implements a lookup table multiplier. The lookup table multiplies the torque command by a value that ranges from 0 (engine speed exceeds limit) to 1 (engine speed does not exceed the limit).

### Estimation

#### Air

**`Number of cylinders, NCyl` — Engine cylinders**
`4` (default) | `scalar`

Number of engine cylinders, $N_{cyl}$.

**`Crank revolutions per power stroke, Cps` — Revolutions per stroke**
`2` (default) | `scalar`

Crankshaft revolutions per power stroke, *Cps*, in rev/stroke.

**`Total displaced volume, Vd` — Volume**
`0.0015` (default) | `scalar`

Displaced volume, $V_d$, in m^3.

**`Ideal gas constant air, Rair` — Constant**
`287` (default) | `scalar`

Ideal gas constant, $R_{air}$, in J/(kg·K).

**`Air standard pressure, Pstd` — Pressure**
`101325` (default) | `scalar`

Standard air pressure, $P_{std}$, in Pa.

**`Air standard temperature, Tstd` — Temperature**
`293.15` (default) | `scalar`

Standard air temperature, $T_{std}$, in K.

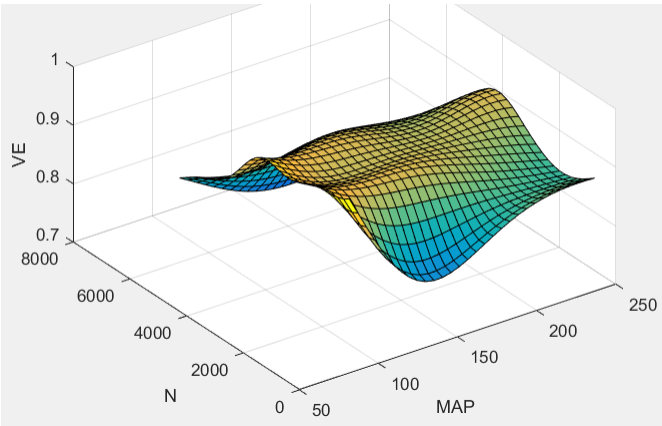
**`Speed density volumetric efficiency, f_nv` — Lookup table**
`array`

The volumetric efficiency lookup table is a function of the intake manifold absolute pressure at intake valve closing (IVC) and engine speed

$$\eta_v = f_{\eta_v}(MAP, N)$$

where:

- $\eta_v$ is engine volumetric efficiency, dimensionless.
- $MAP$ is intake manifold absolute pressure, in KPa.
- $N$ is engine speed, in rpm.




**Speed density intake manifold pressure breakpoints, f_nv_prs_bpt — Breakpoints**
`[95 100.3 105.7 111 116.4 121.7 127.1 132.4 137.8 143.1 148.4 153.8 159.1 164.5 169.8 175.2 180.5 185.9 191.2 196.6 201.9 207.2 212.6 217.9 223.3 228.6 234 239.3 244.7 250]` (default) | `vector`

Intake manifold pressure breakpoints for speed-density volumetric efficiency lookup table, in KPa.

**Speed density engine speed breakpoints, f_nv_n_bpt — Breakpoints**
`[750 956.9 1164 1371 1578 1784 1991 2198 2405 2612 2819 3026 3233 3440 3647 3853 4060 4267 4474 4681 4888 5095 5302 5509 5716 5922 6129 6336 6543 6750]` (default) | `vector`

Engine speed breakpoints for speed-density volumetric efficiency lookup table, in rpm.

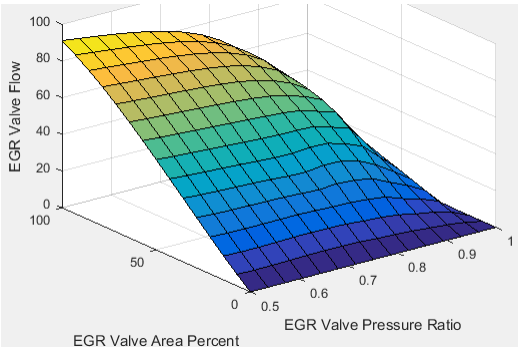
**EGR valve standard flow calibration, f_egr_stdflow — Lookup table**
`array`

The standard exhaust gas recirculation (EGR) mass flow is a lookup table that is a function of the standard flow pressure ratio and EGR valve flow area

$$\dot{m}_{egr,\,std} = f(\frac{MAP}{P_{exh,\,est}}, EGRap)$$

where:

- $\dot{m}_{egr,\,std}$ is the standard EGR valve mass flow, in g/s.
- $P_{exh,est}$ is the estimated exhaust back-pressure, in Pa.
- $MAP$ is the cycle average intake manifold absolute pressure, in Pa.
- $EGRap$ is the measured EGR valve area, in percent.




**`EGR valve standard flow pressure ratio breakpoints, f_egr_stdflow_pr_bpt` — Breakpoints**
`vector`

EGR valve standard flow pressure ratio breakpoints, dimensionless.

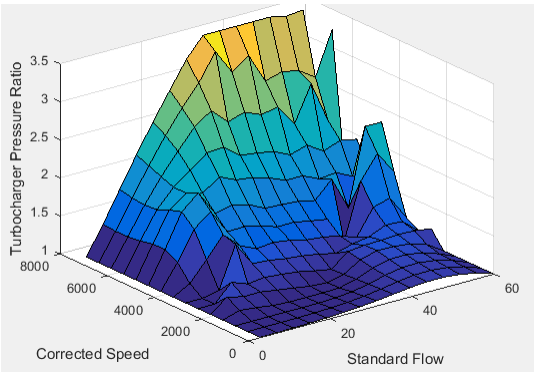
**`EGR valve standard flow area percent breakpoints, f_egr_stdflow_egrap_bpt` — Breakpoints**
`vector`

EGR valve standard flow area percent breakpoints, in percent.

**`Turbocharger pressure ratio, f_turbo_pr` — Lookup table**
`array`

The turbocharger pressure ratio, corrected for variable geometry turbocharger (VGT) speed, is a lookup table that is a function of the standard air mass flow and corrected turbocharger speed, $Pr_{turbo} = f(\dot{m}_{airstd}, N_{vgtcorr})$, where:

- $Pr_{turbo}$ is the turbocharger pressure ratio, corrected for VGT speed.
- $\dot{m}_{airstd}$ is the standard air mass flow, in g/s.
- $N_{vgtcorr}$ is the corrected turbocharger speed, in rpm/K^(1/2).




**`Turbocharger pressure ratio standard flow breakpoints, f_turbo_pr_stdflow_bpt` — Breakpoints**
`vector`

Turbocharger pressure ratio standard flow breakpoints, in g/s.

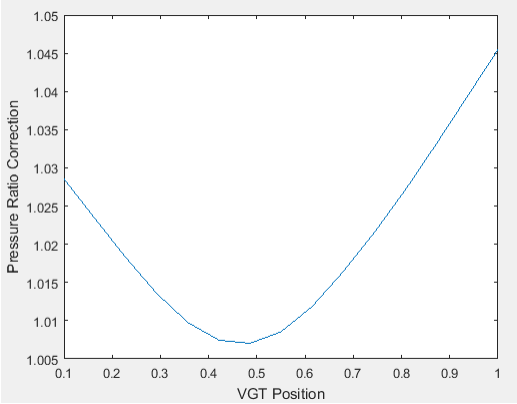
**`Turbocharger pressure ratio corrected speed breakpoints, f_turbo_pr_corrspd_bpt` — Breakpoints**
`vector`

Turbocharger pressure ratio corrected speed breakpoints, in rpm/K^(1/2).

**`Turbocharger pressure ratio VGT position correction, f_turbo_pr_vgtposcorr` — Lookup table**
`array`

The variable geometry turbocharger pressure ratio correction is a function of the rack position, $Pr_{vgtcorr} = f(VGT_{pos})$, where:

- $Pr_{vgtcorr}$ is the turbocharger pressure ratio correction.
- $VGT_{pos}$ is the variable geometry turbocharger (VGT) rack position.




**`Turbocharger pressure ratio VGT position correction breakpoints, f_turbo_pr_vgtposcorr_bpt` — Breakpoints**
`vector`

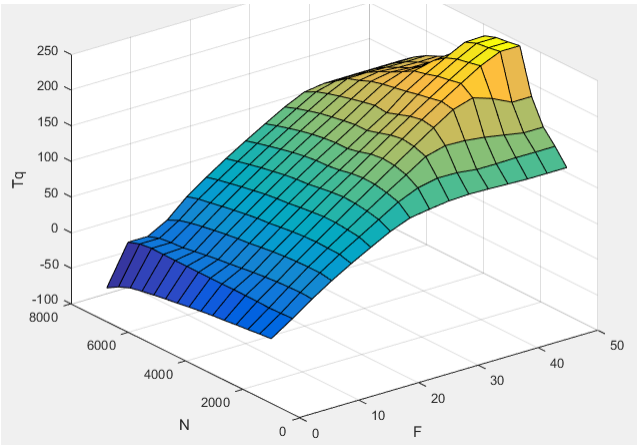
Turbocharger pressure ratio VGT position correction breakpoints, dimensionless.

#### Torque - Simple Torque Lookup

**`Torque table, f_tq_nf` — Lookup table**
`array`

For the simple torque lookup table model, the CI engine uses a lookup table is a function of engine speed and injected fuel mass, $T_{brake} = f_{Tnf}(F, N)$, where:

- $Tq = T_{brake}$ is engine brake torque after accounting for engine mechanical and pumping friction effects, in N·m.
- $F$ is injected fuel mass, in mg per injection.
- $N$ is engine speed, in rpm.




**Dependencies**

To enable this parameter, for **Torque model**, select `Simple Torque Lookup`.

**`Torque table fuel mass per injection breakpoints, f_tq_nf_f_bpt` — Breakpoints**
`[0 3.5714 7.1429 10.7143 14.2857 17.8571 21.4286 25 28.5714 32.1429 35.7143 39.2857 42.8571 46.4286 50]` (default) | `vector`

Torque table fuel mass per injection breakpoints, in mg per injection.

**Dependencies**

To enable this parameter, for **Torque model**, select `Simple Torque Lookup`.

**`Torque table speed breakpoints, f_tq_nf_n_bpt` — Breakpoints**
`[1000 1410.7143 1821.4286 2232.1429 2642.8571 3053.5714 3464.2857 3875 4285.7143 4696.4286 5107.1429 5517.8571 5928.5714 6339.2857 6750]` (default) | `vector`

Engine speed breakpoints, in rpm.

**Dependencies**

To enable this parameter, for **Torque model**, select `Simple Torque Lookup`.

**Torque - Torque Structure**

**`Fuel mass per injection breakpoints, f_tqs_f_bpt` — Breakpoints**
`vector`

Fuel mass per injection breakpoints, in mg per injection.

**Dependencies**

To enable this parameter, for **Torque model**, select `Torque Structure`.

**`Engine speed breakpoints, f_tqs_n_bpt` — Breakpoints**
`[500 750 1000 1250 1500 1750 2000 2250 2500 2750 3000 3250 3500 3750 4000]` (default) | `vector`

Engine speed breakpoints, in rpm.

**Dependencies**

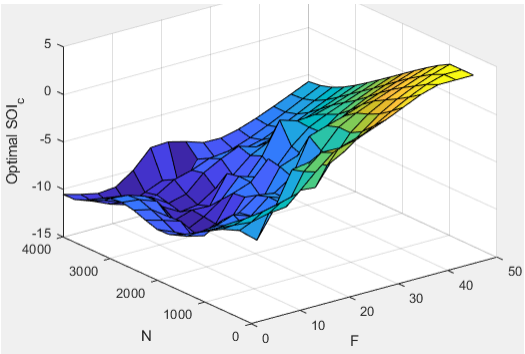
To enable this parameter, for **Torque model**, select `Torque Structure`.

### `Optimal main start of injection timing, f_tqs_mainsoi` — Optimal MAINSOI

`array`

The optimal main start of injection (SOI) timing lookup table, $f_{SOIc}$, is a function of the engine speed and injected fuel mass, $SOI_c = f_{SOIc}(F,N)$, where:

- $SOI_c$ is optimal SOI timing, in degATDC.
- $F$ is compression stroke injected fuel mass, in mg per injection.
- $N$ is engine speed, in rpm.




**Dependencies**

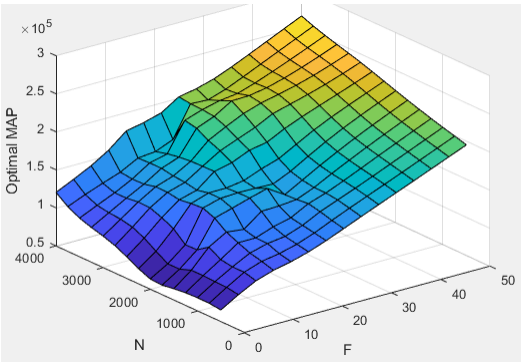
To enable this parameter, for **Torque model**, select `Torque Structure`.

### `Optimal intake manifold gas pressure, f_tqs_map` — Optimal intake MAP

`array`

The optimal intake manifold gas pressure lookup table, $f_{MAP}$, is a function of the engine speed and injected fuel mass, $MAP = f_{MAP}(F,N)$, where:

- $MAP$ is optimal intake manifold gas pressure, in Pa.
- $F$ is compression stroke injected fuel mass, in mg per injection.
- $N$ is engine speed, in rpm.



**Dependencies**

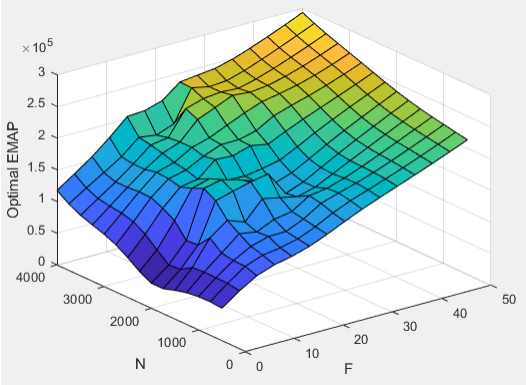
To enable this parameter, for **Torque model**, select `Torque Structure`.

### `Optimal exhaust manifold gas pressure, f_tqs_emap` — Optimal exhaust MAP

`array`

The optimal exhaust manifold gas pressure lookup table, $f_{EMAP}$, is a function of the engine speed and injected fuel mass, $EMAP = f_{EMAP}(F,N)$, where:

- $EMAP$ is optimal exhaust manifold gas pressure, in Pa.
- $F$ is compression stroke injected fuel mass, in mg per injection.
- $N$ is engine speed, in rpm.




**Dependencies**

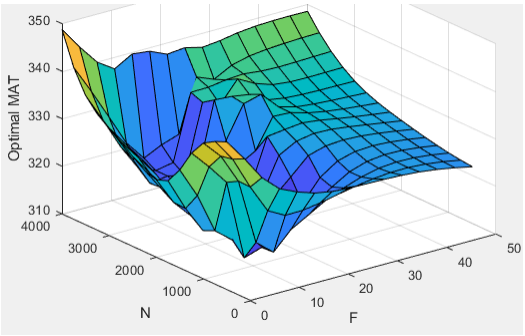
To enable this parameter, for **Torque model**, select `Torque Structure`.

### `Optimal intake manifold gas temperature, f_tqs_mat` — Optimal intake MAT

`array`

The optimal intake manifold gas temperature lookup table, $f_{MAT}$, is a function of the engine speed and injected fuel mass, $MAT = f_{MAT}(F,N)$, where:

- $MAT$ is optimal intake manifold gas temperature, in K.
- $F$ is compression stroke injected fuel mass, in mg per injection.
- $N$ is engine speed, in rpm.



**Dependencies**

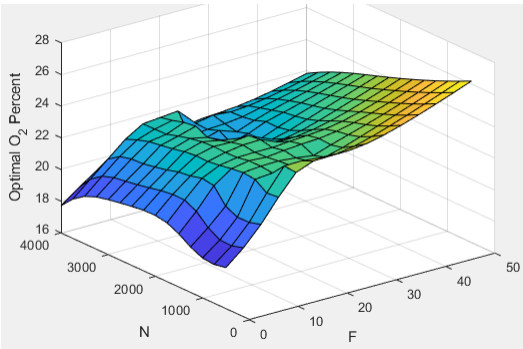
To enable this parameter, for **Torque model**, select `Torque Structure`.

### `Optimal intake gas oxygen percent, f_tqs_o2pct` — Optimal intake gas oxygen

`array`

The optimal intake gas oxygen percent lookup table, $f_{O2}$, is a function of the engine speed and injected fuel mass, $O2PCT = f_{O2}(F,N)$, where:

- *O2PCT* is optimal intake gas oxygen, in percent.
- *F* is compression stroke injected fuel mass, in mg per injection.
- *N* is engine speed, in rpm.




**Dependencies**

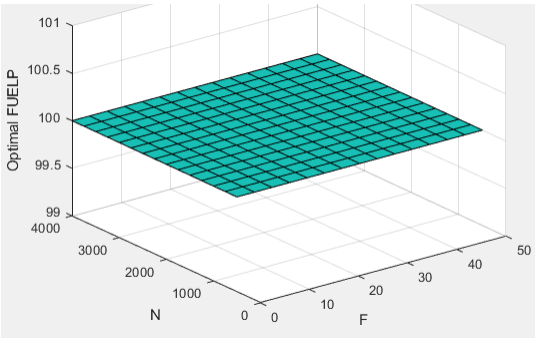
To enable this parameter, for **Torque model**, select `Torque Structure`.

### `Optimal fuel rail pressure, f_tqs_fuelpress` — Optimal fuel rail pressure

`array`

The optimal fuel rail pressure lookup table, $f_{fuelp}$, is a function of the engine speed and injected fuel mass, $FUELP = f_{fuelp}(F,N)$, where:

- *FUELP* is optimal fuel rail pressure, in MPa.
- *F* is compression stroke injected fuel mass, in mg per injection.
- *N* is engine speed, in rpm.



**Dependencies**

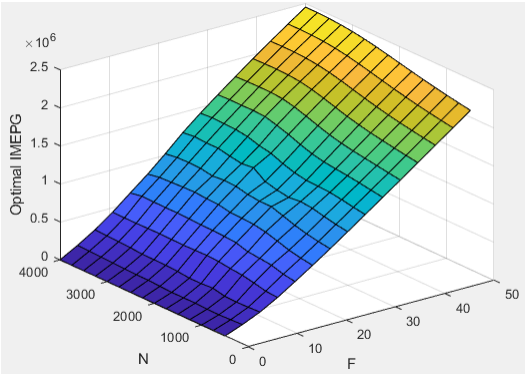
To enable this parameter, for **Torque model**, select `Torque Structure`.

### `Optimal gross indicated mean effective pressure, f_tqs_imepg` — Optimal mean effective pressure

`array`

The optimal gross indicated mean effective pressure lookup table, $f_{imepg}$, is a function of the engine speed and injected fuel mass, $IMEPG = f_{imepg}(F,N)$, where:

- *IMEPG* is optimal gross indicated mean effective pressure, in Pa.
- *F* is compression stroke injected fuel mass, in mg per injection.
- *N* is engine speed, in rpm.




**Dependencies**

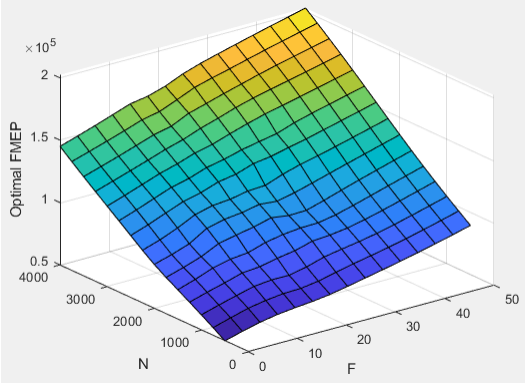
To enable this parameter, for **Torque model**, select `Torque Structure`.

### `Optimal friction mean effective pressure, f_tqs_fmep` — Optimal friction mean effective pressure

`array`

The optimal friction mean effective pressure lookup table, $f_{fmep}$, is a function of the engine speed and injected fuel mass, $FMEP = f_{fmep}(F,N)$, where:

- *FMEP* is optimal friction mean effective pressure, in Pa.
- *F* is compression stroke injected fuel mass, in mg per injection.
- *N* is engine speed, in rpm.




**Dependencies**

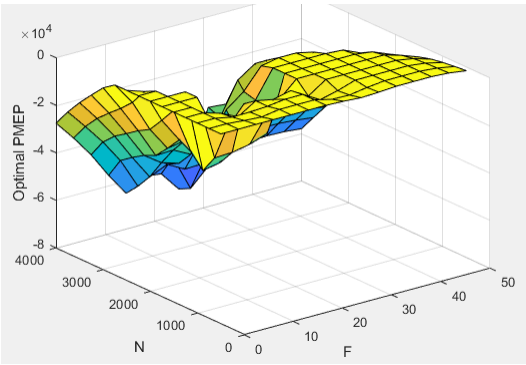
To enable this parameter, for **Torque model**, select `Torque Structure`.

### `Optimal pumping mean effective pressure, f_tqs_pmep` — Optimal pumping mean effective pressure

`array`

The optimal pumping mean effective pressure lookup table, $f_{pmep}$, is a function of the engine speed and injected fuel mass, $PMEP = f_{pmep}(F,N)$, where:

- $PMEP$ is optimal pumping mean effective pressure, in Pa.
- $F$ is compression stroke injected fuel mass, in mg per injection.
- $N$ is engine speed, in rpm.




**Dependencies**

To enable this parameter, for **Torque model**, select `Torque Structure`.

### `Friction multiplier as a function of temperature, f_tqs_fric_temp_mod` — Friction multiplier

`array`

Friction multiplier as a function of temperature, dimensionless.

**Dependencies**

To enable this parameter, for **Torque model**, select `Torque Structure`.

**`Friction multiplier temperature breakpoints, f_tqs_fric_temp_bpt` — Breakpoints**
`vector`

Friction multiplier temperature breakpoints, in K.

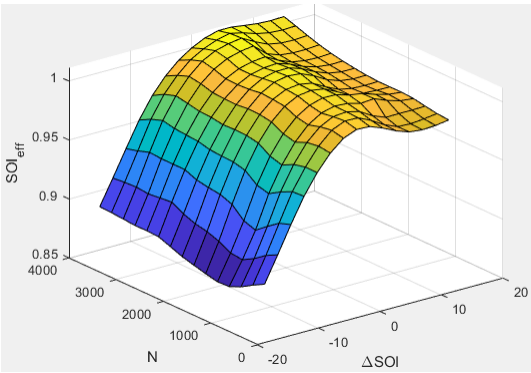
**Dependencies**

To enable this parameter, for **Torque model**, select `Torque Structure`.

**`Main start of injection timing efficiency multiplier, f_tqs_mainsoi_eff` — MAINSOI efficiency multiplier**
`array`

The main start of injection (SOI) timing efficiency multiplier lookup table, $f_{SOIeff}$, is a function of the engine speed and main SOI timing relative to optimal timing, $SOI_{eff} = f_{SOIeff}(\Delta SOI,N)$, where:

- $SOI_{eff}$ is main SOI timing efficiency multiplier, dimensionless.
- $\Delta SOI$ is main SOI timing relative to optimal timing, in degBTDC.
- $N$ is engine speed, in rpm.




**Dependencies**

To enable this parameter, for **Torque model**, select `Torque Structure`.

**`Main start of injection timing relative to optimal timing breakpoints, f_tqs_mainsoi_delta_bpt` — Breakpoints**
`vector`

Main start of injection timing relative to optimal timing breakpoints, in degBTDC.

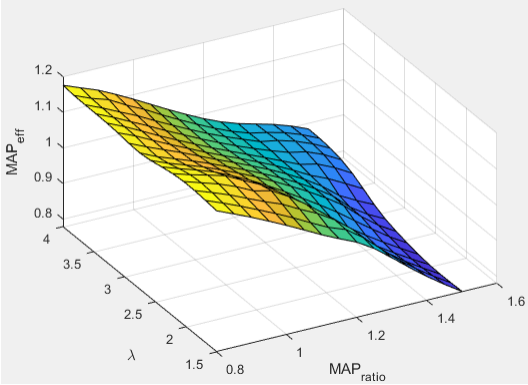
**Dependencies**

To enable this parameter, for **Torque model**, select `Torque Structure`.

**`Intake manifold gas pressure efficiency multiplier, f_tqs_map_eff` — Intake pressure efficiency multiplier**
`array`

The intake manifold gas pressure efficiency multiplier lookup table, $f_{MAPeff}$, is a function of the intake manifold gas pressure ratio relative to optimal pressure ratio and lambda, $MAP_{eff} = f_{MAPeff}(MAP_{ratio},\lambda)$, where:

- $MAP_{eff}$ is intake manifold gas pressure efficiency multiplier, dimensionless.
- $MAP_{ratio}$ is intake manifold gas pressure ratio relative to optimal pressure ratio, dimensionless.
- $\lambda$ is intake manifold gas lambda, dimensionless.




**Dependencies**

To enable this parameter, for **Torque model**, select `Torque Structure`.

**`Intake manifold gas pressure ratio relative to optimal pressure ratio breakpoints, f_tqs_map_ratio_bpt` — Breakpoints**
`[0.8;0.85;0.9;0.95;1;1.05;1.1;1.15;1.2;1.25;1.3;1.35;1.4;1.45;1.5]` (default) | `vector`

Intake manifold gas pressure ratio relative to optimal pressure ratio breakpoints, dimensionless.

**Dependencies**

To enable this parameter, for **Torque model**, select `Torque Structure`.

**`Intake manifold gas lambda breakpoints, f_tqs_lambda_bpt` — Breakpoints**
`[1.5 1.678571428571429 1.857142857142857 2.035714285714286 2.214285714285714 2.392857142857143 2.571428571428571 2.75 2.928571428571429 3.107142857142857 3.285714285714286 3.464285714285714 3.642857142857143 3.821428571428572 4]` (default) | `vector`

Intake manifold gas lambda breakpoints, dimensionless.

**Dependencies**

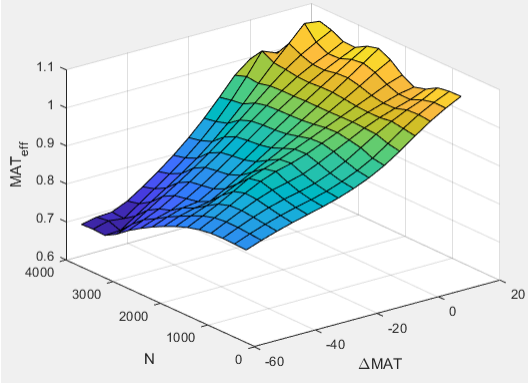
To enable this parameter, for **Torque model**, select `Torque Structure`.

**`Intake manifold gas temperature efficiency multiplier, f_tqs_mat_eff` — Intake temperature efficiency multiplier**
`array`

The intake manifold gas temperature efficiency multiplier lookup table, $f_{MATeff}$, is a function of the engine speed and intake manifold gas temperature relative to optimal temperature, $MAT_{eff} = f_{MATeff}(\Delta MAT,N)$, where:

- $MAT_{eff}$ is intake manifold gas temperature efficiency multiplier, dimensionless.
- $\Delta MAT$ is intake manifold gas temperature relative to optimal temperature, in K.

- $N$ is engine speed, in rpm.




**Dependencies**

To enable this parameter, for **Torque model**, select `Torque Structure`.

**`Intake manifold gas temperature relative to optimal gas temperature breakpoints, f_tqs_mat_delta_bpt` — Breakpoints**
`[-55;-50;-45;-40;-35;-30;-25;-20;-15;-10;-5;0;5;10;15]` (default) | `vector`

Intake manifold gas temperature relative to optimal gas temperature breakpoints, in K.

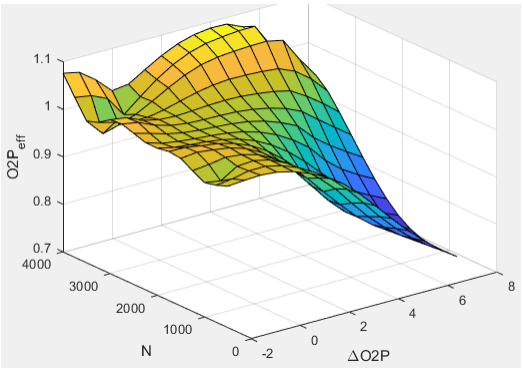
**Dependencies**

To enable this parameter, for **Torque model**, select `Torque Structure`.

**`Intake manifold gas oxygen efficiency multiplier, f_tqs_o2pct_eff` — Intake oxygen efficiency multiplier**
`array`

The intake manifold gas oxygen efficiency multiplier lookup table, $f_{O2Peff}$, is a function of the engine speed and intake manifold gas oxygen percent relative to optimal, $O2P_{eff} = f_{O2Peff}(\Delta O2P,N)$, where:

- $O2P_{eff}$ is intake manifold gas oxygen efficiency multiplier, dimensionless.
- $\Delta O2P$ is intake gas oxygen percent relative to optimal, in percent.
- $N$ is engine speed, in rpm.



#### Dependencies

To enable this parameter, for **Torque model**, select `Torque Structure`.

### `Intake gas oxygen percent relative to optimal breakpoints, f_tqs_o2pct_delta_bpt` — Breakpoints

`vector`

Intake gas oxygen percent relative to optimal breakpoints, in percent.

#### Dependencies

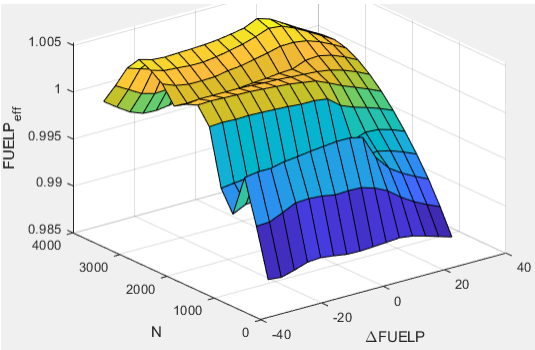
To enable this parameter, for **Torque model**, select `Torque Structure`.

### `Fuel rail pressure efficiency multiplier, f_tqs_fuelpress_eff` — Efficiency multiplier

`array`

The fuel rail pressure efficiency multiplier lookup table, $f_{FUELPeff}$, is a function of the engine speed and fuel rail pressure relative to optimal breakpoints, $FUELP_{eff} = f_{FUELPeff}(\Delta FUELP,N)$, where:

- $FUELP_{eff}$ is fuel rail pressure efficiency multiplier, dimensionless.
- $\Delta FUELP$ is fuel rail pressure relative to optimal, in MPa.
- $N$ is engine speed, in rpm.




#### Dependencies

To enable this parameter, for **Torque model**, select `Torque Structure`.

### `Fuel rail pressure relative to optimal breakpoints, f_tqs_fuelpress_delta_bpt` — Breakpoints

`vector`

Fuel rail pressure relative to optimal breakpoints, in MPa.

#### Dependencies

To enable this parameter, for **Torque model**, select `Torque Structure`.

### `Fuel mass injection type identifier, f_tqs_f_inj_type` — Type identifier

`0` (default) | `scalar`

Fuel mass injection type identifier, dimensionless.

In the CI Core Engine and CI Controller blocks, you can represent multiple injections with the start of injection (SOI) and fuel mass inputs to the model. To specify the type of injection, use the **Fuel mass injection type identifier** parameter.

| Type of Injection | Parameter Value |
|---|---|
| Pilot | `0` |
| Main | `1` |
| Post | `2` |
| Passed | `3` |

The model considers `Passed` fuel injections and fuel injected later than a threshold to be unburned fuel. Use the **Maximum start of injection angle for burned fuel, f_tqs_f_burned_soi_limit** parameter to specify the threshold.

#### Dependencies

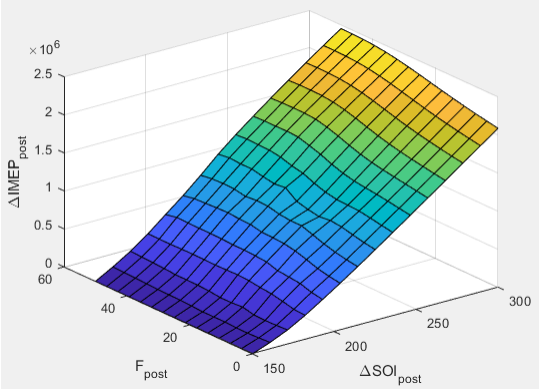
To enable this parameter, for **Torque model**, select `Torque Structure`.

### `Indicated mean effective pressure post inject correction, f_tqs_imep_post_corr` — Post inject correction

`array`

The indicated mean effective pressure post inject correction lookup table, $f_{IMEPpost}$, is a function of the engine speed and fuel rail pressure relative to optimal breakpoints, $\Delta IMEP_{post} = f_{IMEPpost}(\Delta SOI_{post}, F_{post})$, where:

- $\Delta IMEP_{post}$ is indicated mean effective pressure post inject correction, in Pa.
- $\Delta SOI_{post}$ is indicated mean effective pressure post inject start of inject timing centroid, in degATDC.
- $F_{post}$ is indicated mean effective pressure post inject mass sum, in mg per injection.




#### Dependencies

To enable this parameter, for **Torque model**, select `Torque Structure`.

### `Indicated mean effective pressure post inject mass sum breakpoints, f_tqs_f_post_sum_bpt` — Breakpoints

`[0 3.571428571428572 7.142857142857143 10.71428571428571 14.28571428571429 17.85714285714286 21.42857142857143 25 28.57142857142857 32.14285714285715`

35.71428571428572 39.28571428571428 42.85714285714285 46.42857142857143 50]
(default) | vector

Indicated mean effective pressure post inject mass sum breakpoints, in mg per injection.

#### Dependencies

To enable this parameter, for **Torque model**, select `Torque Structure`.

### Indicated mean effective pressure post inject start of inject timing centroid breakpoints, f_tqs_soi_post_cent_bpt — Breakpoints

[150 160.7142857142857 171.4285714285714 182.1428571428571 192.8571428571429
203.5714285714286 214.2857142857143 225 235.7142857142857 246.4285714285714
257.1428571428571 267.8571428571429 278.5714285714286 289.2857142857143 300]
(default) | vector

Indicated mean effective pressure post inject start of inject timing centroid breakpoints, in degATDC.

#### Dependencies

To enable this parameter, for **Torque model**, select `Torque Structure`.

### Maximum start of injection angle for burned fuel, f_tqs_f_burned_soi_limit — Maximum SOI angle for burned fuel

500 (default) | scalar

Maximum start of injection angle for burned fuel, in degATDC.

#### Dependencies

To enable this parameter, for **Torque model**, select `Torque Structure`.

#### Exhaust

### Exhaust gas specific heat at constant pressure, cp_exh — Specific heat

1005 (default) | scalar

Exhaust gas-specific heat, $Cp_{exh}$, in J/(kg·K).

#### Exhaust Temperature - Simple Torque Lookup

### Exhaust temperature table, f_t_exh — Lookup table

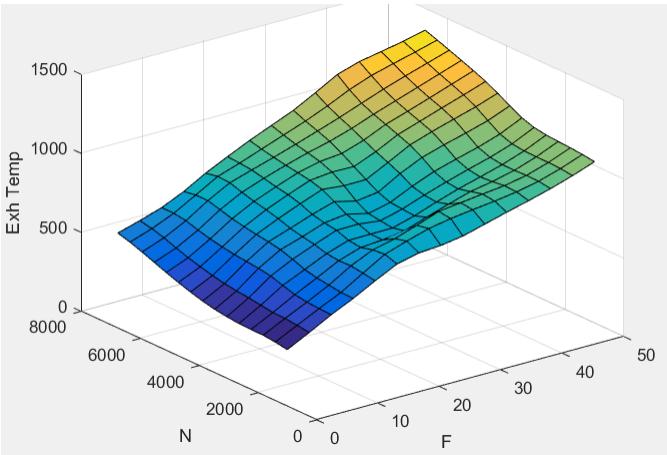
array

The lookup table for the exhaust temperature is a function of injected fuel mass and engine speed

$$T_{exh} = f_{Texh}(F, N)$$

where:

- $T_{exh}$ is exhaust temperature, in K.
- $F$ is injected fuel mass, in mg per injection.
- $N$ is engine speed, in rpm.




**Dependencies**

To enable this parameter, for **Torque model**, select `Simple Torque Lookup`.

**`Fuel mass per injection breakpoints, f_t_exh_f_bpt` — Breakpoints**
`[0 3.5714 7.1429 10.7143 14.2857 17.8571 21.4286 25 28.5714 32.1429 35.7143 39.2857 42.8571 46.4286 50]` (default) | `array`

Engine load breakpoints used for exhaust temperature lookup table, in mg per injection.

**Dependencies**

To enable this parameter, for **Torque model**, select `Simple Torque Lookup`.

**`Speed breakpoints, f_t_exh_n_bpt` — Breakpoints**
`[1000 1410.7143 1821.4286 2232.1429 2642.8571 3053.5714 3464.2857 3875 4285.7143 4696.4286 5107.1429 5517.8571 5928.5714 6339.2857 6750]` (default) | `array`

Engine speed breakpoints used for exhaust temperature lookup table, in rpm.

**Dependencies**

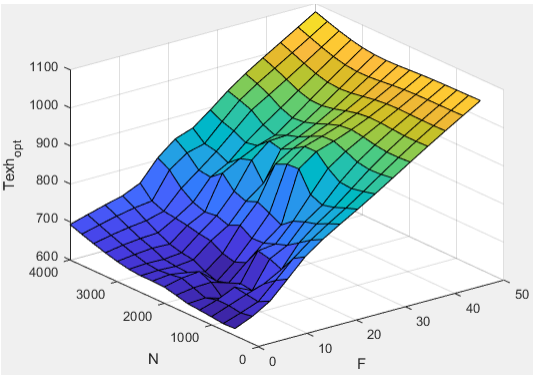
To enable this parameter, for **Torque model**, select `Simple Torque Lookup`.

**Exhaust Temperature - Torque Structure**

**`Optimal exhaust manifold gas temperature, f_tqs_exht` — Optimal exhaust manifold gas temperature**
`array`

The optimal exhaust manifold gas temperature lookup table, $f_{Texh}$, is a function of the engine speed engine speed and injected fuel mass, $Texh_{opt} = f_{Texh}(F,N)$, where:

- $Texh_{opt}$ is optimal exhaust manifold gas temperature, in K.
- $F$ is compression stroke injected fuel mass, in mg per injection.
- $N$ is engine speed, in rpm.



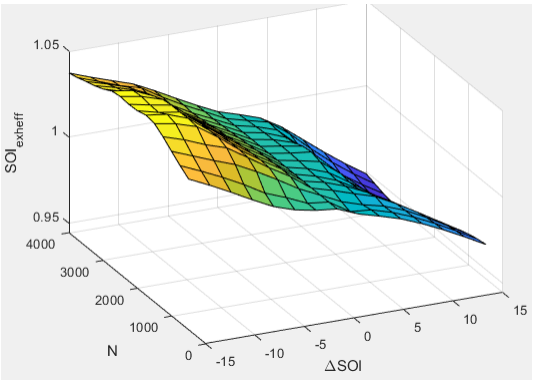

**Dependencies**

To enable this parameter, for **Torque model**, select `Torque Structure`.

**`Main start of injection timing exhaust temperature efficiency multiplier, f_tqs_exht_mainsoi_eff` — Main SOI timing efficiency multiplier**
`array`

The main start of injection (SOI) timing exhaust temperature efficiency multiplier lookup table, $f_{SOIexhteff}$, is a function of the engine speed engine speed and injected fuel mass, $SOI_{exhteff} = f_{SOIexhteff}(\Delta SOI,N)$, where:

- $SOI_{exhteff}$ is main SOI exhaust temperature efficiency multiplier, dimensionless.
- $\Delta SOI$ is main SOI timing relative to optimal timing, in degBTDC.
- $N$ is engine speed, in rpm.



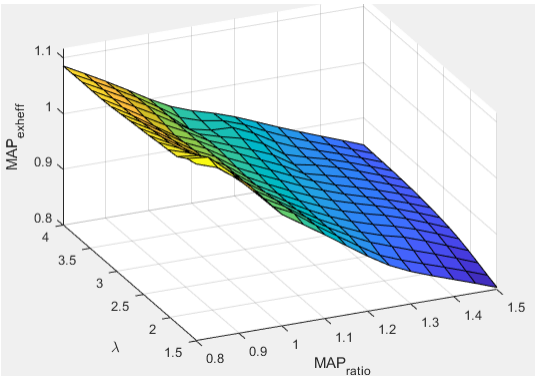

**Dependencies**

To enable this parameter, for **Torque model**, select `Torque Structure`.

**`Intake manifold gas pressure exhaust temperature efficiency multiplier, f_tqs_exht_map_eff` — Intake manifold efficiency multiplier**
`array`

The intake manifold gas pressure exhaust temperature efficiency multiplier lookup table, $f_{MAPexheff}$, is a function of the intake manifold gas pressure ratio relative to optimal pressure ratio and lambda, $MAP_{exheff} = f_{MAPexheff}(MAP_{ratio},\lambda)$, where:

- $MAP_{exheff}$ is intake manifold gas pressure exhaust temperature efficiency multiplier, dimensionless.
- $MAP_{ratio}$ is intake manifold gas pressure ratio relative to optimal pressure ratio, dimensionless.
- $\lambda$ is intake manifold gas lambda, dimensionless.



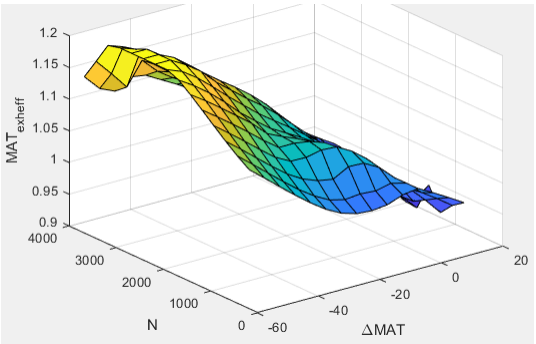

**Dependencies**

To enable this parameter, for **Torque model**, select `Torque Structure`.

**`Intake manifold gas temperature exhaust temperature efficiency multiplier, f_tqs_exht_mat_eff` — Intake manifold efficiency multiplier**
`array`

The intake manifold gas temperature exhaust temperature efficiency multiplier lookup table, $f_{MATexheff}$, is a function of the engine speed and intake manifold gas temperature relative to optimal temperature, $MAT_{exheff} = f_{MATexheff}(\Delta MAT,N)$, where:

- $MAT_{exheff}$ is intake manifold gas temperature exhaust temperature efficiency multiplier, dimensionless.
- $\Delta MAT$ is intake manifold gas temperature relative to optimal temperature, in K.
- $N$ is engine speed, in rpm.



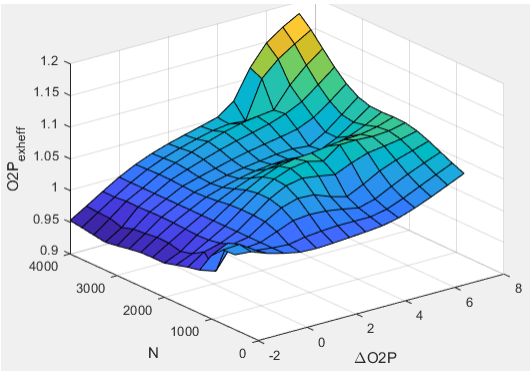

**Dependencies**

To enable this parameter, for **Torque model**, select `Torque Structure`.

**`Intake manifold gas oxygen exhaust temperature efficiency multiplier, f_tqs_exht_o2pct_eff` — Intake manifold efficiency multiplier**
`array`

The intake manifold gas oxygen exhaust temperature efficiency multiplier lookup table, $f_{O2Pexheff}$, is a function of the engine speed and intake manifold gas oxygen percent relative to optimal, $O2P_{exheff} = f_{O2Pexheff}(\Delta O2P,N)$, where:

- $O2P_{exheff}$ is intake manifold gas oxygen exhaust temperature efficiency multiplier, dimensionless.
- $\Delta O2P$ is intake gas oxygen percent relative to optimal, in percent.
- $N$ is engine speed, in rpm.




**Dependencies**

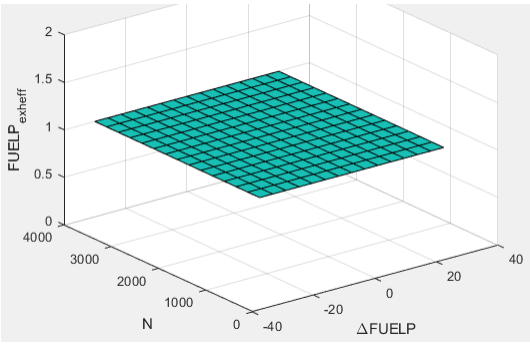
To enable this parameter, for **Torque model**, select `Torque Structure`.

### `Fuel rail pressure exhaust temperature efficiency multiplier, f_tqs_exht_fuelpress_eff` — Fuel rail pressure exhaust temperature efficiency multiplier

`array`

The fuel rail pressure efficiency exhaust temperature multiplier lookup table, $f_{FUELPexheff}$, is a function of the engine speed and fuel rail pressure relative to optimal breakpoints, $FUELP_{exheff} = f_{FUELPexheff}(\Delta FUELP,N)$, where:

- $FUELP_{exheff}$ is fuel rail pressure exhaust temperature efficiency multiplier, dimensionless.
- $\Delta FUELP$ is fuel rail pressure relative to optimal, in MPa.
- $N$ is engine speed, in rpm.




**Dependencies**

To enable this parameter, for **Torque model**, select `Torque Structure`.

**`Post-injection cylinder wall heat loss transfer coefficient, f_tqs_exht_post_inj_wall_htc` — Post-injection offset**
`0` (default) | `scalar`

Post-injection cylinder wall heat loss transfer coefficient, in W/K.

#### Dependencies

To enable this parameter, for **Torque model**, select `Torque Structure`.

## References

[1] Heywood, John B. *Internal Combustion Engine Fundamentals*. New York: McGraw-Hill, 1988.

## Extended Capabilities

### C/C++ Code Generation
Generate C and C++ code using Simulink® Coder™.

## See Also
CI Core Engine | Mapped CI Engine

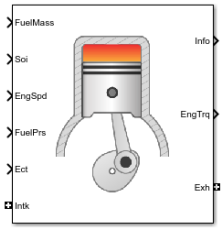
### Topics
“Engine Calibration Maps”
“Generate Mapped CI Engine from a Spreadsheet”

### Introduced in R2017a

# CI Core Engine

Compression-ignition engine from intake to exhaust port

**Library:** Powertrain Blockset / Propulsion / Combustion Engine Components / Core Engine




## Description

The CI Core Engine block implements a compression-ignition (CI) engine from intake to the exhaust port. You can use the block for hardware-in-the-loop (HIL) engine control design or vehicle-level fuel economy and performance simulations.

The CI Core Engine block calculates:

- Brake torque
- Exhaust temperature
- Air-fuel ratio (AFR)
- Fuel rail pressure
- Engine-out (EO) exhaust emissions:
  - Hydrocarbon (HC)
  - Carbon monoxide (CO)
  - Nitric oxide and nitrogen dioxide (NOx)
  - Carbon dioxide ($CO_2$)
  - Particulate matter (PM)

### Air Mass Flow

To calculate the air mass flow, the compression-ignition (CI) engine uses the “CI Engine Speed-Density Air Mass Flow Model”. The speed-density model uses the speed-density equation to calculate the engine air mass flow, relating the engine intake port mass flow to the intake manifold pressure, intake manifold temperature, and engine speed.

### Brake Torque

To calculate the engine torque, you can configure the block to use either of these torque models.

| Brake Torque Model | Description |
|---|---|
| “CI Engine Torque Structure Model” | The CI core engine torque structure model determines the engine torque by reducing the maximum engine torque potential as these engine conditions vary from nominal:<br>• Start of injection (SOI) timing<br>• Exhaust back-pressure<br>• Burned fuel mass<br>• Intake manifold gas pressure, temperature, and oxygen percentage<br>• Fuel rail pressure<br>To account for the effect of post-inject fuel on torque, the model uses a calibrated torque offset table. |
| “CI Engine Simple Torque Model” | For the simple engine torque calculation, the CI engine uses a torque lookup table map that is a function of engine speed and injected fuel mass. |

### Fuel Flow

In the CI Core Engine and CI Controller blocks, you can represent multiple injections with the start of injection (SOI) and fuel mass inputs to the model. To specify the type of injection, use the **Fuel mass injection type identifier** parameter.

| Type of Injection | Parameter Value |
|---|---|
| Pilot | `0` |
| Main | `1` |
| Post | `2` |
| Passed | `3` |

The model considers `Passed` fuel injections and fuel injected later than a threshold to be unburned fuel. Use the **Maximum start of injection angle for burned fuel, f_tqs_f_burned_soi_limit** parameter to specify the threshold.

To calculate the engine fuel mass flow, the CI Core Engine block uses fuel mass flow delivered by the injectors and the engine airflow.

$$\dot{m}_{fuel} = \frac{N \cdot N_{cyl}}{Cps\left(\frac{60s}{\min}\right)\left(\frac{1000mg}{g}\right)} \sum m_{fuel,inj}$$

To calculate the fuel economy for high-fidelity models, the block uses the volumetric fuel flow.

$$Q_{fuel} = \frac{\dot{m}_{fuel}}{\left(\frac{1000kg}{m^3}\right) Sg_{fuel}}$$

The equation uses these variables.

$\dot{m}_{fuel}$ Fuel mass flow, g/s

| | |
|---|---|
| $m_{fuel,inj}$ | Fuel mass per injection |
| $Cps$ | Crankshaft revolutions per power stroke, rev/stroke |
| $N_{cyl}$ | Number of engine cylinders |
| $N$ | Engine speed, rpm |
| $Q_{fuel}$ | Volumetric fuel flow |
| $Sg_{fuel}$ | Specific gravity of fuel |

The block uses the internal signal `FlwDir` to track the direction of the flow.

### Air-Fuel Ratio

To calculate the air-fuel (AFR) ratio, the CI Core Engine and SI Core Engine blocks implement this equation.

$$AFR = \frac{\dot{m}_{air}}{\dot{m}_{fuel}}$$

The CI Core Engine uses this equation to calculate the relative AFR.

$$\lambda = \frac{AFR}{AFR_S}$$

To calculate the exhaust gas recirculation (EGR), the blocks implement this equation. The calculation expresses the EGR as a percent of the total intake port flow.

$$EGR_{pct} = 100\frac{\dot{m}_{intk,b}}{\dot{m}_{intk}} = 100y_{intk,b}$$

The equations use these variables.

| | |
|---|---|
| $AFR$ | Air-fuel ratio |
| $AFR_s$ | Stoichiometric air-fuel ratio |
| $\dot{m}_{intk}$ | Engine air mass flow |
| $\dot{m}_{fuel}$ | Fuel mass flow |
| $\lambda$ | Relative AFR |
| $y_{intk,b}$ | Intake burned mass fraction |
| $EGR_{pct}$ | EGR percent |
| $\dot{m}_{intk,b}$ | Recirculated burned gas mass flow rate |

### Exhaust Temperature

The exhaust temperature calculation depends on the torque model. For both torque models, the block implements lookup tables.

| Torque Model | Description | Equations |
|---|---|---|
| `Simple Torque Lookup` | Exhaust temperature lookup table is a function of the injected fuel mass and engine speed. | $T_{exh} = f_{Texh}(F, N)$ |
| `Torque Structure` | The nominal exhaust temperature, $Texh_{nom}$, is a product of these exhaust temperature efficiencies:<br>• SOI timing<br>• Intake manifold gas pressure<br>• Intake manifold gas temperature<br>• Intake manifold gas oxygen percentage<br>• Fuel rail pressure<br>• Optimal temperature<br>The exhaust temperature, $Texh_{nom}$, is offset by a post temperature effect, $\Delta T_{post}$, that accounts for post and late injections during the expansion and exhaust strokes. | $T_{exhnom} = SOI_{exhteff} MAP_{exhteff} MAT_{exhteff} O2p_{exhteff} FUI$<br>$T_{exh} = T_{exhnom} + \Delta T_{post}$<br>$SOI_{exhteff} = f_{SOI_{exhteff}}(\Delta SOI, N)$<br>$MAP_{exhteff} = f_{MAP_{exhteff}}(MAP_{ratio}, \lambda)$<br>$MAT_{exhteff} = f_{MAT_{exhteff}}(\Delta MAT, N)$<br>$O2p_{exhteff} = f_{O2p_{exhteff}}(\Delta O2p, N)$<br>$Texh_{opt} = f_{Texh}(F, N)$ |

The equations use these variables.

| | |
|---|---|
| $F$ | Compression stroke injected fuel mass |
| $N$ | Engine speed |
| $Texh$ | Exhaust manifold gas temperature |
| $Texh_{opt}$ | Optimal exhaust manifold gas temperature |
| $\Delta T_{post}$ | Post injection temperature effect |
| $Texh_{nom}$ | Nominal exhaust temperature |
| $SOI_{exhteff}$ | Main SOI exhaust temperature efficiency multiplier |
| $\Delta SOI$ | Main SOI timing relative to optimal timing |
| $MAP_{exheff}$ | Intake manifold gas pressure exhaust temperature efficiency multiplier |
| $MAP_{ratio}$ | Intake manifold gas pressure ratio relative to optimal pressure ratio |
| $\lambda$ | Intake manifold gas lambda |
| $MAT_{exheff}$ | Intake manifold gas temperature exhaust temperature efficiency multiplier |
| $\Delta MAT$ | Intake manifold gas temperature relative to optimal temperature |
| $O2P_{exheff}$ | Intake manifold gas oxygen exhaust temperature efficiency multiplier |
| $\Delta O2P$ | Intake gas oxygen percent relative to optimal |
| $FUELP_{exheff}$ | Fuel rail pressure exhaust temperature efficiency multiplier |
| $\Delta FUELP$ | Fuel rail pressure relative to optimal |

### EO Exhaust Emissions

The block calculates these engine-out (EO) exhaust emissions:

- Hydrocarbon (HC)
- Carbon monoxide (CO)
- Nitric oxide and nitrogen dioxide (NOx)
- Carbon dioxide ($CO_2$)
- Particulate matter (PM)

The exhaust temperature determines the specific enthalpy.

$$h_{exh} = Cp_{exh}T_{exh}$$

The exhaust mass flow rate is the sum of the intake port air mass flow and the fuel mass flow.

$$\dot{m}_{exh} = \dot{m}_{intake} + \dot{m}_{fuel}$$

To calculate the exhaust emissions, the block multiplies the emission mass fraction by the exhaust mass flow rate. To determine the emission mass fractions, the block uses lookup tables that are functions of the engine torque and speed.

$$y_{exh,i} = f_{i_frac}(T_{brake}, N)$$
$$\dot{m}_{exh,i} = \dot{m}_{exh}y_{exh,i}$$

The fraction of air and fuel entering the intake port, injected fuel, and stoichiometric AFR determine the air mass fraction that exits the exhaust.

$$y_{exh,air} = \max\left[y_{in,air} - \frac{\dot{m}_{fuel} + y_{in,fuel}\dot{m}_{intake}}{\dot{m}_{fuel} + \dot{m}_{intake}}AFR_s\right]$$

If the engine is operating at the stoichiometric or fuel rich AFR, no air exits the exhaust. Unburned hydrocarbons and burned gas comprise the remainder of the exhaust gas. This equation determines the exhaust burned gas mass fraction.

$$y_{exh,b} = \max[(1 - y_{exh,air} - y_{exh,HC}), 0]$$

The equations use these variables.

| | |
|---|---|
| $T_{exh}$ | Engine exhaust temperature |
| $h_{exh}$ | Exhaust manifold inlet-specific enthalpy |
| $Cp_{exh}$ | Exhaust gas specific heat |
| $\dot{m}_{intk}$ | Intake port air mass flow rate |
| $\dot{m}_{fuel}$ | Fuel mass flow rate |
| $\dot{m}_{exh}$ | Exhaust mass flow rate |
| $y_{in,fuel}$ | Intake fuel mass fraction |
| $y_{exh,i}$ | Exhaust mass fraction for i = $CO_2$, CO, HC, NOx, air, burned gas, and PM |
| $\dot{m}_{exh,i}$ | Exhaust mass flow rate for i = $CO_2$, CO, HC, NOx, air, burned gas, and PM |

| | |
|---|---|
| $T_{brake}$ | Engine brake torque |
| $N$ | Engine speed |
| $y_{exh,air}$ | Exhaust air mass fraction |
| $y_{exh,b}$ | Exhaust air burned mass fraction |

### Power Accounting

For the power accounting, the block implements equations that depend on **Torque model**.

When you set **Torque model** to `Simple Torque Lookup`, the block implements these equations.

| Bus Signal | | | Description | Equations |
|---|---|---|---|---|
| `PwrInfo` | `PwrTrnsfrd` — Power transferred between blocks<br>• Positive signals indicate flow into block<br>• Negative signals indicate flow out of block | `PwrIntkHeatFlw` | Intake heat flow | $\dot{m}_{intk}h_{intk}$ |
| | | `PwrExhHeatFlw` | Exhaust heat flow | $-\dot{m}_{exh}h_{exh}$ |
| | | `PwrCrkshft` | Crankshaft power | $-T_{brake}\omega$ |
| | `PwrNotTrnsfrd` — Power crossing the block boundary, but not transferred<br>• Positive signals indicate an input<br>• Negative signals indicate a loss | `PwrFuel` | Fuel input power | $\dot{m}_{fuel}LHV$ |
| | | `PwrLoss` | All losses | $T_{brake}\omega - \dot{m}_{fuel}LHV - \dot{m}_{intk}h_{intk} + \dot{m}_{exh}h_{exh}$ |

| Bus Signal | | | Description | Equations |
|---|---|---|---|---|
| | `PwrStored` — Stored energy rate of change<br>• Positive signals indicate an increase<br>• Negative signals indicate a decrease | *Not used* | | |

When you set **Torque model** to `Torque Structure`, the block implements these equations.

| Bus Signal | | | Description | Equations |
|---|---|---|---|---|
| `PwrInfo` | `PwrTrnsfrd` — Power transferred between blocks<br>• Positive signals indicate flow into block<br>• Negative signals indicate flow out of block | `PwrIntkHeatFlw` | Intake heat flow | $\dot{m}_{intk}h_{intk}$ |
| | | `PwrExhHeatFlw` | Exhaust heat flow | $-\dot{m}_{exh}h_{exh}$ |
| | | `PwrCrkshft` | Crankshaft power | $-T_{brake}\omega$ |
| | `PwrNotTrnsfrd` — Power crossing the block boundary, but not transferred<br>• Positive signals indicate an input<br>• Negative signals indicate a loss | `PwrFuel` | Fuel input power | $\dot{m}_{fuel}LHV$ |
| | | `PwrFricLoss` | Friction loss | $-T_{fric}\omega$ |
| | | `PwrPumpLoss` | Pumping loss | $-T_{pump}\omega$ |
| | | `PwrHeatTrnsfrLoss` | Heat transfer loss | $T_{brake}\omega - \dot{m}_{fuel}LHV - \dot{m}_{intk}h_{intk} + \dot{m}_{exh}h_{exh} + T_{fric}\omega + T_{pump}\omega$ |

| Bus Signal | | | Description | Equations |
|---|---|---|---|---|
| | `PwrStored` — Stored energy rate of change<br>• Positive signals indicate an increase<br>• Negative signals indicate a decrease | *Not used* | | |

| | |
|---|---|
| $h_{exh}$ | Exhaust manifold inlet-specific enthalpy |
| $h_{intk}$ | Intake port specific enthalpy |
| $\dot{m}_{intk}$ | Intake port air mass flow rate |
| $\dot{m}_{fuel}$ | Fuel mass flow rate |
| $\dot{m}_{exh}$ | Exhaust mass flow rate |
| $\omega$ | Engine speed |
| $T_{brake}$ | Brake torque |
| $T_{pump}$ | Engine pumping work offset to inner torque |
| $T_{fric}$ | Engine friction torque |
| $LHV$ | Fuel lower heating value |

## Ports

### Input

#### `FuelMass` — Fuel injector pulse-width

`vector`

Fuel mass per injection, $m_{fuel,inj}$, in mg per injection.

#### `Soi` — Start of fuel injection timing

`vector`

Fuel injection timing, *SOI*, in degrees crank angle after top dead center (degATDC). First vector value, `Soi(1)`, is main injection timing.

##### Dependencies

To enable this parameter, for **Torque model**, select `Torque Structure`.

#### `EngSpd` — Engine speed

`scalar`

Engine speed, *N*, in rpm.

### `FuelPrs` — Fuel rail pressure

`scalar`

Fuel rail pressure, *FUELP*, in MPa.

#### Dependencies

To enable this parameter, for **Torque model**, select `Torque Structure`.

### `Ect` — Engine cooling temperature

`scalar`

Engine cooling temperature, $T_{coolant}$, in K.

#### Dependencies

To enable this parameter, for **Torque model**, select `Torque Structure`.

### `Intk` — Intake port pressure, temperature, enthalpy, mass fractions

two-way connector port

Bus containing the upstream:

- `Prs` — Pressure, in Pa
- `Temp` — Temperature, in K
- `Enth` — Specific enthalpy, in J/kg
- `MassFrac` — Intake port mass fractions, dimensionless. Exhaust gas recirculation (EGR) mass flow at the intake port is burned gas.

  Specifically, a bus with these mass fractions:

  - `O2MassFrac` — Oxygen
  - `N2MassFrac` — Nitrogen
  - `UnbrndFuelMassFrac` — Unburned fuel
  - `CO2MassFrac` — Carbon dioxide
  - `H2OMassFrac` — Water
  - `COMassFrac` — Carbon monoxide
  - `NOMassFrac` — Nitric oxide
  - `NO2MassFrac` — Nitrogen dioxide
  - `NOxMassFrac` — Nitric oxide and nitrogen dioxide
  - `PmMassFrac` — Particulate matter
  - `AirMassFrac` — Air
  - `BrndGasMassFrac` — Burned gas

### `Exh` — Exhaust port pressure, temperature, enthalpy, mass fractions

two-way connector port

Bus containing the exhaust:

- `Prs` — Pressure, in Pa

- `Temp` — Temperature, in K
- `Enth` — Specific enthalpy, in J/kg
- `MassFrac` — Exhaust port mass fractions, dimensionless.

  Specifically, a bus with these mass fractions:

  - `O2MassFrac` — Oxygen
  - `N2MassFrac` — Nitrogen
  - `UnbrndFuelMassFrac` — Unburned fuel
  - `CO2MassFrac` — Carbon dioxide
  - `H2OMassFrac` — Water
  - `COMassFrac` — Carbon monoxide
  - `NOMassFrac` — Nitric oxide
  - `NO2MassFrac` — Nitrogen dioxide
  - `NOxMassFrac` — Nitric oxide and nitrogen dioxide
  - `PmMassFrac` — Particulate matter
  - `AirMassFrac` — Air
  - `BrndGasMassFrac` — Burned gas

**Output**

**`Info` — Bus signal**
bus

Bus signal containing these block calculations.

| Signal | Description | Variable | Units |
|---|---|---|---|
| `IntkGasMassFlw` | Engine intake air mass flow. | $\dot{m}_{air}$ | kg/s |
| `IntkAirMassFlw` | Engine intake port mass flow. | $\dot{m}_{intk}$ | kg/s |
| `NrmlzdAirChrg` | Engine load (that is, normalized cylinder air mass) corrected for final steady-state cam phase angles | $L$ | N/A |
| `Afr` | Air-fuel ratio at engine exhaust port | $AFR$ | N/A |
| `FuelMassFlw` | Fuel flow into engine | $\dot{m}_{fuel}$ | kg/s |
| `FuelVolFlw` | Volumetric fuel flow | $Q_{fuel}$ | $m^3/s$ |
| `ExhManGasTemp` | Exhaust gas temperature at exhaust manifold inlet | $T_{exh}$ | K |
| `EngTrq` | Engine brake torque | $T_{brake}$ | N·m |
| `EngSpd` | Engine speed | $N$ | rpm |

| Signal | | | Description | Variable | Units |
|---|---|---|---|---|---|
| `IntkCamPhase` | | | Intake cam phaser angle | $\varphi_{ICP}$ i | degrees crank advance |
| `ExhCamPhase` | | | Exhaust cam phaser angle | $\varphi_{ECP}$ | degrees crank retard |
| `CrkAng` | | | Engine crankshaft absolute angle | $\int_0^{(360)Cps} EngSpd \frac{180}{30} d\theta$<br><br>where $Cps$ is crankshaft revolutions per power stroke | degrees crank angle |
| `EgrPct` | | | EGR percent | $EGR_{pct}$ | N/A |
| `EoAir` | | | EO air mass flow rate | $\dot{m}_{exh}$ | kg/s |
| `EoBrndGas` | | | EO burned gas mass flow rate | $y_{exh,b}$ | kg/s |
| `EoHC` | | | EO hydrocarbon emission mass flow rate | $y_{exh,HC}$ | kg/s |
| `EoCO` | | | EO carbon monoxide emission mass flow rate | $y_{exh,CO}$ | kg/s |
| `EoNOx` | | | EO nitric oxide and nitrogen dioxide emissions mass flow rate | $y_{exh,NOx}$ | kg/s |
| `EoCO2` | | | EO carbon dioxide emission mass flow rate | $y_{exh,CO2}$ | kg/s |
| `EoPm` | | | EO particulate matter emission mass flow rate | $y_{exh,PM}$ | kg/s |
| `PwrInfo` | `PwrTrnsfrd` | `PwrIntkHeatFlw` | Intake heat flow | $\dot{m}_{intk} h_{intk}$ | W |
| | | `PwrExhHeatFlw` | Exhaust heat flow | $-\dot{m}_{exh} h_{exh}$ | W |
| | | `PwrCrkshft` | Crankshaft power | $-T_{brake}\omega$ | W |
| | `PwrNotTrnsfrd` | `PwrFuel` | Fuel input power | $\dot{m}_{fuel} LHV$ | W |
| | | `PwrLoss` | For **Torque model** set to `Simple Torque Lookup`:<br><br>All losses | $T_{brake}\omega - \dot{m}_{fuel} LHV$<br>$- \dot{m}_{intk} h_{intk} + \dot{m}_{exh} h_{exh}$ | W |
| | | `PwrFricLoss` | For **Torque model** set to `Torque Structure`:<br><br>Friction loss | $-T_{fric}\omega$ | W |

| Signal | | | Description | Variable | Units |
|---|---|---|---|---|---|
| | | PwrPumpLoss | For **Torque model** set to `Torque Structure`:<br><br>Pumping loss | $-T_{pump}\omega$ | W |
| | | PwrHeatTrnsfrLoss | For **Torque model** set to `Torque Structure`:<br><br>Heat transfer loss | $T_{brake}\omega - \dot{m}_{fuel}LHV - \dot{m}_{intk}h_{intk} + \dot{m}_{exh}h_{exh} + T_{fric}\omega + T_{pump}\omega$ | W |
| | PwrStored | *Not used* | | | |

**`EngTrq` — Engine brake torque**
`scalar`

Engine brake torque, $T_{brake}$, in N·m.

**`Intk` — Intake port mass flow rate, heat flow rate, temperature, mass fraction**
two-way connector port

Bus containing:

- `MassFlwRate` — Intake port mass flow rate, in kg/s
- `HeatFlwRate` — Intake port heat flow rate, in J/s
- `ExhManGasTemp` — Intake port temperature, in K
- `MassFrac` — Intake port mass fractions, dimensionless.

  Specifically, a bus with these mass fractions:

  - `O2MassFrac` — Oxygen
  - `N2MassFrac` — Nitrogen
  - `UnbrndFuelMassFrac` — Unburned fuel
  - `CO2MassFrac` — Carbon dioxide
  - `H2OMassFrac` — Water
  - `COMassFrac` — Carbon monoxide
  - `NOMassFrac` — Nitric oxide
  - `NO2MassFrac` — Nitrogen dioxide
  - `NOxMassFrac` — Nitric oxide and nitrogen dioxide
  - `PmMassFrac` — Particulate matter
  - `AirMassFrac` — Air
  - `BrndGasMassFrac` — Burned gas

**`Exh` — Exhaust port mass flow rate, heat flow rate, temperature, mass fraction**
two-way connector port

Bus containing:

- `MassFlwRate` — Exhaust port mass flow rate, in kg/s

- `HeatFlwRate` — Exhaust heat flow rate, in J/s
- `ExhManGasTemp` — Exhaust port temperature, in K
- `MassFrac` — Exhaust port mass fractions, dimensionless.

  Specifically, a bus with these mass fractions:

  - `O2MassFrac` — Oxygen
  - `N2MassFrac` — Nitrogen
  - `UnbrndFuelMassFrac` — Unburned fuel
  - `CO2MassFrac` — Carbon dioxide
  - `H2OMassFrac` — Water
  - `COMassFrac` — Carbon monoxide
  - `NOMassFrac` — Nitric oxide
  - `NO2MassFrac` — Nitrogen dioxide
  - `NOxMassFrac` — Nitric oxide and nitrogen dioxide
  - `PmMassFrac` — Particulate matter
  - `AirMassFrac` — Air
  - `BrndGasMassFrac` — Burned gas

## Parameters

### Block Options

#### `Torque model` — Select torque model

`Torque Structure` (default) | `Simple Torque Lookup`

To calculate the engine torque, you can configure the block to use either of these torque models.

| Brake Torque Model | Description |
|---|---|
| “CI Engine Torque Structure Model” | The CI core engine torque structure model determines the engine torque by reducing the maximum engine torque potential as these engine conditions vary from nominal:<br>• Start of injection (SOI) timing<br>• Exhaust back-pressure<br>• Burned fuel mass<br>• Intake manifold gas pressure, temperature, and oxygen percentage<br>• Fuel rail pressure<br>To account for the effect of post-inject fuel on torque, the model uses a calibrated torque offset table. |
| “CI Engine Simple Torque Model” | For the simple engine torque calculation, the CI engine uses a torque lookup table map that is a function of engine speed and injected fuel mass. |

### Air

**`Number of cylinders, NCyl` — Engine cylinders**
`4` (default) | `scalar`

Number of engine cylinders, $N_{cyl}$.

**`Air standard temperature, Pstd` — Temperature**
`293.15` (default) | `scalar`

Standard air temperature, $T_{std}$, in K.

**`Crank revolutions per power stroke, Cps` — Revolutions per stroke**
`2` (default) | `scalar`

Crankshaft revolutions per power stroke, *Cps*, in rev/stroke.

**`Total displaced volume, Vd` — Volume**
`0.0015` (default) | `scalar`

Displaced volume, $V_d$, in m^3.

**`Ideal gas constant air, Rair` — Constant**
`287.05` (default) | `scalar`

Ideal gas constant, $R_{air}$, in J/(kg·K).

**`Air standard pressure, Pstd` — Pressure**
`101325` (default) | `scalar`

Standard air pressure, $P_{std}$, in Pa.

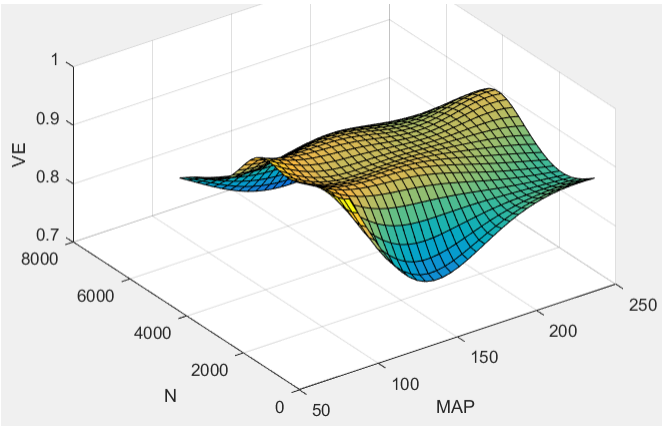
**`Speed-density volumetric efficiency, f_nv` — Lookup table**
`array`

The volumetric efficiency lookup table is a function of the intake manifold absolute pressure at intake valve closing (IVC) and engine speed

$$\eta_v = f_{\eta_v}(MAP, N)$$

where:

- $\eta_v$ is engine volumetric efficiency, dimensionless.
- *MAP* is intake manifold absolute pressure, in KPa.
- $N$ is engine speed, in rpm.




**`Speed-density intake manifold pressure breakpoints, f_nv_prs_bpt` — Breakpoints**
`vector`

Intake manifold pressure breakpoints for speed-density volumetric efficiency lookup table, in KPa.

**`Speed-density engine speed breakpoints, f_nv_n_bpt` — Breakpoints**
`vector`

Engine speed breakpoints for speed-density volumetric efficiency lookup table, in rpm.

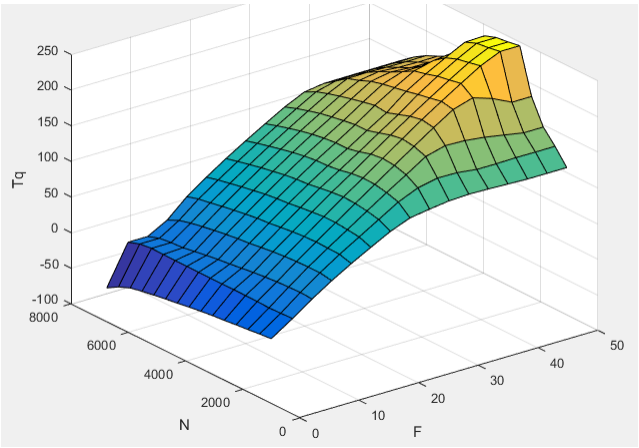
### Torque

#### Torque - Simple Torque Lookup

**`Torque table, f_tq_nf` — Lookup table**
`array`

For the simple torque lookup table model, the CI engine uses a lookup table is a function of engine speed and injected fuel mass, $T_{brake} = f_{Tnf}(F, N)$, where:

- $Tq = T_{brake}$ is engine brake torque after accounting for engine mechanical and pumping friction effects, in N·m.
- $F$ is injected fuel mass, in mg per injection.
- $N$ is engine speed, in rpm.



**Dependencies**

To enable this parameter, for **Torque model**, select `Simple Torque Lookup`.

**`Torque table fuel mass per injection breakpoints, f_tq_nf_f_bpt` — Breakpoints**
`[0 3.5714 7.1429 10.7143 14.2857 17.8571 21.4286 25 28.5714 32.1429 35.7143 39.2857 42.8571 46.4286 50]` (default) | `vector`

Torque table fuel mass per injection breakpoints, in mg per injection.

**Dependencies**

To enable this parameter, for **Torque model**, select `Simple Torque Lookup`.

**`Torque table speed breakpoints, f_tq_nf_n_bpt` — Breakpoints**
`[1000 1410.7143 1821.4286 2232.1429 2642.8571 3053.5714 3464.2857 3875 4285.7143 4696.4286 5107.1429 5517.8571 5928.5714 6339.2857 6750]` (default) | `vector`

Engine speed breakpoints, in rpm.

**Dependencies**

To enable this parameter, for **Torque model**, select `Simple Torque Lookup`.

**Torque - Torque Structure**

**`Fuel mass per injection breakpoints, f_tqs_f_bpt` — Breakpoints**
`vector`

Fuel mass per injection breakpoints, in mg per injection.

**Dependencies**

To enable this parameter, for **Torque model**, select `Torque Structure`.

**`Engine speed breakpoints, f_tqs_n_bpt` — Breakpoints**
`[500 750 1000 1250 1500 1750 2000 2250 2500 2750 3000 3250 3500 3750 4000]` (default) | `vector`

Engine speed breakpoints, in rpm.

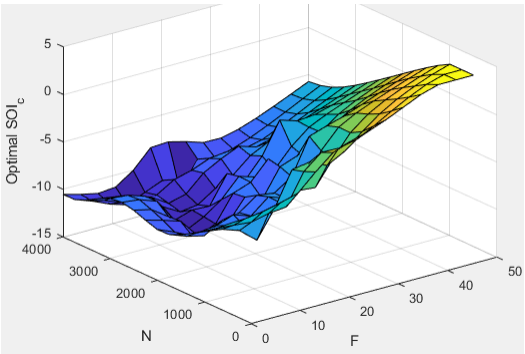
**Dependencies**

To enable this parameter, for **Torque model**, select `Torque Structure`.

**`Optimal main start of injection timing, f_tqs_mainsoi` — Optimal MAINSOI**
`array`

The optimal main start of injection (SOI) timing lookup table, $f_{SOIc}$, is a function of the engine speed and injected fuel mass, $SOI_c = f_{SOIc}(F,N)$, where:

- $SOI_c$ is optimal SOI timing, in degATDC.
- $F$ is compression stroke injected fuel mass, in mg per injection.
- $N$ is engine speed, in rpm.




**Dependencies**

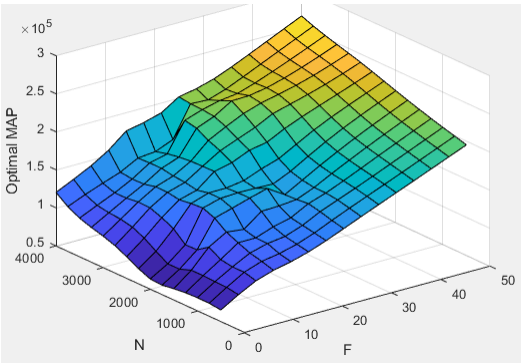
To enable this parameter, for **Torque model**, select `Torque Structure`.

### `Optimal intake manifold gas pressure, f_tqs_map` — Optimal intake MAP

`array`

The optimal intake manifold gas pressure lookup table, $f_{MAP}$, is a function of the engine speed and injected fuel mass, $MAP = f_{MAP}(F,N)$, where:

- $MAP$ is optimal intake manifold gas pressure, in Pa.
- $F$ is compression stroke injected fuel mass, in mg per injection.
- $N$ is engine speed, in rpm.




**Dependencies**

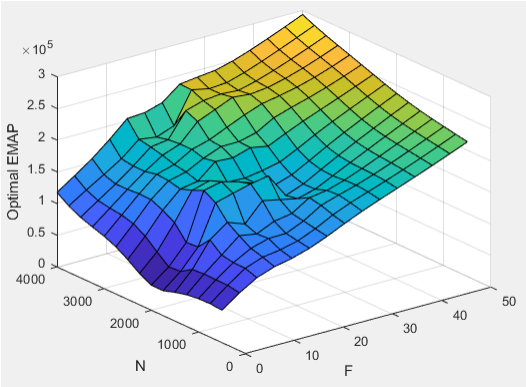
To enable this parameter, for **Torque model**, select `Torque Structure`.

### `Optimal exhaust manifold gas pressure, f_tqs_emap` — Optimal exhaust MAP

`array`

The optimal exhaust manifold gas pressure lookup table, $f_{EMAP}$, is a function of the engine speed and injected fuel mass, $EMAP = f_{EMAP}(F,N)$, where:

- $EMAP$ is optimal exhaust manifold gas pressure, in Pa.
- $F$ is compression stroke injected fuel mass, in mg per injection.
- $N$ is engine speed, in rpm.



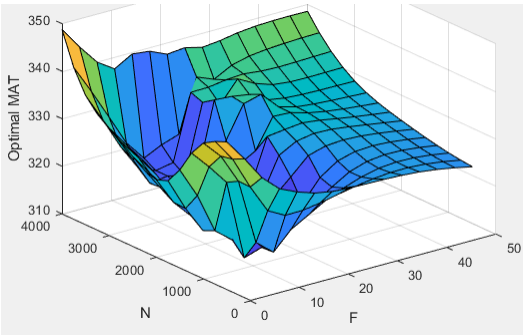

**Dependencies**

To enable this parameter, for **Torque model**, select `Torque Structure`.

**`Optimal intake manifold gas temperature, f_tqs_mat` — Optimal intake MAT**
`array`

The optimal intake manifold gas temperature lookup table, $f_{MAT}$, is a function of the engine speed and injected fuel mass, $MAT = f_{MAT}(F,N)$, where:

- $MAT$ is optimal intake manifold gas temperature, in K.
- $F$ is compression stroke injected fuel mass, in mg per injection.
- $N$ is engine speed, in rpm.



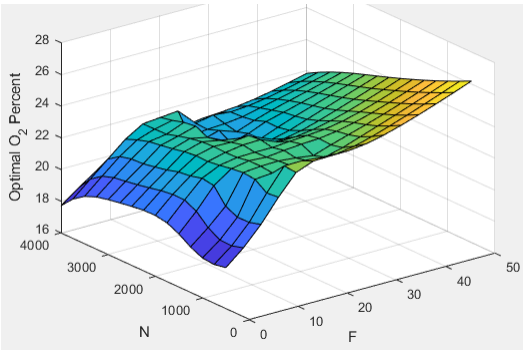

**Dependencies**

To enable this parameter, for **Torque model**, select `Torque Structure`.

**`Optimal intake gas oxygen percent, f_tqs_o2pct` — Optimal intake gas oxygen**
`array`

The optimal intake gas oxygen percent lookup table, $f_{O2}$, is a function of the engine speed and injected fuel mass, $O2PCT = f_{O2}(F,N)$, where:

- $O2PCT$ is optimal intake gas oxygen, in percent.
- $F$ is compression stroke injected fuel mass, in mg per injection.
- $N$ is engine speed, in rpm.




**Dependencies**

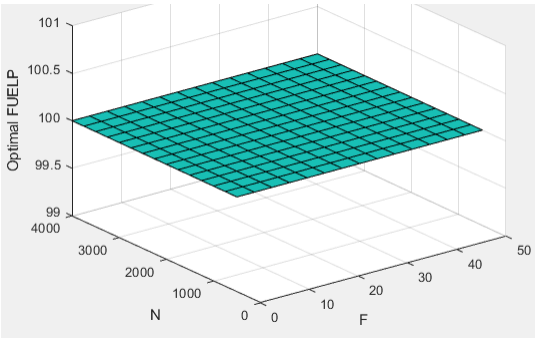
To enable this parameter, for **Torque model**, select `Torque Structure`.

### `Optimal fuel rail pressure, f_tqs_fuelpress` — Optimal fuel rail pressure

`array`

The optimal fuel rail pressure lookup table, $f_{fuelp}$, is a function of the engine speed and injected fuel mass, $FUELP = f_{fuelp}(F,N)$, where:

- $FUELP$ is optimal fuel rail pressure, in MPa.
- $F$ is compression stroke injected fuel mass, in mg per injection.
- $N$ is engine speed, in rpm.




**Dependencies**

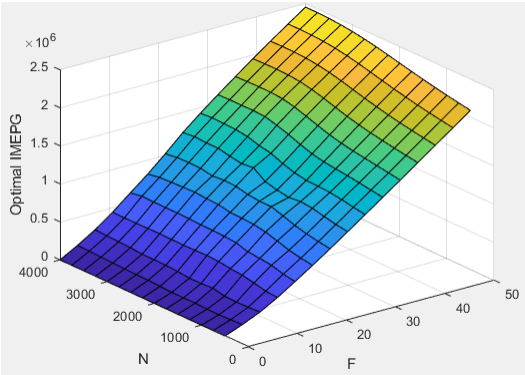
To enable this parameter, for **Torque model**, select `Torque Structure`.

### `Optimal gross indicated mean effective pressure, f_tqs_imepg` — Optimal mean effective pressure

`array`

The optimal gross indicated mean effective pressure lookup table, $f_{imepg}$, is a function of the engine speed and injected fuel mass, $IMEPG = f_{imepg}(F,N)$, where:

- $IMEPG$ is optimal gross indicated mean effective pressure, in Pa.
- $F$ is compression stroke injected fuel mass, in mg per injection.
- $N$ is engine speed, in rpm.



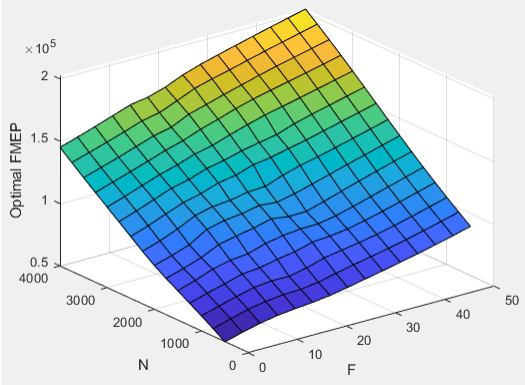

**Dependencies**

To enable this parameter, for **Torque model**, select `Torque Structure`.

**`Optimal friction mean effective pressure, f_tqs_fmep` — Optimal friction mean effective pressure**
`array`

The optimal friction mean effective pressure lookup table, $f_{fmep}$, is a function of the engine speed and injected fuel mass, $FMEP = f_{fmep}(F,N)$, where:

- $FMEP$ is optimal friction mean effective pressure, in Pa.
- $F$ is compression stroke injected fuel mass, in mg per injection.
- $N$ is engine speed, in rpm.




**Dependencies**

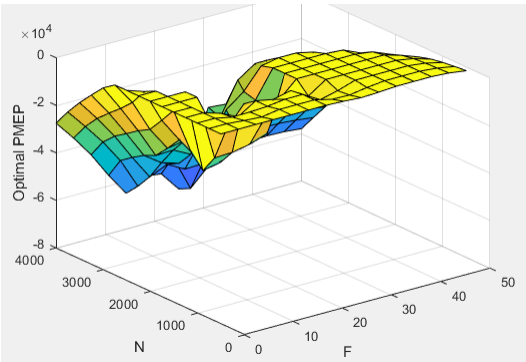
To enable this parameter, for **Torque model**, select `Torque Structure`.

**`Optimal pumping mean effective pressure, f_tqs_pmep` — Optimal pumping mean effective pressure**
`array`

The optimal pumping mean effective pressure lookup table, $f_{pmep}$, is a function of the engine speed and injected fuel mass, $PMEP = f_{pmep}(F,N)$, where:

- $PMEP$ is optimal pumping mean effective pressure, in Pa.

- $F$ is compression stroke injected fuel mass, in mg per injection.
- $N$ is engine speed, in rpm.




**Dependencies**

To enable this parameter, for **Torque model**, select `Torque Structure`.

**`Friction multiplier as a function of temperature, f_tqs_fric_temp_mod` — Friction multiplier**
`array`

Friction multiplier as a function of temperature, dimensionless.

**Dependencies**

To enable this parameter, for **Torque model**, select `Torque Structure`.

**`Friction multiplier temperature breakpoints, f_tqs_fric_temp_bpt` — Breakpoints**
`vector`

Friction multiplier temperature breakpoints, in K.

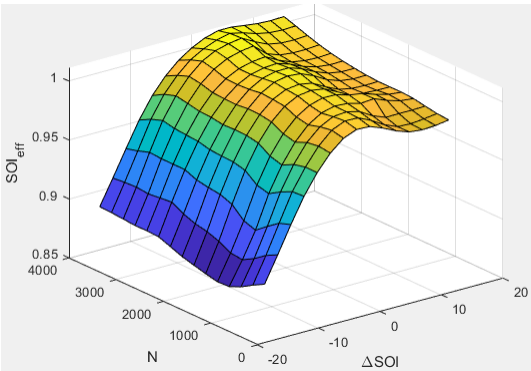
**Dependencies**

To enable this parameter, for **Torque model**, select `Torque Structure`.

**`Main start of injection timing efficiency multiplier, f_tqs_mainsoi_eff` — MAINSOI efficiency multiplier**
`array`

The main start of injection (SOI) timing efficiency multiplier lookup table, $f_{SOIeff}$, is a function of the engine speed and main SOI timing relative to optimal timing, $SOI_{eff} = f_{SOIeff}(\Delta SOI,N)$, where:

- $SOI_{eff}$ is main SOI timing efficiency multiplier, dimensionless.
- $\Delta SOI$ is main SOI timing relative to optimal timing, in degBTDC.
- $N$ is engine speed, in rpm.




**Dependencies**

To enable this parameter, for **Torque model**, select `Torque Structure`.

**`Main start of injection timing relative to optimal timing breakpoints, f_tqs_mainsoi_delta_bpt` — Breakpoints**
`vector`

Main start of injection timing relative to optimal timing breakpoints, in degBTDC.

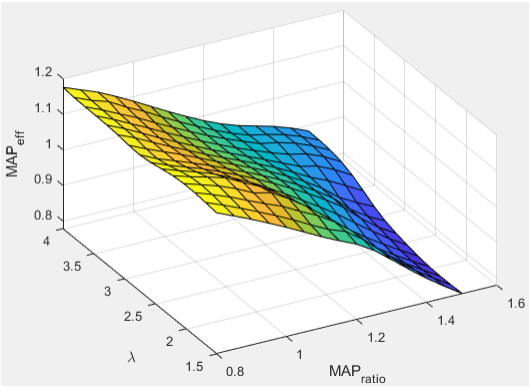
**Dependencies**

To enable this parameter, for **Torque model**, select `Torque Structure`.

**`Intake manifold gas pressure efficiency multiplier, f_tqs_map_eff` — Intake pressure efficiency multiplier**
`array`

The intake manifold gas pressure efficiency multiplier lookup table, $f_{MAPeff}$, is a function of the intake manifold gas pressure ratio relative to optimal pressure ratio and lambda, $MAP_{eff} = f_{MAPeff}(MAP_{ratio}, \lambda)$, where:

- $MAP_{eff}$ is intake manifold gas pressure efficiency multiplier, dimensionless.
- $MAP_{ratio}$ is intake manifold gas pressure ratio relative to optimal pressure ratio, dimensionless.
- $\lambda$ is intake manifold gas lambda, dimensionless.




**Dependencies**

To enable this parameter, for **Torque model**, select `Torque Structure`.

**`Intake manifold gas pressure ratio relative to optimal pressure ratio breakpoints, f_tqs_map_ratio_bpt` — Breakpoints**
`[0.8;0.85;0.9;0.95;1;1.05;1.1;1.15;1.2;1.25;1.3;1.35;1.4;1.45;1.5]` (default) | `vector`

Intake manifold gas pressure ratio relative to optimal pressure ratio breakpoints, dimensionless.

**Dependencies**

To enable this parameter, for **Torque model**, select `Torque Structure`.

**`Intake manifold gas lambda breakpoints, f_tqs_lambda_bpt` — Breakpoints**
`[1.5 1.678571428571429 1.857142857142857 2.035714285714286 2.214285714285714 2.392857142857143 2.571428571428571 2.75 2.928571428571429 3.107142857142857 3.285714285714286 3.464285714285714 3.642857142857143 3.821428571428572 4]` (default) | `vector`

Intake manifold gas lambda breakpoints, dimensionless.

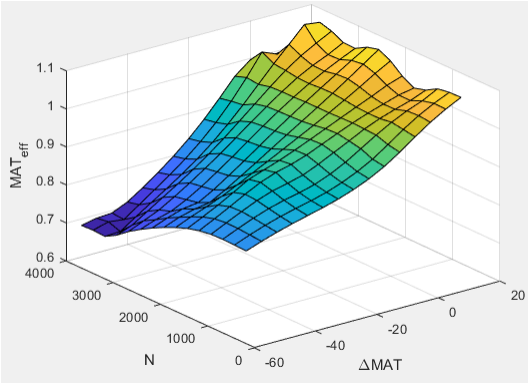
**Dependencies**

To enable this parameter, for **Torque model**, select `Torque Structure`.

**`Intake manifold gas temperature efficiency multiplier, f_tqs_mat_eff` — Intake temperature efficiency multiplier**
`array`

The intake manifold gas temperature efficiency multiplier lookup table, $f_{MATeff}$, is a function of the engine speed and intake manifold gas temperature relative to optimal temperature, $MAT_{eff} = f_{MATeff}(\Delta MAT,N)$, where:

- $MAT_{eff}$ is intake manifold gas temperature efficiency multiplier, dimensionless.
- $\Delta MAT$ is intake manifold gas temperature relative to optimal temperature, in K.
- $N$ is engine speed, in rpm.




**Dependencies**

To enable this parameter, for **Torque model**, select `Torque Structure`.

**`Intake manifold gas temperature relative to optimal gas temperature breakpoints, f_tqs_mat_delta_bpt` — Breakpoints**
`[-55;-50;-45;-40;-35;-30;-25;-20;-15;-10;-5;0;5;10;15]` (default) | `vector`

Intake manifold gas temperature relative to optimal gas temperature breakpoints, in K.

**Dependencies**

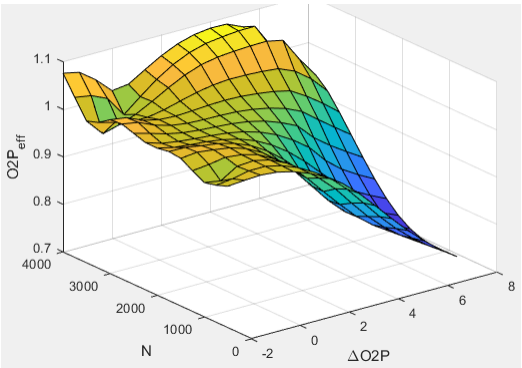
To enable this parameter, for **Torque model**, select `Torque Structure`.

### `Intake manifold gas oxygen efficiency multiplier, f_tqs_o2pct_eff` — Intake oxygen efficiency multiplier

`array`

The intake manifold gas oxygen efficiency multiplier lookup table, $f_{O2Peff}$, is a function of the engine speed and intake manifold gas oxygen percent relative to optimal, $O2P_{eff} = f_{O2Peff}(\Delta O2P,N)$, where:

- $O2P_{eff}$ is intake manifold gas oxygen efficiency multiplier, dimensionless.
- $\Delta O2P$ is intake gas oxygen percent relative to optimal, in percent.
- $N$ is engine speed, in rpm.




**Dependencies**

To enable this parameter, for **Torque model**, select `Torque Structure`.

### `Intake gas oxygen percent relative to optimal breakpoints, f_tqs_o2pct_delta_bpt` — Breakpoints

`vector`

Intake gas oxygen percent relative to optimal breakpoints, in percent.

**Dependencies**

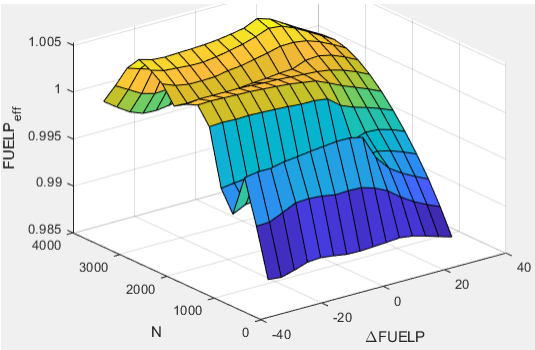
To enable this parameter, for **Torque model**, select `Torque Structure`.

### `Fuel rail pressure efficiency multiplier, f_tqs_fuelpress_eff` — Efficiency multiplier

`array`

The fuel rail pressure efficiency multiplier lookup table, $f_{FUELPeff}$, is a function of the engine speed and fuel rail pressure relative to optimal breakpoints, $FUELP_{eff} = f_{FUELPeff}(\Delta FUELP,N)$, where:

- $FUELP_{eff}$ is fuel rail pressure efficiency multiplier, dimensionless.
- $\Delta FUELP$ is fuel rail pressure relative to optimal, in MPa.
- $N$ is engine speed, in rpm.




**Dependencies**

To enable this parameter, for **Torque model**, select `Torque Structure`.

**`Fuel rail pressure relative to optimal breakpoints, f_tqs_fuelpress_delta_bpt` — Breakpoints**
`vector`

Fuel rail pressure relative to optimal breakpoints, in MPa.

**Dependencies**

To enable this parameter, for **Torque model**, select `Torque Structure`.

**`Fuel mass injection type identifier, f_tqs_f_inj_type` — Type identifier**
`0` (default) | `scalar`

Fuel mass injection type identifier, dimensionless.

In the CI Core Engine and CI Controller blocks, you can represent multiple injections with the start of injection (SOI) and fuel mass inputs to the model. To specify the type of injection, use the **Fuel mass injection type identifier** parameter.

| Type of Injection | Parameter Value |
|---|---|
| Pilot | `0` |
| Main | `1` |
| Post | `2` |
| Passed | `3` |

The model considers `Passed` fuel injections and fuel injected later than a threshold to be unburned fuel. Use the **Maximum start of injection angle for burned fuel, f_tqs_f_burned_soi_limit** parameter to specify the threshold.

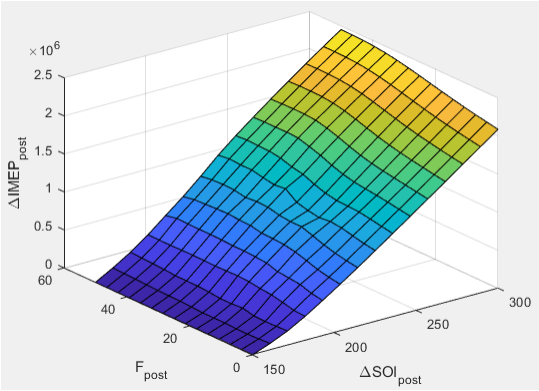
**Dependencies**

To enable this parameter, for **Torque model**, select `Torque Structure`.

**`Indicated mean effective pressure post inject correction, f_tqs_imep_post_corr` — Post inject correction**
`array`

The indicated mean effective pressure post inject correction lookup table, $f_{IMEPpost}$, is a function of the engine speed and fuel rail pressure relative to optimal breakpoints, $\Delta IMEP_{post} = f_{IMEPpost}(\Delta SOI_{post}, F_{post})$, where:

- $\Delta IMEP_{post}$ is indicated mean effective pressure post inject correction, in Pa.
- $\Delta SOI_{post}$ is indicated mean effective pressure post inject start of inject timing centroid, in degATDC.
- $F_{post}$ is indicated mean effective pressure post inject mass sum, in mg per injection.




**Dependencies**

To enable this parameter, for **Torque model**, select `Torque Structure`.

**`Indicated mean effective pressure post inject mass sum breakpoints, f_tqs_f_post_sum_bpt` — Breakpoints**
`[0 3.571428571428572 7.142857142857143 10.71428571428571 14.28571428571429 17.85714285714286 21.42857142857143 25 28.57142857142857 32.14285714285715 35.71428571428572 39.28571428571428 42.85714285714285 46.42857142857143 50]` (default) | `vector`

Indicated mean effective pressure post inject mass sum breakpoints, in mg per injection.

**Dependencies**

To enable this parameter, for **Torque model**, select `Torque Structure`.

**`Indicated mean effective pressure post inject start of inject timing centroid breakpoints, f_tqs_soi_post_cent_bpt` — Breakpoints**
`[150 160.7142857142857 171.4285714285714 182.1428571428571 192.8571428571429 203.5714285714286 214.2857142857143 225 235.7142857142857 246.4285714285714 257.1428571428571 267.8571428571429 278.5714285714286 289.2857142857143 300]` (default) | `vector`

Indicated mean effective pressure post inject start of inject timing centroid breakpoints, in degATDC.

**Dependencies**

To enable this parameter, for **Torque model**, select `Torque Structure`.

**`Maximum start of injection angle for burned fuel, f_tqs_f_burned_soi_limit` — Maximum SOI angle for burned fuel**
`500` (default) | `scalar`

Maximum start of injection angle for burned fuel, in degATDC.

**Dependencies**

To enable this parameter, for **Torque model**, select `Torque Structure`.

### Exhaust

#### Exhaust Temperature - Simple Torque Lookup

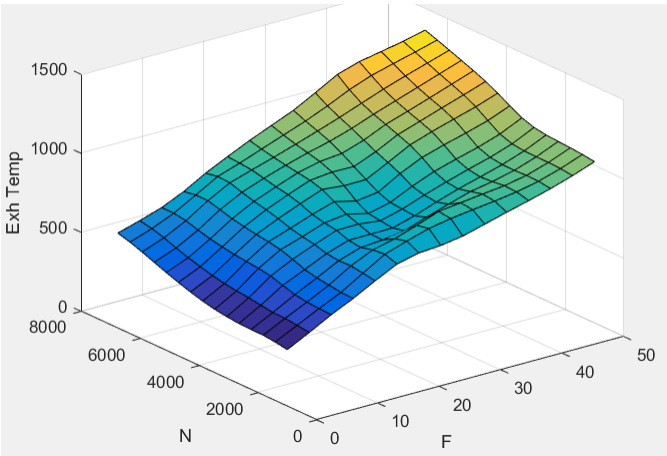
**`Exhaust temperature table, f_t_exh` — Lookup table**
`array`

The lookup table for the exhaust temperature is a function of injected fuel mass and engine speed

$$T_{exh} = f_{Texh}(F, N)$$

where:

- $T_{exh}$ is exhaust temperature, in K.
- $F$ is injected fuel mass, in mg per injection.
- $N$ is engine speed, in rpm.




**Dependencies**

To enable this parameter, for **Torque model**, select `Simple Torque Lookup`.

**`Fuel mass per injection breakpoints, f_t_exh_f_bpt` — Breakpoints**
`[0 3.5714 7.1429 10.7143 14.2857 17.8571 21.4286 25 28.5714 32.1429 35.7143 39.2857 42.8571 46.4286 50]` (default) | `array`

Engine load breakpoints used for exhaust temperature lookup table, in mg per injection.

**Dependencies**

To enable this parameter, for **Torque model**, select `Simple Torque Lookup`.

### Speed breakpoints, f_t_exh_n_bpt — Breakpoints

`[1000 1410.7143 1821.4286 2232.1429 2642.8571 3053.5714 3464.2857 3875 4285.7143 4696.4286 5107.1429 5517.8571 5928.5714 6339.2857 6750]` (default) | `array`

Engine speed breakpoints used for exhaust temperature lookup table, in rpm.

#### Dependencies

To enable this parameter, for **Torque model**, select `Simple Torque Lookup`.

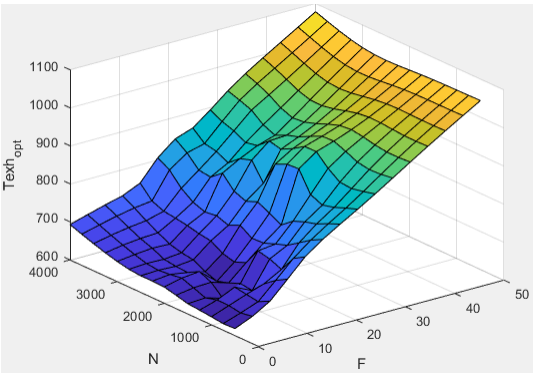
#### Exhaust Temperature - Torque Structure

### Optimal exhaust manifold gas temperature, f_tqs_exht — Optimal exhaust manifold gas temperature

`array`

The optimal exhaust manifold gas temperature lookup table, $f_{Texh}$, is a function of the engine speed engine speed and injected fuel mass, $Texh_{opt} = f_{Texh}(F,N)$, where:

- $Texh_{opt}$ is optimal exhaust manifold gas temperature, in K.
- $F$ is compression stroke injected fuel mass, in mg per injection.
- $N$ is engine speed, in rpm.




#### Dependencies

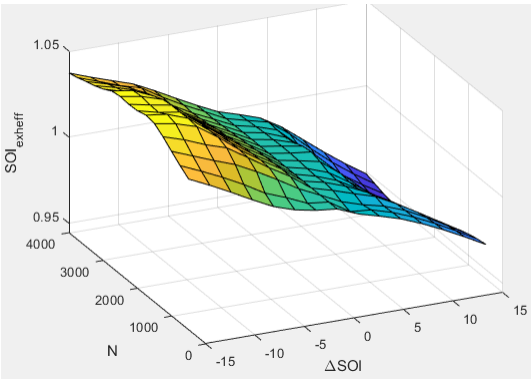
To enable this parameter, for **Torque model**, select `Torque Structure`.

### Main start of injection timing exhaust temperature efficiency multiplier, f_tqs_exht_mainsoi_eff — Main SOI timing efficiency multiplier

`array`

The main start of injection (SOI) timing exhaust temperature efficiency multiplier lookup table, $f_{SOIexhteff}$, is a function of the engine speed engine speed and injected fuel mass, $SOI_{exhteff} = f_{SOIexhteff}(\Delta SOI,N)$, where:

- $SOI_{exhteff}$ is main SOI exhaust temperature efficiency multiplier, dimensionless.
- $\Delta SOI$ is main SOI timing relative to optimal timing, in degBTDC.
- $N$ is engine speed, in rpm.



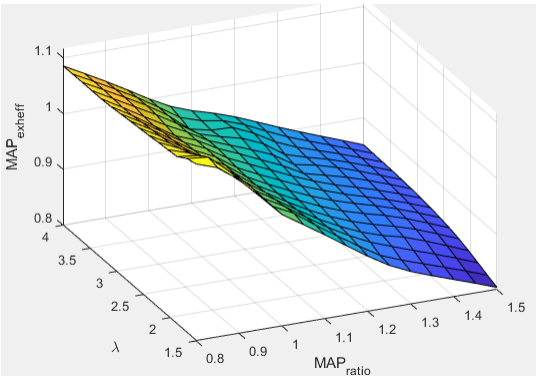

**Dependencies**

To enable this parameter, for **Torque model**, select `Torque Structure`.

**`Intake manifold gas pressure exhaust temperature efficiency multiplier, f_tqs_exht_map_eff` — Intake manifold efficiency multiplier**
`array`

The intake manifold gas pressure exhaust temperature efficiency multiplier lookup table, $f_{MAPexheff}$, is a function of the intake manifold gas pressure ratio relative to optimal pressure ratio and lambda, $MAP_{exheff} = f_{MAPexheff}(MAP_{ratio},\lambda)$, where:

- $MAP_{exheff}$ is intake manifold gas pressure exhaust temperature efficiency multiplier, dimensionless.
- $MAP_{ratio}$ is intake manifold gas pressure ratio relative to optimal pressure ratio, dimensionless.
- $\lambda$ is intake manifold gas lambda, dimensionless.



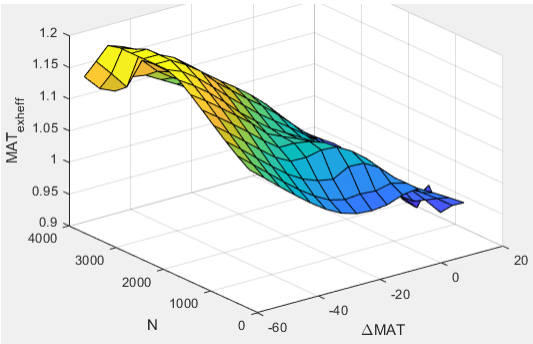

**Dependencies**

To enable this parameter, for **Torque model**, select `Torque Structure`.

**`Intake manifold gas temperature exhaust temperature efficiency multiplier, f_tqs_exht_mat_eff` — Intake manifold efficiency multiplier**
`array`

The intake manifold gas temperature exhaust temperature efficiency multiplier lookup table, $f_{MATexheff}$, is a function of the engine speed and intake manifold gas temperature relative to optimal temperature, $MAT_{exheff} = f_{MATexheff}(\Delta MAT,N)$, where:

- $MAT_{exheff}$ is intake manifold gas temperature exhaust temperature efficiency multiplier, dimensionless.
- $\Delta MAT$ is intake manifold gas temperature relative to optimal temperature, in K.
- $N$ is engine speed, in rpm.



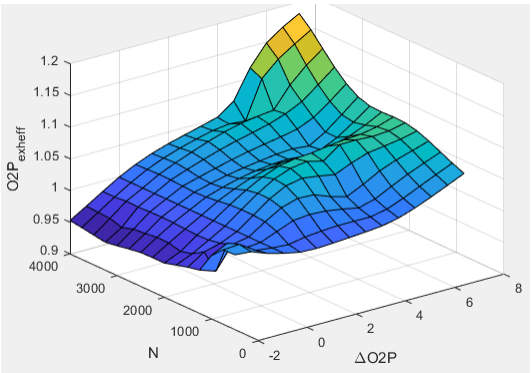

**Dependencies**

To enable this parameter, for **Torque model**, select `Torque Structure`.

**`Intake manifold gas oxygen exhaust temperature efficiency multiplier, f_tqs_exht_o2pct_eff` — Intake manifold efficiency multiplier**
`array`

The intake manifold gas oxygen exhaust temperature efficiency multiplier lookup table, $f_{O2Pexheff}$, is a function of the engine speed and intake manifold gas oxygen percent relative to optimal, $O2P_{exheff} = f_{O2Pexheff}(\Delta O2P,N)$, where:

- $O2P_{exheff}$ is intake manifold gas oxygen exhaust temperature efficiency multiplier, dimensionless.
- $\Delta O2P$ is intake gas oxygen percent relative to optimal, in percent.
- $N$ is engine speed, in rpm.



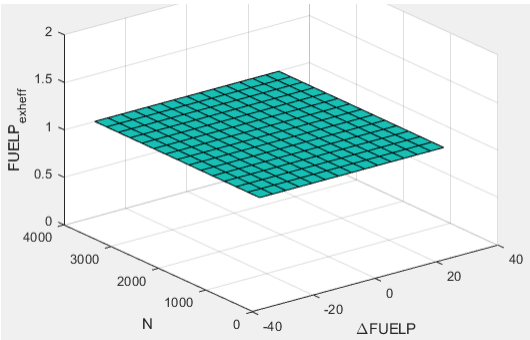

**Dependencies**

To enable this parameter, for **Torque model**, select `Torque Structure`.

**`Fuel rail pressure exhaust temperature efficiency multiplier, f_tqs_exht_fuelpress_eff` — Fuel rail pressure exhaust temperature efficiency multiplier**
`array`

The fuel rail pressure efficiency exhaust temperature multiplier lookup table, $f_{FUELPexheff}$, is a function of the engine speed and fuel rail pressure relative to optimal breakpoints, $FUELP_{exheff} = f_{FUELPexheff}(\Delta FUELP,N)$, where:

- $FUELP_{exheff}$ is fuel rail pressure exhaust temperature efficiency multiplier, dimensionless.
- $\Delta FUELP$ is fuel rail pressure relative to optimal, in MPa.
- $N$ is engine speed, in rpm.




**Dependencies**

To enable this parameter, for **Torque model**, select `Torque Structure`.

**`Post-injection cylinder wall heat loss transfer coefficient, f_tqs_exht_post_inj_wall_htc` — Post-injection offset**
`0` (default) | `scalar`

Post-injection cylinder wall heat loss transfer coefficient, in W/K.

**Dependencies**

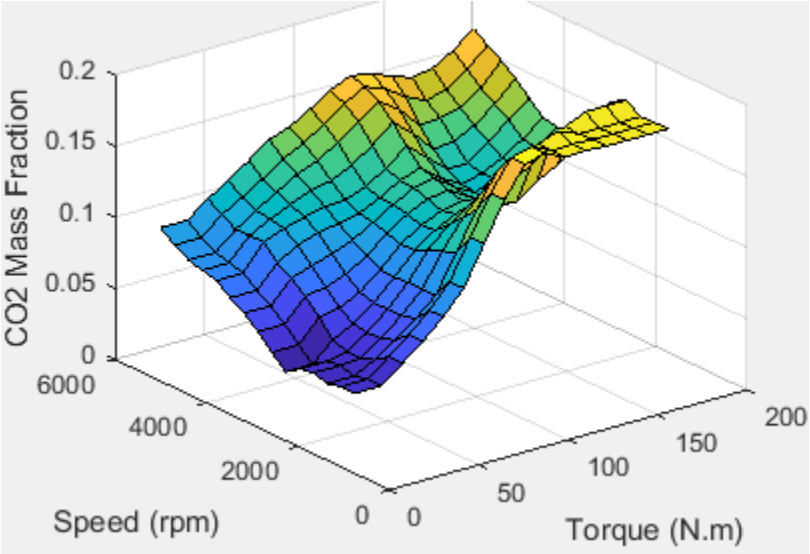
To enable this parameter, for **Torque model**, select `Torque Structure`.

**Emissions**

**`CO2 mass fraction table, f_CO2_frac` — Carbon dioxide ($CO_2$) emission lookup table**
`array`

The CI Core Engine $CO_2$ emission mass fraction lookup table is a function of engine torque and engine speed, $CO2\ Mass\ Fraction = f(Speed, Torque)$, where:

- *CO2 Mass Fraction* is the $CO_2$ emission mass fraction, dimensionless.
- *Speed* is engine speed, in rpm.
- *Torque* is engine torque, in N·m.



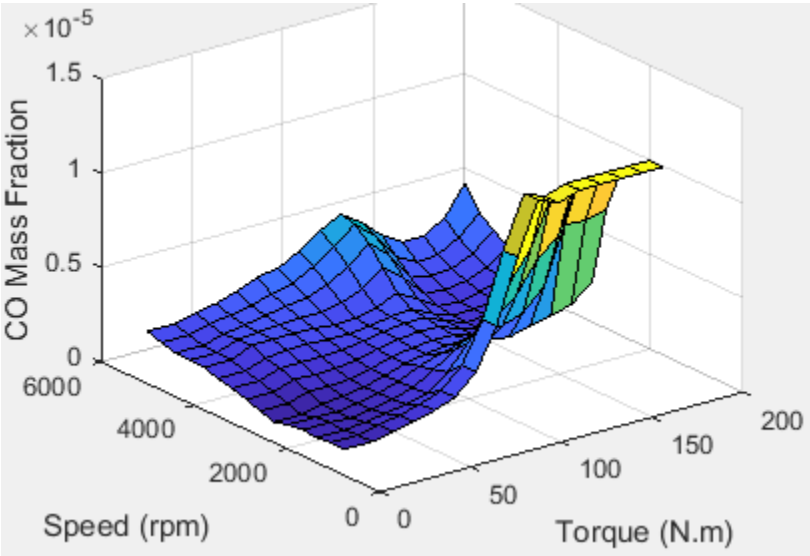

**Dependencies**

To enable this parameter, on the **Exhaust** tab, select **CO2**.

**`CO mass fraction table, f_CO_frac` — Carbon monoxide (CO) emission lookup table**
`array`

The CI Core Engine CO emission mass fraction lookup table is a function of engine torque and engine speed, *CO Mass Fraction* = *f*(*Speed*, *Torque*), where:

- *CO Mass Fraction* is the CO emission mass fraction, dimensionless.
- *Speed* is engine speed, in rpm.
- *Torque* is engine torque, in N·m.




**Dependencies**

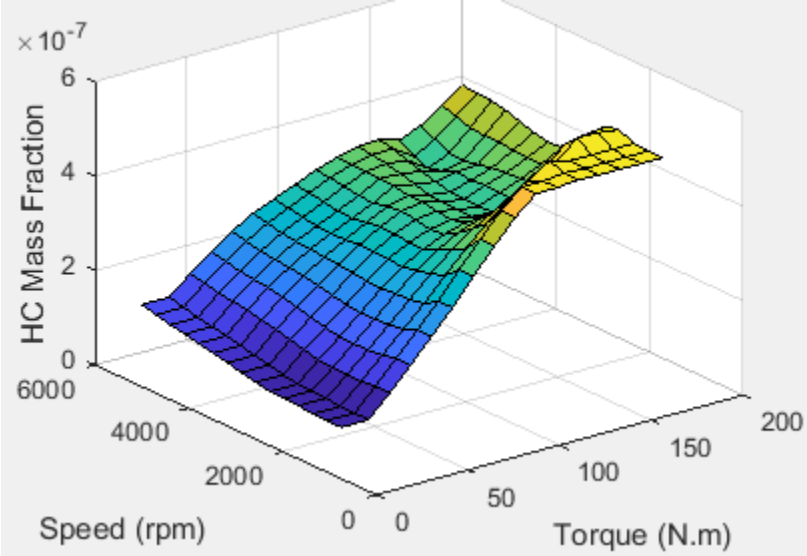
To enable this parameter, on the **Exhaust** tab, select **CO**.

### `HC mass fraction table, f_HC_frac` — Hydrocarbon (HC) emission lookup table

`array`

The CI Core Engine HC emission mass fraction lookup table is a function of engine torque and engine speed, *HC Mass Fraction* = *f*(*Speed*, *Torque*), where:

- *HC Mass Fraction* is the HC emission mass fraction, dimensionless.
- *Speed* is engine speed, in rpm.
- *Torque* is engine torque, in N·m.




#### Dependencies

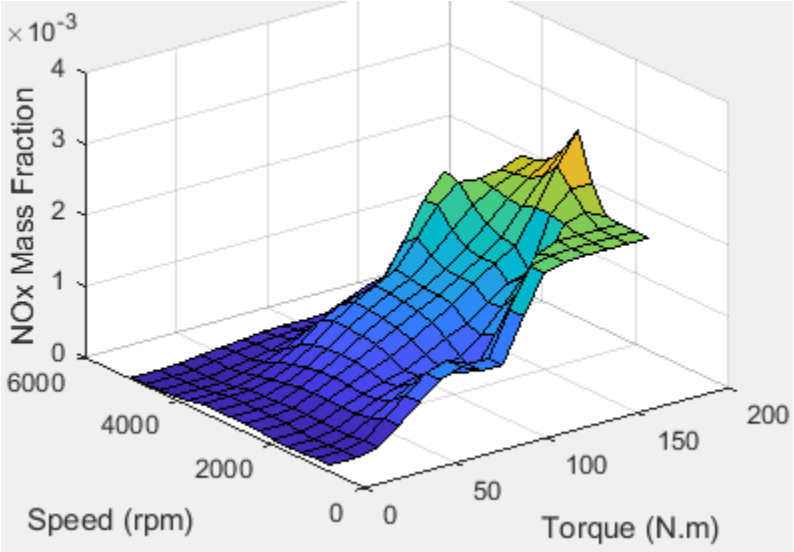
To enable this parameter, on the **Exhaust** tab, select **HC**.

### `NOx mass fraction table, f_NOx_frac` — Nitric oxide and nitrogen dioxide (NOx) emission lookup table

`array`

The CI Core Engine NOx emission mass fraction lookup table is a function of engine torque and engine speed, *NOx Mass Fraction* = *f*(*Speed*, *Torque*), where:

- *NOx Mass Fraction* is the NOx emission mass fraction, dimensionless.
- *Speed* is engine speed, in rpm.
- *Torque* is engine torque, in N·m.




**Dependencies**

To enable this parameter, on the **Exhaust** tab, select **NOx**.

**`PM mass fraction table, f_PM_frac` — Particulate matter (PM) emission lookup table**
`array`

The CI Core Engine PM emission mass fraction lookup table is a function of engine torque and engine speed where:

- *PM* is the PM emission mass fraction, dimensionless.
- *Speed* is engine speed, in rpm.
- *Torque* is engine torque, in N·m.

**Dependencies**

To enable this parameter, on the **Exhaust** tab, select **PM**.

**`Engine speed breakpoints, f_exhfrac_n_bpt` — Breakpoints**
`[750 1053.57142857143 1357.14285714286 1660.71428571429 1964.28571428571 2267.85714285714 2571.42857142857 2875 3178.57142857143 3482.14285714286 3785.71428571429 4089.28571428571 4392.85714285714 4696.42857142857 5000]` (default) | `vector`

Engine speed breakpoints used for the emission mass fractions lookup tables, in rpm.

**Dependencies**

To enable this parameter, on the **Exhaust** tab, select **CO2**, **CO**, **NOx**, **HC**, or **PM**.

**`Engine torque breakpoints, f_exhfrac_trq_bpt` — Breakpoints**
`[0 15 26.4285714285714 37.8571428571429 49.2857142857143 60.7142857142857 72.1428571428571 83.5714285714286 95 106.428571428571 117.857142857143 129.285714285714 140.714285714286 152.142857142857 163.571428571429 175]` (default) | `vector`

Engine torque breakpoints used for the emission mass fractions lookup tables, in N·m.

**Dependencies**

To enable this parameter, on the **Exhaust** tab, select **CO2**, **CO**, **NOx**, **HC**, or **PM**.

**`Exhaust gas specific heat at constant pressure, cp_exh` — Specific heat**
`1005` (default) | `scalar`

Exhaust gas-specific heat, $Cp_{exh}$, in J/(kg·K).

**Fuel**

**`Stoichiometric air-fuel ratio, afr_stoich` — Air-fuel ratio**
`14.6` (default) | `scalar`

Air-fuel ratio, *AFR*.

**`Fuel lower heating value, fuel_lhv` — Heating value**
`42e6` (default) | `scalar`

Fuel lower heating value, *LHV*, in J/kg.

**`Fuel specific gravity, fuel_sg` — Specific gravity**
`0.832` (default) | `scalar`

Specific gravity of fuel, $Sg_{fuel}$, dimensionless.

## References

[1] Heywood, John B. *Internal Combustion Engine Fundamentals*. New York: McGraw-Hill, 1988.

## Extended Capabilities

**C/C++ Code Generation**
Generate C and C++ code using Simulink® Coder™.

## See Also

CI Controller | Mapped CI Engine

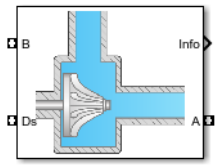
**Topics**
“CI Core Engine Air Mass Flow and Torque Production”
“Engine Calibration Maps”

**Introduced in R2017a**

# Compressor

Compressor for boosted engines

**Library:** Powertrain Blockset / Propulsion / Combustion Engine Components / Boost

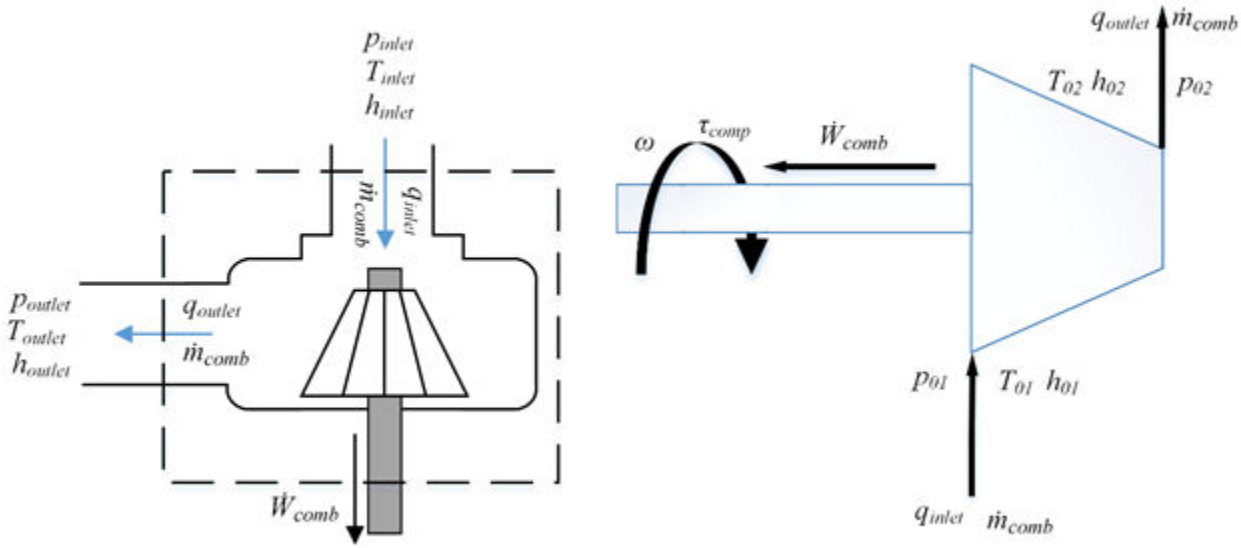



## Description

The Compressor block simulates engine boost by using the drive shaft energy to increase the intake manifold pressure. The block is a component of supercharger and turbocharger models. The block uses two-way ports to connect to the inlet and outlet control volumes and the drive shaft. The control volumes provide the pressure, temperature, and specific enthalpy for the compressor to calculate the mass and energy flow rates. To calculate the torque and flow rates, the drive shaft provides the speed to the compressor. Typically, compressor manufacturers provide the mass flow rate and efficiency tables as a function of corrected speed and pressure ratio. You can specify the lookup tables to calculate the mass flow rate and efficiency. The block does not support reverse mass flow.

If you have Model-Based Calibration Toolbox, click **Calibrate Performance Maps** to virtually calibrate the mass flow rate and turbine efficiency lookup tables using measured data.

The mass flows from the inlet control volume to the outlet control volume.



### Virtual Calibration

If you have Model-Based Calibration Toolbox, click **Calibrate Performance Maps** to virtually calibrate the mass flow rate and turbine efficiency lookup tables using measured data. The dialog box steps through these tasks.

<table>
<tr><th>Task</th><th>Description</th></tr>
<tr><td>Import compressor data</td><td>Import this compressor data from a file. For more information, see “Using Data” (Model-Based Calibration Toolbox).<ul><li>Speed, Spd, in rad/s</li><li>Mass flow rate, MassFlwRate, in kg/s</li><li>Pressure ratio, PrsRatio, dimensionless</li><li>Efficiency, Eff, dimensionless</li></ul>The speed, mass flow rate, pressure ratio, and efficiency are in the 2nd-5th columns of the data file, respectively. The first and second rows of the data file provide the variable names and units. For example, use this format.<table><tr><td>Name:</td><td>Spd</td><td>MassFlwRate</td><td>PrsRatio</td><td>Eff</td></tr><tr><td>Unit:</td><td>rad/s</td><td>kg/s</td><td></td><td></td></tr><tr><td>Data:</td><td>8373.3</td><td>0.02</td><td>1.21</td><td>0.44</td></tr><tr><td></td><td>...</td><td>...</td><td>...</td><td>...</td></tr></table>Model-Based Calibration Toolbox limits the speed and pressure ratio breakpoint values to the maximum values in the file.<br><br>To filter or edit the data, select Edit in Application. The Model-Based Calibration Toolbox Data Editor opens.</td></tr>
<tr><td>Generate response models</td><td>Model-Based Calibration Toolbox fits the imported data to the response models.<table><tr><th>Data</th><th>Response Model</th></tr><tr><td>Mass flow rate</td><td>Extended ellipse response model described in Modeling and Control of Engines and Drivelines[2]</td></tr><tr><td>Efficiency</td><td>Polynomial</td></tr></table>To assess or adjust the response model fit, select Edit in Application. The Model-Based Calibration Toolbox Model Browser opens. For more information, see “Model Assessment” (Model-Based Calibration Toolbox).</td></tr>
<tr><td>Generate calibration</td><td>Model-Based Calibration Toolbox calibrates the response model and generates calibrated tables.<br><br>To assess or adjust the calibration, select Edit in Application. The Model-Based Calibration Toolbox CAGE Browser opens. For more information, see “Calibration Lookup Tables” (Model-Based Calibration Toolbox).</td></tr>
</table>

| Task | Description |
|---|---|
| Update block parameters | Update these mass flow rate and efficiency parameters with the calibration.<br>• **Corrected mass flow rate table, mdot_corr_tbl**<br>• **Efficiency table, eta_comp_tbl**<br>• **Corrected speed breakpoints, w_corr_bpts1**<br>• **Pressure ratio breakpoints, Pr_bpts2** |

### Thermodynamics

The block uses these equations to model the thermodynamics.

| Calculation | Equations |
|---|---|
| Forward mass flow | $\dot{m}_{comp} > 0$<br>$p_{01} = p_{inlet}$<br>$p_{02} = p_{outlet}$<br>$T_{01} = T_{inlet}$<br>$h_{01} = h_{inlet}$ |
| First law of thermodynamics | $\dot{W}_{comp} = \dot{m}_{comp} c_p (T_{01} - T_{02})$ |
| Isentropic efficiency | $\eta_{comp} = \frac{h_{02s} - h_{01}}{h_{02} - h_{01}} = \frac{T_{02s} - T_{01}}{T_{02} - T_{01}}$ |
| Isentropic outlet temperature, assuming ideal gas and constant specific heats | $T_{02s} = T_{01}\left(\frac{p_{02}}{p_{01}}\right)^{\frac{\gamma - 1}{\gamma}}$ |
| Specific heat ratio | $\gamma = \frac{c_p}{c_p - R}$ |
| Outlet temperature | $T_{02} = T_{01} + \frac{T_{01}}{\eta_{comb}}\left\{\left(\frac{p_{02}}{p_{01}}\right)^{\frac{\gamma - 1}{\gamma}} - 1\right\}$ |
| Heat flows | $q_{inlet} = \dot{m}_{comp} h_{01}$<br>$q_{outlet} = \dot{m}_{comp} h_{02} = \dot{m}_{comp} c_p T_{02}$ |
| Corrected mass flow rate | $\dot{m}_{corr} = \dot{m}_{comp} \frac{\sqrt{T_{01}/T_{ref}}}{p_{01}/p_{ref}}$ |
| Corrected speed | $\omega_{corr} = \frac{\omega}{\sqrt{T_{01}/T_{ref}}}$ |
| Pressure ratio | $p_r = \frac{p_{01}}{p_{02}}$ |

The block uses the internal signal `FlwDir` to track the direction of the flow.

The equations use these variables.

| | |
|---|---|
| $p_{\text{inlet}}$, $p_{01}$ | Inlet control volume total pressure |
| $T_{inlet}$, $T_{01}$ | Inlet control volume total temperature |
| $h_{inlet}$, $h_{01}$ | Inlet control volume total specific enthalpy |
| $p_{outlet}$, $p_{02}$ | Outlet control volume total pressure |
| $T_{outlet}$ | Outlet control volume total temperature |
| $h_{outlet}$ | Outlet control volume total specific enthalpy |
| $\dot{W}_{comp}$ | Drive shaft power |
| $T_{02}$ | Outlet total temperature |
| $h_{02}$ | Outlet total specific enthalpy |
| $\dot{m}_{comp}$ | Mass flow rate through compressor |
| $q_{inlet}$ | Inlet heat flow rate |
| $q_{outlet}$ | Outlet heat flow rate |
| $\eta_{comp}$ | Compressor isentropic efficiency |
| $T_{02s}$ | Isentropic outlet total temperature |
| $h_{02s}$ | Isentropic outlet total specific enthalpy |
| $R$ | Ideal gas constant |
| $c_p$ | Specific heat at constant pressure |
| $\gamma$ | Specific heat ratio |
| $\dot{m}_{corr}$ | Corrected mass flow rate |
| $\omega$ | Drive shaft speed |
| $\omega_{corr}$ | Corrected drive shaft speed |
| $T_{ref}$ | Lookup table reference temperature |
| $P_{ref}$ | Lookup table reference pressure |
| $\tau_{comp}$ | Compressor drive shaft torque |
| $p_r$ | Pressure ratio |
| $\eta_{comb,\,tbl}$ | Compressor efficiency 3-D lookup table |
| $\dot{m}_{corr,\,tbl}$ | Corrected mass flow rate 3-D lookup table |
| $\omega_{corr,\,bpts1}$ | Corrected speed breakpoints |
| $p_{r,\,bpts2}$ | Pressure ratio breakpoints |

### Power Accounting

For the power accounting, the block implements these equations.

| Bus Signal | | | Description | Equations |
|---|---|---|---|---|
| PwrInfo | PwrTrnsfrd — Power transferred between blocks<br>• Positive signals indicate flow into block<br>• Negative signals indicate flow out of block | PwrDriveshft | Power transmitted from the shaft | $-\dot{W}_{turb}$ |
| | | PwrHeatFlwIn | Heat flow rate at port A | $q_{outlet}$ |
| | | PwrHeatFlwOut | Heat flow rate at port B | $q_{outlet}$ |
| | PwrNotTrnsfrd — Power crossing the block boundary, but not transferred<br>• Positive signals indicate an input<br>• Negative signals indicate a loss | PwrLoss | Power loss | $-q_{inlet} - q_{outlet} + \dot{W}_{turb}$ |
| | PwrStored — Stored energy rate of change<br>• Positive signals indicate an increase<br>• Negative signals indicate a decrease | *Not used* | | |

The equations use these variables.

| | |
|---|---|
| $\dot{W}_{turb}$ | Drive shaft power |
| $q_{outlet}$ | Total outlet heat flow rate |
| $q_{inlet}$ | Total inlet heat flow rate |

## Ports

### Input

#### Ds — Drive shaft speed

two-way connector port

ShftSpd — Signal containing the drive shaft angular speed, $\omega$, in rad/s.

#### A — Inlet pressure, temperature, enthalpy, mass fractions

two-way connector port

Bus containing the inlet control volume:

- InPrs — Pressure, $p_{\text{inlet}}$, in Pa
- InTemp — Temperature, $T_{inlet}$, in K
- InEnth — Specific enthalpy, $h_{inlet}$, in J/kg

#### B — Outlet pressure, temperature, enthalpy, mass fractions

two-way connector port

Bus containing the outlet control volume:

- `OutPrs` — Pressure, $p_{outlet}$, in Pa
- `OutTemp` — Temperature, $T_{outlet}$, in K
- `OutEnth` — Specific enthalpy, $h_{outlet}$, in J/kg

### Output

#### `Info` — Bus signal

bus

Bus signal containing these block calculations.

| Signal | | | Description | Units |
|---|---|---|---|---|
| `CmprsOutletTemp` | | | Temperature exiting the compressor | K |
| `DriveshftPwr` | | | Drive shaft power | W |
| `DriveshftTrq` | | | Drive shaft torque | N·m |
| `CmprsMassFlw` | | | Mass flow rate through compressor | kg/s |
| `PrsRatio` | | | Pressure ratio | N/A |
| `DriveshftCorrSpd` | | | Corrected drive shaft speed | rad/s |
| `CmprsEff` | | | Compressor isentropic efficiency | N/A |
| `CorrMassFlw` | | | Corrected mass flow rate | kg/s |
| `PwrInfo` | `PwrTrnsfrd` | `PwrDriveshft` | Power transmitted from the shaft | W |
| | | `PwrHeatFlwIn` | Heat flow rate at port A | W |
| | | `PwrHeatFlwOut` | Heat flow rate at port B | W |
| | `PwrNotTrnsfrd` | `PwrLoss` | Power loss | W |
| | `PwrStored` | | *Not used* | W |

#### `Ds` — Drive shaft torque

two-way connector port

`Trq` — Signal containing the drive shaft torque, $\tau_{comp}$, in N·m.

#### `A` — Inlet mass flow rate, heat flow rate, temperature, mass fractions

two-way connector port

Bus containing:

- `MassFlwRate` — Mass flow rate through inlet,$\dot{m}_{comp}$, in kg/s
- `HeatFlwRate` — Inlet heat flow rate, $q_{inlet}$, in J/s
- `Temp` — Inlet temperature, in K
- `MassFrac` — Inlet mass fractions, dimensionless.

  Specifically, a bus with these mass fractions:

- `O2MassFrac` — Oxygen
- `N2MassFrac` — Nitrogen
- `UnbrndFuelMassFrac` — Unburned fuel
- `CO2MassFrac` — Carbon dioxide
- `H2OMassFrac` — Water
- `COMassFrac` — Carbon monoxide
- `NOMassFrac` — Nitric oxide
- `NO2MassFrac` — Nitrogen dioxide
- `NOxMassFrac` — Nitric oxide and nitrogen dioxide
- `PmMassFrac` — Particulate matter
- `AirMassFrac` — Air
- `BrndGasMassFrac` — Burned gas

**`B` — Outlet mass flow rate, heat flow rate, temperature, mass fractions**
two-way connector port

Bus containing:

- `MassFlwRate` — Outlet mass flow rate, $\dot{m}_{comp}$, in kg/s
- `HeatFlwRate` — Outlet heat flow rate, $q_{outlet}$, in J/s
- `Temp` — Outlet temperature, in K
- `MassFrac` — Outlet mass fractions, dimensionless.

  Specifically, a bus with these mass fractions:

  - `O2MassFrac` — Oxygen
  - `N2MassFrac` — Nitrogen
  - `UnbrndFuelMassFrac` — Unburned fuel
  - `CO2MassFrac` — Carbon dioxide
  - `H2OMassFrac` — Water
  - `COMassFrac` — Carbon monoxide
  - `NOMassFrac` — Nitric oxide
  - `NO2MassFrac` — Nitrogen dioxide
  - `NOxMassFrac` — Nitric oxide and nitrogen dioxide
  - `PmMassFrac` — Particulate matter
  - `AirMassFrac` — Air
  - `BrndGasMassFrac` — Burned gas

## Parameters

#### Performance Tables

**`Calibrate Performance Maps` — Calibrate tables with measured data**
`selection`

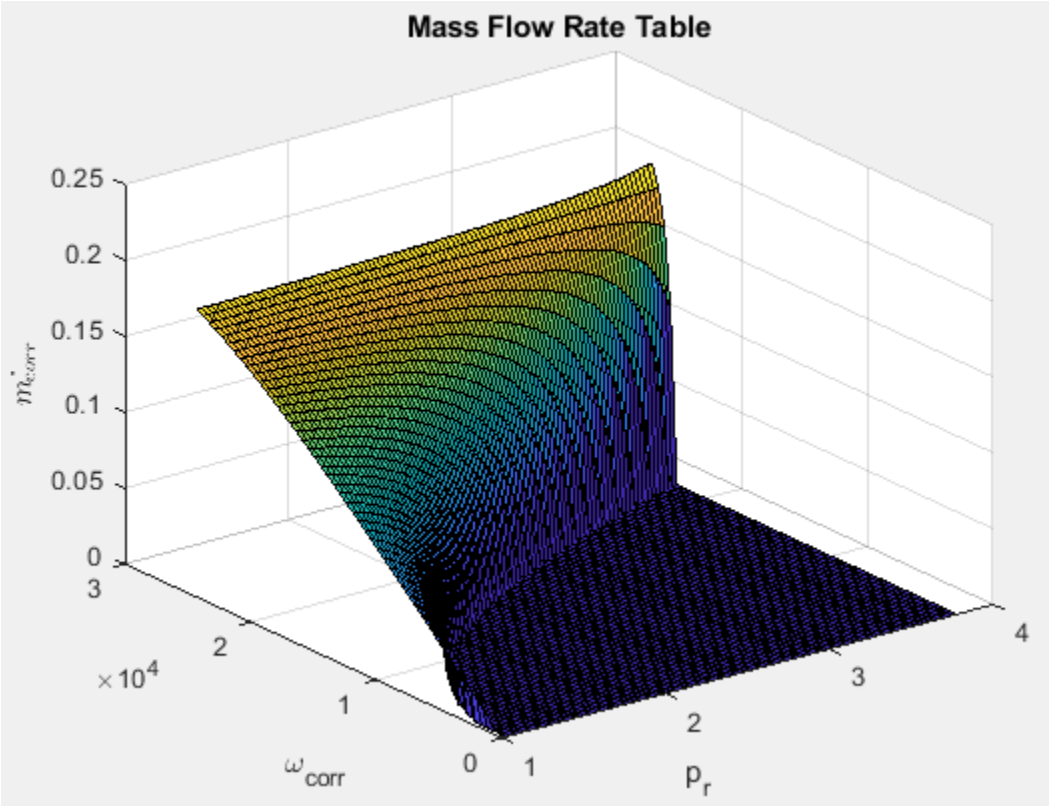
If you have Model-Based Calibration Toolbox, click **Calibrate Performance Maps** to virtually calibrate the mass flow rate and turbine efficiency lookup tables using measured data. The dialog box steps through these tasks.

<table>
<tr><th>Task</th><th>Description</th></tr>
<tr><td>Import compressor data</td><td>Import this compressor data from a file. For more information, see “Using Data” (Model-Based Calibration Toolbox).<ul><li>Speed, Spd, in rad/s</li><li>Mass flow rate, MassFlwRate, in kg/s</li><li>Pressure ratio, PrsRatio, dimensionless</li><li>Efficiency, Eff, dimensionless</li></ul>The speed, mass flow rate, pressure ratio, and efficiency are in the 2nd-5th columns of the data file, respectively. The first and second rows of the data file provide the variable names and units. For example, use this format.<table><tr><td>Name:</td><td>Spd</td><td>MassFlwRate</td><td>PrsRatio</td><td>Eff</td></tr><tr><td>Unit:</td><td>rad/s</td><td>kg/s</td><td></td><td></td></tr><tr><td>Data:</td><td>8373.3</td><td>0.02</td><td>1.21</td><td>0.44</td></tr><tr><td></td><td>...</td><td>...</td><td>...</td><td>...</td></tr></table>Model-Based Calibration Toolbox limits the speed and pressure ratio breakpoint values to the maximum values in the file.<br><br>To filter or edit the data, select Edit in Application. The Model-Based Calibration Toolbox Data Editor opens.</td></tr>
<tr><td>Generate response models</td><td>Model-Based Calibration Toolbox fits the imported data to the response models.<table><tr><th>Data</th><th>Response Model</th></tr><tr><td>Mass flow rate</td><td>Extended ellipse response model described in Modeling and Control of Engines and Drivelines[2]</td></tr><tr><td>Efficiency</td><td>Polynomial</td></tr></table>To assess or adjust the response model fit, select Edit in Application. The Model-Based Calibration Toolbox Model Browser opens. For more information, see “Model Assessment” (Model-Based Calibration Toolbox).</td></tr>
<tr><td>Generate calibration</td><td>Model-Based Calibration Toolbox calibrates the response model and generates calibrated tables.<br><br>To assess or adjust the calibration, select Edit in Application. The Model-Based Calibration Toolbox CAGE Browser opens. For more information, see “Calibration Lookup Tables” (Model-Based Calibration Toolbox).</td></tr>
</table>

| Task | Description |
|---|---|
| Update block parameters | Update these mass flow rate and efficiency parameters with the calibration.<br>• **Corrected mass flow rate table, mdot_corr_tbl**<br>• **Efficiency table, eta_comp_tbl**<br>• **Corrected speed breakpoints, w_corr_bpts1**<br>• **Pressure ratio breakpoints, Pr_bpts2** |

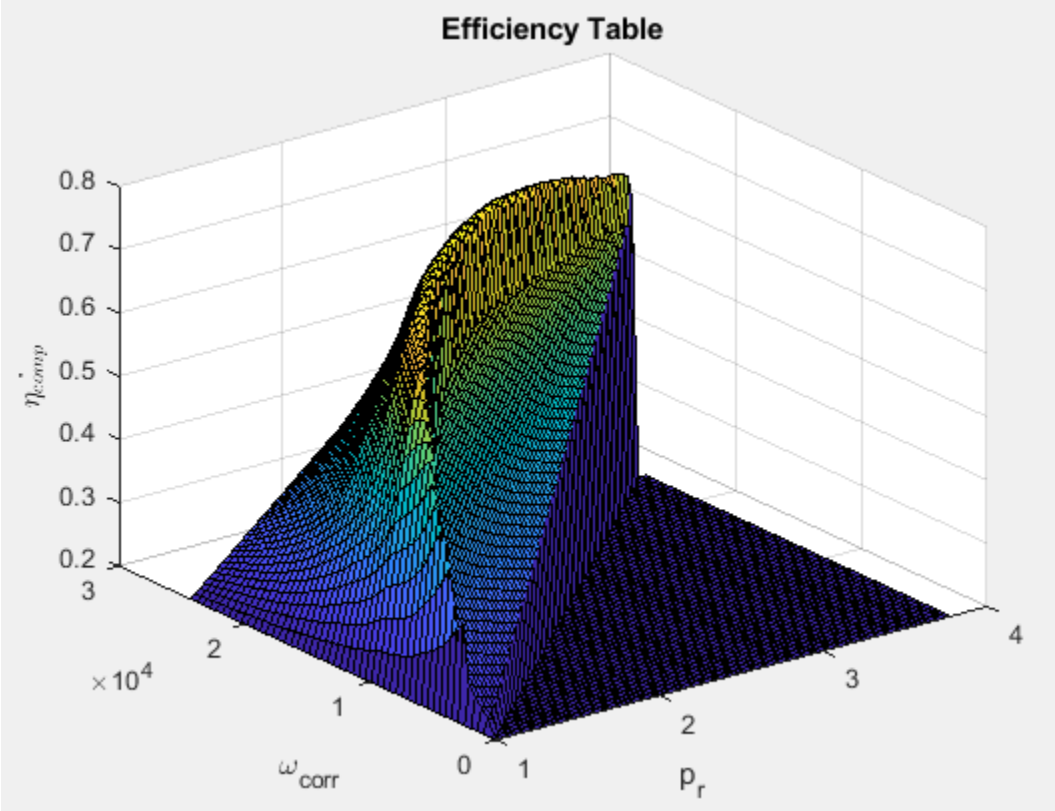
**`Corrected mass flow rate table, mdot_corr_tbl` — Lookup table**
`array`

Corrected mass flow rate lookup table, $\dot{m}_{corr,tbl}$, as a function of corrected driveshaft speed, $\omega_{corr}$, and pressure ratio, $p_r$, in kg/s.




**`Efficiency table, eta_comp_tbl` — Lookup table**
`array`

Efficiency lookup table, $\eta_{comb,tbl}$, as a function of corrected driveshaft speed, $\omega_{corr}$, and pressure ratio, $p_r$, dimensionless.




**`Corrected speed breakpoints, w_corr_bpts1` — Breakpoints**
`vector`

Corrected drive shaft speed breakpoints, $\omega_{corr,\,bpts1}$, in rad/s.

**`Pressure ratio breakpoints, Pr_bpts2` — Breakpoints**
`vector`

Pressure ratio breakpoints, $p_{r,\,bpts2}$.

**`Reference temperature, T_ref` — Reference**
`293.15` (default) | `scalar`

Lookup table reference temperature, $T_{ref}$, in K.

**`Reference pressure, P_ref` — Reference**
`101325` (default) | `scalar`

Lookup table reference pressure, $P_{ref}$, in Pa.

#### Gas Properties

**`Ideal gas constant, R` — Constant**
`287` (default) | `scalar`

Ideal gas constant, $R$, in J/(kg*K).

**`Specific heat at constant pressure, cp` — Specific heat**
`1005` (default) | `scalar`

Specific heat at constant pressure, $c_p$, in J/(kg*K).

## References

[1] Heywood, John B. *Internal Combustion Engine Fundamentals*. New York: McGraw-Hill, 1988.

[2] Eriksson, Lars and Lars Nielsen. *Modeling and Control of Engines and Drivelines*. Chichester, West Sussex, United Kingdom: John Wiley & Sons Ltd, 2014.

## Extended Capabilities

**C/C++ Code Generation**
Generate C and C++ code using Simulink® Coder™.

## See Also
Boost Drive Shaft | Turbine

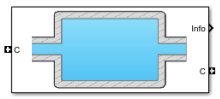
**Topics**
“Model-Based Calibration Toolbox”

**Introduced in R2017a**

# Control Volume System

Constant volume open thermodynamic system with heat transfer

**Library:** Powertrain Blockset / Propulsion / Combustion Engine Components / Fundamental Flow




## Description

The Control Volume System block models a constant volume open thermodynamic system with heat transfer. The block uses the conservation of mass and energy, assuming an ideal gas, to determine the pressure and temperature. The block implements an automotive-specific Constant Volume Pneumatic Chamber block that includes thermal effects related to the under hood of passenger vehicles. You can specify heat transfer models:

- Constant
- External input
- External wall convection

You can use the Control Volume System block to represent engine components that contain volume, including pipes and manifolds.

### Thermodynamics

The Control Volume System block implements a constant volume chamber containing an ideal gas. To determine the rate changes in temperature and pressure, the block uses the continuity equation and the first law of thermodynamics.

$$\frac{dT_{vol}}{dt} = \frac{RT_{vol}}{c_v V_{ch} P_{vol}} \left( \sum (q_i - T_{vol} c_v \dot{m}_i) - Q_{wall} \right)$$

$$\frac{dP_{vol}}{dt} = \frac{P_{vol}}{T_{vol}} \frac{dT_{vol}}{dt} + \frac{RT_{vol}}{V_{ch}} \sum \dot{m}_i$$

The block uses this equation for the volume-specific enthalpy.

$$h_{vol} = c_p T_{vol}$$

The equations use these variables.

| | |
|---|---|
| $\dot{m}_i$ | Mass flow rate at port |
| $q_i$ | Heat flow rate at port |
| $V_{ch}$ | Chamber volume |
| $P_{vol}$ | Absolute pressure in the chamber |
| $R$ | Ideal gas constant |
| $c_v$ | Specific heat at constant volume |

| | |
|---|---|
| $T_{vol}$ | Absolute gas temperature |
| $Q_{wall}$ | Wall heat transfer rate |
| $h_{vol}$ | Volume-specific enthalpy |
| $c_p$ | Specific heat capacity |

### Mass Fractions

The Control Volume Source block is part of a flow network. Blocks in the network determine the mass fractions that the block will track during simulation. The block can track these mass fractions:

- `O2` — Oxygen
- `N2` — Nitrogen
- `UnburnedFuel` — Unburned fuel
- `CO2` — Carbon dioxide
- `H2O` — Water
- `CO` — Carbon monoxide
- `NO` — Nitric oxide
- `NO2` — Nitrogen dioxide
- `PM` — Particulate matter
- `Air` — Air
- `BurnedGas` — Burned gas

Using the conservation of mass for each gas constituent, this equation determines the rate change:

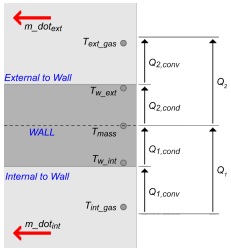
$$\frac{dy_{vol,\,j}}{dt} = \frac{RT_{vol}}{P_{vol}V_{ch}}\left(\sum \dot{m}_i y_{i,\,j} + y_{vol,\,j}\sum \dot{m}_i\right)$$

The equations use these variables.

| | |
|---|---|
| $V_{ch}$ | Chamber volume |
| $P_{vol}$ | Absolute pressure in the chamber |
| $R$ | Ideal gas constant |
| $T_{vol}$ | Absolute gas temperature |
| $y_{i,j}$ | I-th port mass fraction for j = $O_2$, $N_2$, unburned fuel, $CO_2$, $H_2O$, CO, NO, $NO_2$, PM, air, and burned gas |
| $y_{vol,j}$ | Control volume mass fraction for j = $O_2$, $N_2$, unburned fuel, $CO_2$, $H_2O$, CO, NO, $NO_2$, PM, air, and burned gas |
| $\dot{m}_i$ | Mass flow rate for i = $O_2$, $N_2$, unburned fuel, $CO_2$, $H_2O$, CO, NO, $NO_2$, PM, air, and burned gas |

### External Wall Convection Heat Transfer Model

To calculate the heat transfer, you can configure the Control Volume Source block to calculate the heat transfer across the wall of the control volume.




The block implements these equations to calculate the heat transfer, $Q_1$, from the internal control volume gas to the internal wall depth, $D_{int_cond}$.

$$Q_1 = Q_{1,conv} = Q_{1,cond}$$

$$Q_{1,conv} = h_{int}(x_{int}) \bullet A_{int_conv} \bullet \left(T_{int_gas} - T_{w_int}\right)$$

$$Q_{1,cond} = k_{int} \bullet \frac{A_{int_cond}}{D_{int_cond}} \bullet \left(T_{w_int} - T_{mass}\right)$$

The block implements these equations to calculate the heat transfer, $Q_2$, from the external wall depth, $D_{ext_cond}$ to the external gas.

$$Q_2 = Q_{2,conv} = h_{ext}(x_{ext}) \bullet A_{ext_conv} \bullet \left(T_{w_ext} - T_{ext_gas}\right)$$

$$Q_{2,cond} = k_{ext} \bullet \frac{A_{ext_cond}}{D_{ext_cond}} \bullet \left(T_{mass} - T_{w_ext}\right)$$

This equation expresses the heat stored in the thermal mass.

$$\frac{dT_{mass}}{dt} = \frac{Q_1 - Q_2}{c_{p_{wall}} m_{wall}}$$

The block determines the interior convection heat transfer coefficient using a lookup table that is a function of the average mass flow rate.

$$\dot{m}_{int_gas} = \frac{1}{2}\sum|\dot{m}_i|$$

The equations use these variables.

| | |
|---|---|
| $Q_1$ | Heat flow from the internal gas to a specified wall depth |
| $Q_{1,conv}$ | Heat flow convection from the internal gas to the internal wall |
| $Q_{1,cond}$ | Conduction heat transfer rate |
| $Q_2$ | Heat transfer rate |
| $Q_{2,conv}$ | Convection heat transfer |
| $Q_{2,cond}$ | Heat flow conduction from the external middle portion of the wall to the external wall |
| $Q_{mass}$ | Heat stored in thermal mass |
| $h_{int}$ | Internal convection heat transfer coefficient |
| $x_{int}$ | Internal mass flow rate breakpoints |

| | |
|---|---|
| $A_{int_conv}$ | Internal flow convection area |
| $T_{int_gas}$ | Temperature of the gas inside the chamber |
| $T_{w_int}$ | Temperature of the inside wall of the chamber |
| $k_{int}$ | Internal wall thermal conductivity |
| $A_{int_cond}$ | Internal conduction area |
| $D_{int_cond}$ | Internal wall thickness |
| $h_{ext}$ | External convection heat transfer coefficient |
| $x_{ext}$ | External velocity breakpoints |
| $A_{ext_conv}$ | External convection area |
| $T_{ext_gas}$ | External gas temperature |
| $T_{w_ext}$ | Temperature of the external wall of the chamber |
| $k_{ext}$ | External wall thermal conductivity |
| $A_{ext_cond}$ | External conduction area |
| $D_{ext_cond}$ | External wall thickness |
| $T_{mass}$ | Temperature of the thermal mass |
| $c_{p_wall}$ | Wall heat capacity |
| $m_{wall}$ | Thermal mass |
| $Flw_{spd}$ | External flow velocity |
| $\dot{m}_{int_gas}$ | Average internal mass flow rate |

### Power Accounting

For the power accounting, the block implements these equation based on the number of inlet and outlet ports.

| Bus Signal | | | Description | Equations |
|---|---|---|---|---|
| `PwrInfo` | `PwrTrnsfrd` — Power transferred between blocks<br>• Positive signals indicate flow into block<br>• Negative signals indicate flow out of block | `PwrHeatFlwi` | Port $i$ heat flow | $q_i$ |
| | `PwrNotTrnsfrd` — Power crossing the block boundary, but not transferred<br>• Positive signals indicate an input<br>• Negative signals indicate a loss | `PwrHeatTrnsfr` | Heat transfer rate from wall to control volume | $-Q_{wall}$ |

| Bus Signal | | | Description | Equations |
|---|---|---|---|---|
| | `PwrStored` — Stored energy rate of change<br>• Positive signals indicate an increase<br>• Negative signals indicate a decrease | `PwrHeatStored` | Rate of heat stored in the control volume | $\left(\sum (q_i) - Q_{wall}\right)$ |

For example, if you configure your block with 3 input ports and 2 outlet ports, the block implements these equations

| Bus Signal | | | Description | Equations |
|---|---|---|---|---|
| `PwrInfo` | `PwrTrnsfrd` — Power transferred between blocks<br>• Positive signals indicate flow into block<br>• Negative signals indicate flow out of block | `PwrHeatFlw1` | Inlet port 1 heat flow | $q_1$ |
| | | `PwrHeatFlw2` | Inlet port 2 heat flow | $q_2$ |
| | | `PwrHeatFlw3` | Inlet port 3 heat flow | $q_3$ |
| | | `PwrHeatFlw4` | Outlet port 4 heat flow | $q_4$ |
| | | `PwrHeatFlw5` | Outlet port 5 heat flow | $q_5$ |
| | `PwrNotTrnsfrd` — Power crossing the block boundary, but not transferred<br>• Positive signals indicate an input<br>• Negative signals indicate a loss | `PwrHeatTrnsfr` | Heat transfer rate from wall to control volume | $-Q_{wall}$ |
| | `PwrStored` — Stored energy rate of change<br>• Positive signals indicate an increase<br>• Negative signals indicate a decrease | `PwrHeatStored` | Rate of heat stored in the control volume | $\left(\sum (q_i) - Q_{wall}\right)$ |

## Ports

### Input

#### `C` — Inlet mass flow rate, heat flow rate, mass fractions

two-way connector port

Bus containing:

- `MassFlw` — Mass flow rate through inlet, in kg/s
- `HeatFlw` — Inlet heat flow rate, in J/s

- `MassFrac` — Inlet mass fractions, dimensionless.

  Specifically, a bus with these mass fractions:

  - `O2MassFrac` — Oxygen
  - `N2MassFrac` — Nitrogen
  - `UnbrndFuelMassFrac` — Unburned fuel
  - `CO2MassFrac` — Carbon dioxide
  - `H2OMassFrac` — Water
  - `COMassFrac` — Carbon monoxide
  - `NOMassFrac` — Nitric oxide
  - `NO2MassFrac` — Nitrogen dioxide
  - `NOxMassFrac` — Nitric oxide and nitrogen dioxide
  - `PmMassFrac` — Particulate matter
  - `AirMassFrac` — Air
  - `BrndGasMassFrac` — Burned gas

**Dependencies**

To create input ports, specify the **Number of inlet ports** parameter.

### `HeatTrnsfrRate` — Heat transfer

`scalar`

External heat transfer input to control volume, $q_{he}$, in Kg/s.

**Dependencies**

To create this port, select `External input` for the **Heat transfer model** parameter.

### `ExtnlFlwVel` — External flow velocity

scalar

External flow velocity, $Flw_{spd}$, in m/s.

**Dependencies**

To create this port, select `External wall convection` for the **Heat transfer model** parameter.

### `ExtnlTemp` — Ambient temperature, K

`scalar`

**Dependencies**

To create this port, select `External wall convection` for the **Heat transfer model** parameter.

## Output

### `Info` — Bus signal

bus

Bus signal containing these block calculations.

| Signal | | | Description | Units |
|---|---|---|---|---|
| Vol | Prs | | Volume pressure | Pa |
| | Temp | | Volume temperature | K |
| | Enth | | Volume specific enthalpy | J/kg |
| | Species | O2MassFrac | Oxygen mass fraction | NA |
| | | N2MassFrac | Nitrogen mass fraction | NA |
| | | UnbrndFuelMassFrac | Unburned gas mass fraction | NA |
| | | CO2MassFrac | Carbon dioxide mass fraction | NA |
| | | H2OMassFrac | Water mass fraction | NA |
| | | COMassFrac | Carbon monoxide mass fraction | NA |
| | | NOMassFrac | Nitric oxide mass fraction | NA |
| | | NO2MassFrac | Nitrogen dioxide mass fraction | NA |
| | | NOxMassFrac | Nitric oxide and nitrogen dioxide mass fraction | NA |
| | | PmMassFrac | Particulate matter mass fraction | NA |
| | | AirMassFrac | Air mass fraction | NA |
| | | BrndGasMassFrac | Burned gas mass fraction | NA |
| HeatTrnsfr | HeatTrnsfrRate | | Wall heat transfer rate | J/s |
| | MassFlw | | Average internal mass flow rate | kg/s |
| | IntrnTemp | | Temperature of gas inside chamber | K |
| PwrInfo | PwrTrnsfrd | PwrHeatFlw*i* | Port *i* heat flow | W |
| | PwrNotTrnsfrd | PwrHeatTrnsfr | Heat transfer rate from wall to control volume | W |
| | PwrStored | PwrHeatStored | Rate of heat stored in the control volume | W |

**C — Outlet pressure, temperature, enthalpy, mass fractions**
two-way connector port

Bus containing the outlet control volume:

- Prs — Chamber pressure, in Pa
- Temp — Gas temperature, in K

- `Enth` — Specific enthalpy, in J/kg
- `MassFrac` — Mass fractions, dimensionless.

  Specifically, a bus with these mass fractions:

  - `O2MassFrac` — Oxygen
  - `N2MassFrac` — Nitrogen
  - `UnbrndFuelMassFrac` — Unburned fuel
  - `CO2MassFrac` — Carbon dioxide
  - `H2OMassFrac` — Water
  - `COMassFrac` — Carbon monoxide
  - `NOMassFrac` — Nitric oxide
  - `NO2MassFrac` — Nitrogen dioxide
  - `NOxMassFrac` — Nitric oxide and nitrogen dioxide
  - `PmMassFrac` — Particulate matter
  - `AirMassFrac` — Air
  - `BrndGasMassFrac` — Burned gas

#### Dependencies

To create outlet ports, specify the **Number of outlet ports** parameter.

## Parameters

#### Block Options

**`Number of inlet ports` — Number of ports**
`1` (default) | `0` | `2` | `3` | `4`

Number of inlet ports.

#### Dependencies

To create inlet ports, specify the number.

**`Number of outlet ports` — Number of ports**
`1` (default) | `0` | `2` | `3` | `4`

Number of outlet ports.

#### Dependencies

To create outlet ports, specify the number.

**`Heat transfer model` — Select model**
`Constant` (default) | `External input` | `External wall convection`

#### Dependencies

Selecting `Constant` or `External wall convection` enables the **Heat Transfer** parameters.

**`Image type` — Icon color**
`Cold` (default) | `Hot`

Select color for block icon:

- `Cold` for blue
- `Hot` for red

#### General

**`Chamber volume, Vch` — Volume**
`0.0029` (default) | `scalar`

Chamber volume, $V_{ch}$, in m^3.

**`Initial chamber pressure, Pinit` — Pressure**
`101325` (default) | `scalar`

Initial chamber pressure, $P_{vol}$, in Pa.

**`Initial chamber temperature, Tinit` — Temperature**
`298` (default) | `scalar`

Initial chamber temperature, $T_{vol}$, in K.

**`Ideal gas constant, R` — Ideal gas constant**
`287` (default) | `scalar`

Ideal gas constant, $R$, in J/(kg*K).

**`Specific heat capacity, cp` — Specific heat**
`1005` (default) | `scalar`

Specific heat capacity, $c_p$, in J/(kg·K).

#### Heat Transfer

**`Heat transfer rate, q_he` — Rate**
`0` (default) | `scalar`

Constant heat transfer rate, $q_{he}$, in J/s.

##### Dependencies

To enable this parameter, select `Constant` for the **Heat transfer model** parameter.

**`External convection heat transfer coefficient, ext_tbl` — Manifold external air**
`[40 160 740 2000]` (default) | `vector`

External convection heat transfer coefficient, $h_{ext}$, in W/(m^2K).

##### Dependencies

To enable this parameter, select `External wall convection` for the **Heat transfer model** parameter.

**`External velocity breakpoints, ext_bpts` — Manifold external air**
`linspace(0,180,4)` (default) | `vector`

External velocity breakpoints, $x_{ext}$, in m/s.

**Dependencies**

To enable this parameter, select `External wall convection` for the **Heat transfer model** parameter.

**`External convection area, Aext_conv` — Manifold external air**
`0.125` (default) | `scalar`

External convection area, $A_{ext_conv}$, in m^2.

**Dependencies**

To enable this parameter, select `External wall convection` for the **Heat transfer model** parameter.

**`Thermal mass, m_wall` — Manifold wall general**
`7` (default) | `scalar`

Thermal mass, $m_{wall}$, in kg.

**Dependencies**

To enable this parameter, select `External wall convection` for the **Heat transfer model** parameter.

**`Wall heat capacity, cp_wall` — Manifold wall general**
`900` (default) | `scalar`

Wall heat capacity, $c_{p_wall}$, in J/(kg·K).

**Dependencies**

To enable this parameter, select `External wall convection` for the **Heat transfer model** parameter.

**`Initial mass temperature, Tmass` — Manifold wall general**
`293.15` (default) | `scalar`

Initial mass temperature, $T_{mass}$, in K.

**Dependencies**

To enable this parameter, select `External wall convection` for the **Heat transfer model** parameter.

**`External wall thickness, Dext_cond` — Manifold wall external**
`0.004` (default) | `scalar`

External wall thickness, $D_{ext_cond}$, in m.

**Dependencies**

To enable this parameter, select `External wall convection` for the **Heat transfer model** parameter.

**`External conduction area, Aext_cond` — Manifold wall external**
`0.003` (default) | `scalar`

External conduction area, $A_{ext_cond}$, in m^2.

**Dependencies**

To enable this parameter, select `External wall convection` for the **Heat transfer model** parameter.

**`External wall thermal conductivity, kint` — Manifold wall external**
`25` (default) | `scalar`

External wall thermal conductivity, $k_{ext}$, in W/(m·K).

**Dependencies**

To enable this parameter, select `External wall convection` for the **Heat transfer model** parameter.

**`Internal wall thickness, Dint_cond` — Manifold wall internal**
`0.004` (default) | `scalar`

Internal wall thickness, $D_{int_cond}$, in m.

**Dependencies**

To enable this parameter, select `External wall convection` for the **Heat transfer model** parameter.

**`Internal conduction area, Aint_cond` — Manifold wall internal**
`0.003` (default) | `scalar`

Internal conduction area, $A_{int_cond}$, in m^2.

**Dependencies**

To enable this parameter, select `External wall convection` for the **Heat transfer model** parameter.

**`Internal wall thermal conductivity, kint` — Manifold wall internal**
`25` (default) | `scalar`

Internal wall thermal conductivity, $k_{int}$, in W/(m·K).

**Dependencies**

To enable this parameter, select `External wall convection` for the **Heat transfer model** parameter.

**`Internal convection heat transfer coefficient, int_tbl` — Manifold internal air**
`[40 160 740 2000]` (default) | `vector`

Internal convection heat transfer coefficient, $h_{int}$, in W/(m^2K).

**Dependencies**

To enable this parameter, select `External wall convection` for the **Heat transfer model** parameter.

**`Internal mass flow rate breakpoints, int_bpts` — Manifold internal air**
`linspace(0.0020,0.1100,4)` (default) | `vector`

Internal velocity breakpoints, $x_{int}$, in kg/s.

**Dependencies**

To enable this parameter, select `External wall convection` for the **Heat transfer model** parameter.

**`Internal flow convection area, Aint_conv` — Manifold internal air**
`0.125` (default) | `scalar`

Internal convection area, $A_{int_conv}$, in m^2.

**Dependencies**

To enable this parameter, select `External wall convection` for the **Heat transfer model** parameter.

## References

[1] Heywood, John B. *Internal Combustion Engine Fundamentals*. New York: McGraw-Hill, 1988.

## Extended Capabilities

**C/C++ Code Generation**
Generate C and C++ code using Simulink® Coder™.

## See Also

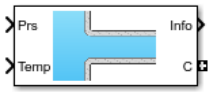
Constant Volume Pneumatic Chamber (Simscape) | Flow Restriction | Heat Exchanger

**Introduced in R2017a**

# Flow Boundary

Flow boundary for ambient temperature and pressure

**Library:** Powertrain Blockset / Propulsion / Combustion Engine Components / Fundamental Flow




## Description

The Flow Boundary block implements a flow boundary that typically represents ambient temperature and pressure. Engine models require flow boundaries at the intake inlet and exhaust outlet. In dynamic engine models, flow-modifying components (for example, flow restriction, turbines, and compressors) connect to control volumes and flow boundaries.

You can specify these block configurations:

- Constant pressure and temperature
- Externally input pressure and temperature

The Flow Boundary block outputs pressure, temperature, and specific enthalpy:

$$h = c_p T$$

The block models the mass fractions as dry air, resulting in these mass fractions:

- $y_{N2} = 0.767$
- $y_{O2} = .233$

The equation uses these variables.

| | |
|---|---|
| $T$ | Temperature |
| $h$ | Specific enthalpy |
| $c_p$ | Specific heat at constant pressure |
| $y_{N2}$ | Nitrogen mass fraction |
| $y_{O2}$ | Oxygen mass fraction |

### Power Accounting

For the power accounting, the block implements these equations.

| Bus Signal | | | Description | Equations |
|---|---|---|---|---|
| PwrInfo | PwrTrnsfrd — Power transferred between blocks<br>• Positive signals indicate flow into block<br>• Negative signals indicate flow out of block | PwrBndrFlw | Heat flow rate to flow restriction | $q_{orf}$ |
| | PwrNotTrnsfrd — Power crossing the block boundary, but not transferred<br>• Positive signals indicate an input<br>• Negative signals indicate a loss | PwrEnv | Heat flow rate to environment | $-q_{orf}$ |
| | PwrStored — Stored energy rate of change<br>• Positive signals indicate an increase<br>• Negative signals indicate a decrease | *Not used* | | |

## Ports

### Input

#### Prs — Pressure
scalar

External input pressure, *P*, in Pa.

##### Dependencies

To create this port, select `External input` for the **Pressure and temperature source** parameter.

#### Temp — Temperature
scalar

External input temperature, *T*, in K.

##### Dependencies

To create this port, select `External input` for the **Pressure and temperature source** parameter.

### Output

#### Info — Bus signal
bus

Bus signal containing these block calculations.

| Signal | | | Description | Units |
|---|---|---|---|---|
| BndryPrs | | | Boundary pressure | Pa |
| BndryTemp | | | Boundary temperature | K |
| BndryEnth | | | Boundary specific enthalpy | J/kg |
| PwrInfo | PwrTrnsfrd | PwrBndryFlw | Heat flow rate to flow restriction | W |
| | PwrNotTrnsfrd | PwrEnv | Heat flow rate to environment | W |
| | PwrStored | | *Not used* | |

**C — Boundary pressure, temperature, enthalpy, mass fractions**
two-way connector port

Bus containing the flow boundary:

- Prs — Pressure, *P*, in Pa
- Temp — Temperature, *T*, in K
- Enth — Specific enthalpy, *h*, in J/kg
- MassFrac — Mass fractions, dimensionless.

  Specifically, a bus with these mass fractions:

  - O2MassFrac — Oxygen
  - N2MassFrac — Nitrogen
  - UnbrndFuelMassFrac — Unburned fuel
  - CO2MassFrac — Carbon dioxide
  - H2OMassFrac — Water
  - COMassFrac — Carbon monoxide
  - NOMassFrac — Nitric oxide
  - NO2MassFrac — Nitrogen dioxide
  - NOxMassFrac — Nitric oxide and nitrogen dioxide
  - PmMassFrac — Particulate matter
  - AirMassFrac — Air
  - BrndGasMassFrac — Burned gas

## Parameters

#### Block Options

**Pressure and temperature source — Select source**
External input (default) | Constant

Pressure and temperature source.

#### Dependencies

The table summarizes the parameter and port dependencies.

| Value | Enables Parameters | Creates Ports |
|---|---|---|
| `Constant` | **Pressure, Pcnst**<br><br>**Temperature, Tcnst** | None |
| `External input` | None | `Prs`<br><br>`Temp` |

**`Image type` — Icon color**
`Cold` (default) | `Hot`

Select color for block icon:

- `Cold` for blue
- `Hot` for red

**`Pressure, Pcnst` — Constant**
`101325` (default) | `scalar`

Constant pressure, *P*, in Pa.

**Dependencies**

To enable this parameter, select `Constant` for the **Pressure and temperature source** parameter.

**`Temperature, Tcnst` — Constant**
`298.15` (default) | `scalar`

Constant temperature, *T*, in K.

**Dependencies**

To enable this parameter, select `Constant` for the **Pressure and temperature source** parameter.

**`Specific heat at constant pressure, cp` — Constant**
`1005` (default) | `scalar`

Specific heat at constant pressure, in J/(kg·K).

## References

[1] Heywood, John B. *Internal Combustion Engine Fundamentals*. New York: McGraw-Hill, 1988.

## Extended Capabilities

**C/C++ Code Generation**
Generate C and C++ code using Simulink® Coder™.

## See Also

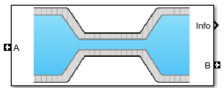
Compressor | Flow Restriction | Turbine

**Introduced in R2017a**

# Flow Restriction

Isentropic ideal gas flow through an orifice

**Library:** Powertrain Blockset / Propulsion / Combustion Engine Components / Fundamental Flow




## Description

The Flow Restriction block models isentropic ideal gas flow through an orifice. The block uses the conservation of mass and energy to determine the mass flow rate. The flow velocity is limited by choked flow.

You can specify these orifice area models:

- Constant
- External input
- Throttle body geometry

### Equations

The Flow Restriction block implements these equations.

| Calculation | Equations |
|---|---|
| Standard orifice | $\dot{m}_{orf} = \Gamma \cdot \Psi(P_{ratio})$<br>$P_{ratio} = \frac{P_{downstr}}{P_{upstr}}$<br>$\Gamma = \frac{A_{eff} \cdot P_{upstr}}{\sqrt{R \cdot T_{upstr}}}$<br>$P_{cr} = \left(\frac{2}{\gamma+1}\right)^{\frac{\gamma}{\gamma-1}}$<br>$\Psi = \begin{cases} \sqrt{\gamma\left(\frac{2}{\gamma+1}\right)^{\frac{\gamma+1}{\gamma-1}}} & P_{ratio} < P_{cr} \\ \sqrt{\frac{2\gamma}{\gamma-1}\left(P_{ratio}^{\frac{2}{\gamma}} - P_{ratio}^{\frac{\gamma+1}{\gamma}}\right)} & P_{cr} \le P_{ratio} \le P_{lim} \\ \frac{P_{ratio}-1}{P_{lim}-1}\sqrt{\frac{2\gamma}{\gamma-1}\left(P_{lim}^{\frac{2}{\gamma}} - P_{lim}^{\frac{\gamma+1}{\gamma}}\right)} & P_{lim} < P_{ratio} \end{cases}$ |
| Constituent mass flow rates | $\dot{m}_i = \dot{m}_{orf} y_{upstr,i}$ |
| Constant orifice area | $A_{eff} = A_{orf_cnst} \cdot Cd_{cnst}$ |

| Calculation | Equations |
|---|---|
| External input orifice area | $A_{eff} = A_{orf_ext} \cdot Cd_{ext}$ |
| Throttle body geometry | $\theta_{thr} = Pct_{thr} \cdot \frac{90}{100}$<br><br>$A_{eff_thr} = \frac{\pi}{4} D_{thr}{}^2 C_{d_thr}(\theta_{thr})$ |
| Heat flow rate | $q_{orf} = \dot{m}_{orf} h_{upstr}$ |

The equations use these variables.

| | |
|---|---|
| $A_{eff}$, $A_{eff_thr}$ | Effective orifice cross-sectional area |
| $A_{orf_cnst}$, $A_{orf_ext}$ | Orifice area |
| $Cd_{cnst}$, $Cd_{ext}$ | Discharge coefficient |
| $R$ | Ideal gas constant |
| $P_{cr}$ | Critical pressure at which choked flow occurs |
| $\gamma$ | Ratio of specific heats |
| $\Gamma$ | Flow function based on pressure ratio |
| $P_{ratio}$ | Pressure ratio |
| $P_{upstr}$ | Upstream orifice pressure |
| $P_{downstr}$ | Downstream orifice pressure |
| $P_{lim}$ | Pressure ratio limit to avoid singularities as the pressure ratio approaches 1 |
| $y_{upstr,i}$ | Upstream species mass fraction for i = $O_2$, $N_2$, unburned fuel, $CO_2$, $H_2O$, CO, NO, $NO_2$, PM, air, and burned gas |
| $\dot{m}_i$ | Mass flow rate for i = $O_2$, $N_2$, unburned fuel, $CO_2$, $H_2O$, CO, NO, $NO_2$, PM, air, and burned gas |
| $\theta_{thr}$ | Throttle angle |
| $Pct_{thr}$ | Percentage of throttle body that is open |
| $C_{d_thr}$ | Throttle discharge coefficient |
| $D_{thr}$ | Throttle body diameter at opening |
| $\dot{m}_{orf}$ | Orifice mass flow |
| $h_{upstr}$ | Upstream specific enthalpy |
| $q_{orf}$ | Heat flow rate |

The block uses the internal signal `FlwDir` to track the direction of the flow.

### Power Accounting

For the power accounting, the block implements these equations.

| Bus Signal | | | Description | Equations |
|---|---|---|---|---|
| PwrInfo | PwrTrnsfrd — Power transferred between blocks<br>• Positive signals indicate flow into block<br>• Negative signals indicate flow out of block | PwrHeatFlwIn | Heat flow rate at port A | $q_{orf}$ |
| | | PwrHeatFlwOut | Heat flow rate at port B | $-q_{orf}$ |
| | PwrNotTrnsfrd — Power crossing the block boundary, but not transferred<br>• Positive signals indicate an input<br>• Negative signals indicate a loss | *Not used* | | |
| | PwrStored — Stored energy rate of change<br>• Positive signals indicate an increase<br>• Negative signals indicate a decrease | *Not used* | | |

## Ports

### Input

#### A — Inlet orifice pressure, temperature, enthalpy, mass fractions

two-way connector port

Bus containing orifice:

- Prs — Pressure, in Pa
- Temp — Temperature, in K
- Enth — Specific enthalpy, in J/kg
- MassFrac — Inlet mass fractions, dimensionless.

  Specifically, a bus with these mass fractions:

  - O2MassFrac — Oxygen
  - N2MassFrac — Nitrogen
  - UnbrndFuelMassFrac — Unburned fuel
  - CO2MassFrac — Carbon dioxide
  - H2OMassFrac — Water
  - COMassFrac — Carbon monoxide
  - NOMassFrac — Nitric oxide
  - NO2MassFrac — Nitrogen dioxide
  - NOxMassFrac — Nitric oxide and nitrogen dioxide
  - PmMassFrac — Particulate matter
  - AirMassFrac — Air
  - BrndGasMassFrac — Burned gas

### B — Outlet orifice pressure, temperature, enthalpy, mass fractions

two-way connector port

Bus containing orifice:

- `Prs` — Pressure, in Pa
- `Temp` — Temperature, in K
- `Enth` — Specific enthalpy, in J/kg
- `MassFrac` — Outlet mass fractions, dimensionless.

  Specifically, a bus with these mass fractions:

  - `O2MassFrac` — Oxygen
  - `N2MassFrac` — Nitrogen
  - `UnbrndFuelMassFrac` — Unburned fuel
  - `CO2MassFrac` — Carbon dioxide
  - `H2OMassFrac` — Water
  - `COMassFrac` — Carbon monoxide
  - `NOMassFrac` — Nitric oxide
  - `NO2MassFrac` — Nitrogen dioxide
  - `NOxMassFrac` — Nitric oxide and nitrogen dioxide
  - `PmMassFrac` — Particulate matter
  - `AirMassFrac` — Air
  - `BrndGasMassFrac` — Burned gas

### Area — Orifice area

scalar

External area input for orifice area, $A_{orf_ext}$, in m^2.

#### Dependencies

To create this port, select `External input` for the **Orifice area model** parameter.

### ThrPct — Throttle body percent open

scalar

Percentage of throttle body that is open, $Pct_{thr}$.

#### Dependencies

To create this port, select `Throttle body geometry` for the **Orifice area model** parameter.

## Output

### A — Inlet mass flow rate, heat flow rate, temperature

two-way connector port

Bus containing:

- `MassFlw` — Mass flow rate through inlet, in kg/s
- `HeatFlw` — Inlet heat flow rate, in J/s
- `Temp` — Inlet temperature, in K
- `MassFrac` — Inlet mass fractions, dimensionless.

  Specifically, a bus with these mass fractions:

  - `O2MassFrac` — Oxygen
  - `N2MassFrac` — Nitrogen
  - `UnbrndFuelMassFrac` — Unburned fuel
  - `CO2MassFrac` — Carbon dioxide
  - `H2OMassFrac` — Water
  - `COMassFrac` — Carbon monoxide
  - `NOMassFrac` — Nitric oxide
  - `NO2MassFrac` — Nitrogen dioxide
  - `NOxMassFrac` — Nitric oxide and nitrogen dioxide
  - `PmMassFrac` — Particulate matter
  - `AirMassFrac` — Air
  - `BrndGasMassFrac` — Burned gas

**B — Outlet mass flow rate, heat flow rate, temperature**
two-way connector port

Bus containing:

- `MassFlw` — Outlet mass flow rate, in kg/s
- `HeatFlw` — Outlet heat flow rate, in J/s
- `Temp` — Outlet temperature, in K
- `MassFrac` — Outlet mass fractions, dimensionless.

  Specifically, a bus with these mass fractions:

  - `O2MassFrac` — Oxygen
  - `N2MassFrac` — Nitrogen
  - `UnbrndFuelMassFrac` — Unburned fuel
  - `CO2MassFrac` — Carbon dioxide
  - `H2OMassFrac` — Water
  - `COMassFrac` — Carbon monoxide
  - `NOMassFrac` — Nitric oxide
  - `NO2MassFrac` — Nitrogen dioxide
  - `NOxMassFrac` — Nitric oxide and nitrogen dioxide
  - `PmMassFrac` — Particulate matter
  - `AirMassFrac` — Air
  - `BrndGasMassFrac` — Burned gas

**`Info` — Bus signal**
bus

Bus signal containing these block calculations.

| Signal | | | | Description | Units |
|---|---|---|---|---|---|
| `Flw` | `PrsAdj` | | `DwnstrmPrs` | Downstream pressure | Pa |
| | | | `UpstrmPrs` | Upstream pressure | Pa |
| | | | `PrsRatio` | Pressure ratio | NA |
| | | | `DwnstrmTemp` | Downstream temperature | K |
| | | | `UpstrmTemp` | Upstream temperature | K |
| | `OrfMassFlw` | | | Mass flow rate through orifice | kg/s |
| | `Species` | | `O2MassFlw` | Oxygen mass flow rate | kg/s |
| | | | `N2MassFlw` | Nitrogen mass flow rate | kg/s |
| | | | `UnbrndFuelMassFlw` | Unburned gas mass flow rate | kg/s |
| | | | `CO2MassFlw` | Carbon dioxide mass flow rate | kg/s |
| | | | `H2OMassFlw` | Water mass flow rate | kg/s |
| | | | `COMassFlw` | Carbon monoxide mass flow rate | kg/s |
| | | | `NOMassFlw` | Nitric oxide mass flow rate | kg/s |
| | | | `NO2MassFlw` | Nitrogen dioxide mass flow rate | kg/s |
| | | | `NOxMassFlw` | Nitric oxide and nitrogen dioxide mass flow rate | kg/s |
| | | | `PmMassFlw` | Particulate matter mass flow rate | kg/s |
| | | | `AirMassFlw` | Air mass flow rate | kg/s |
| | | | `BrnedGasMassFlw` | Burned gas mass flow rate | kg/s |
| | `PwrInfo` | `PwrTrnsfrd` | `PwrHeatFlwIn` | Heat flow rate at port A | W |
| | | | `PwrHeatFlwOut` | Heat flow rate at port B | W |
| | | `PwrNotTrnsfrd` | | *Not used* | |
| | | `PwrStored` | | *Not used* | |
| `Area` | `FlwArea` | | | Cross-sectional flow area | m^2 |
| | `EffctArea` | | | Effective orifice cross-sectional area | m^2 |
| | `ThrAng` | | | Throttle area, if applicable | deg |

## Parameters

#### Block Options

**`Orifice area model` — Select model**
`Constant` (default) | `External input` | `Throttle body geometry`

Orifice area model.

#### Dependencies

The orifice area model enables the parameters on the **Area Parameters** tab.

**`Image type` — Icon color**
`Cold` (default) | `Hot`

Block icon color:

- `Cold` for blue.
- `Hot` for red.

#### General

**`Ratio of specific heats, gamma` — Ratio**
`1.3998` (default) | `scalar`

Ratio of specific heats, $\gamma$.

**`Ideal gas constant, R` — Constant**
`287.05` (default) | `scalar`

Ideal gas constant, $R$, in J/(kg·K).

**`Pressure ratio linearize limit, Plim` — Limit**
`0.95` (default) | `scalar`

Pressure ratio limit to avoid singularities as the pressure ratio approaches 1, $P_{lim}$.

#### Area

**`Constant area value, Aorf_cnst` — Area**
`.1` (default) | `scalar`

Constant area value, $A_{orf_cnst}$, in m^2.

#### Dependencies

To enable this parameter, select `Constant` for the **Orifice area model** parameter.

**`Discharge coefficient, Cd_cnst` — Coefficient**
`1` (default) | `scalar`

Discharge coefficient for constant area, $Cd_{cnst}$.

#### Dependencies

To enable this parameter, select `Constant` for the **Orifice area model** parameter.

**`Discharge coefficient, Cd_ext` — Coefficient**
`1` (default) | `scalar`

Discharge coefficient for external area input, $Cd_{ext}$.

**Dependencies**

To enable this parameter, select `External input` for the **Orifice area model** parameter.

**`Throttle diameter, Dthr` — Diameter**
`50` (default) | `scalar`

Throttle body diameter at opening, $D_{thr}$, in mm.

**Dependencies**

To enable this parameter, select `Throttle body geometry` for the **Orifice area model** parameter.

**`Discharge coefficient table, ThrCd` — Coefficient**
`[0.001; 0.735]` (default) | `vector`

Discharge coefficient table, $C_{d_thr}$.

**Dependencies**

To enable this parameter, select `Throttle body geometry` for the **Orifice area model** parameter.

**`Angle breakpoints, ThrAngBpts` — Angle**
`[0; 90]` (default) | `vector`

Angle breakpoints, $Thr_{ang_bpts}$, in deg.

**Dependencies**

To enable this parameter, select `Throttle body geometry` for the **Orifice area model** parameter.

## References

[1] Heywood, John B. *Internal Combustion Engine Fundamentals*. New York: McGraw-Hill, 1988.

## Extended Capabilities

**C/C++ Code Generation**
Generate C and C++ code using Simulink® Coder™.

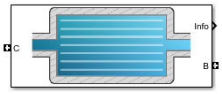
## See Also
Control Volume System | Heat Exchanger

**Introduced in R2017a**

# Heat Exchanger

Intercooler or exhaust gas recirculation (EGR) cooler

**Library:** Powertrain Blockset / Propulsion / Combustion Engine Components / Fundamental Flow




## Description

The Heat Exchanger block models a heat exchanger, for example, an intercooler or exhaust gas recirculation (EGR) cooler. The inlet (port C) connects to an engine flow component (flow restriction, compressor, turbine, or engine block). The outlet (port B) connects to a volume (control volume or environment). Based on the upstream temperature, heat exchanger effectiveness, and cooling medium temperature, the block determines the heat transfer rate and downstream temperature.

For the heat exchanger effectiveness and cooling medium temperature, you can specify either a constant value or an external input. For example, if you specify a heat exchanger effectiveness that is:

- Equal to 1, the downstream temperature is equal to the cooling medium temperature.
- Equal to 0, there is no heat transfer to the cooling medium. The downstream temperature is equal to the upstream temperature.

The block assumes no pressure drop. To model pressure losses, use a Flow Restriction block.

### Equations

The Heat Exchanger block implements equations that use these variables.

| | |
|---|---|
| $T_{upstr}$ | Upstream temperature |
| $T_{dnstr}$ | Downstream temperature |
| $T_{cool}$ | Cooling medium temperature |
| $T_{cool,cnst}$ | Constant cooling medium temperature |
| $T_{cool,input}$ | External input cooling medium temperature |
| $\varepsilon$ | Heat exchanger effectiveness |
| $\varepsilon_{cnst}$ | Constant heat exchanger effectiveness |
| $\varepsilon_{input}$ | Input heat exchanger effectiveness |
| $c_p$ | Specific heat at constant pressure |
| $q_{ht}$ | Heat exchanger heat transfer rate |
| $p_{flw,in}$ | Pressure at inlet |
| $p_{vol,out}$ | Pressure at outlet |
| $T_{vol,out}$ | Temperature at outlet |
| $h_{vol,out}$ | Specific enthalpy at outlet |

| | |
|---|---|
| $q_{in}$ | Heat flow rate at inlet |
| $q_{out}$ | Heat flow rate at outlet |
| $\dot{m}$ | Heat exchanger mass flow rate |
| $T_{flw,in}$ | Temperature at inlet |
| $T_{in}$ | Heat exchanger inlet temperature |
| $T_{out}$ | Heat exchanger outlet temperature |
| $h_{in}$ | Inlet specific enthalpy |

#### Heat Exchanger Effectiveness

Heat exchanger effectiveness measures the effectiveness of heat transfer from the incoming hot fluid to the cooling medium:

$$\varepsilon = \frac{T_{upstr} - T_{dnstr}}{T_{upstr} - T_{cool}}$$

In an ideal heat exchanger, the downstream temperature equals the cooling temperature. The effectiveness is equal to 1.

$$T_{dnstr} = T_{cool}$$
$$\varepsilon = 1$$

The Heat Exchanger block uses the effectiveness to determine the downstream temperature and heat transfer rate.

$$T_{dnstr} = T_{upstr} - \varepsilon\left(T_{upstr} - T_{cool}\right)$$
$$q_{ht} = \dot{m}c_p\left(T_{upstr} - T_{dnstr}\right)$$

#### Fluid Flow

Since the block assumes no pressure drop, $P_{flw,in} = P_{vol,out}$.

The flow component connection to the heat exchanger inlet determines the direction of the mass flow. Based on the mass flow rate direction, these temperature and heat flow equations apply.

| Fluid Flow | Mass Flow Rate | Temperatures and Heat Flow |
|---|---|---|
| Forward — From engine flow component to outlet volume | $\dot{m} \geq 0$ | $T_{upstr} = T_{flw,in}$<br>$T_{in} = T_{upstr}$<br>$T_{out} = T_{dnstr}$<br>$q_{out} = \dot{m}c_pT_{dnstr}$ |
| Reverse — From outlet volume to engine flow component | $\dot{m} < 0$ | $T_{upstr} = T_{vol,out}$<br>$T_{in} = T_{dnstr}$<br>$T_{out} = T_{vol,out}$<br>$h_{in} = c_pT_{dnstr}$<br>$q_{out} = \dot{m}h_{vol,out}$ |

The block uses the internal signal `FlwDir` to track the direction of the flow.

### Power Accounting

For the power accounting, the block implements these equations.

| Bus Signal | | | Description | Equations |
|---|---|---|---|---|
| PwrInfo | PwrTrnsfrd — Power transferred between blocks<br>• Positive signals indicate flow into block<br>• Negative signals indicate flow out of block | PwrHeatFlwIn | Heat flow rate at port C | $q_{in}$ |
| | | PwrHeatFlwOut | Heat flow rate at port B | $-q_{out}$ |
| | PwrNotTrnsfrd — Power crossing the block boundary, but not transferred<br>• Positive signals indicate an input<br>• Negative signals indicate a loss | PwrHeatTrnsfr | Heat transfer rate to cooling medium | $-q_{ht}$ |
| | PwrStored — Stored energy rate of change<br>• Positive signals indicate an increase<br>• Negative signals indicate a decrease | | *Not used* | |

## Ports

### Input

#### C — Inlet mass flow rate, heat flow rate, temperature, mass fractions

two-way connector port

Bus containing the heat exchanger:

- MassFlwRate — Mass flow rate at inlet, $\dot{m}$, in kg/s
- HeatFlwRate — Heat flow rate at inlet, $q_{in}$, in J/s
- Temp — Temperature at inlet, $T_{flw,\,in}$, in K
- MassFrac — Inlet mass fractions, dimensionless.

  Specifically, a bus with these mass fractions:

  - O2MassFrac — Oxygen
  - N2MassFrac — Nitrogen
  - UnbrndFuelMassFrac — Unburned fuel
  - CO2MassFrac — Carbon dioxide
  - H2OMassFrac — Water
  - COMassFrac — Carbon monoxide
  - NOMassFrac — Nitric oxide
  - NO2MassFrac — Nitrogen dioxide

- `NOxMassFrac` — Nitric oxide and nitrogen dioxide
- `PmMassFrac` — Particulate matter
- `AirMassFrac` — Air
- `BrndGasMassFrac` — Burned gas

### B — Outlet volume pressure, temperature, enthalpy, mass fractions

two-way connector port

Bus containing the heat exchanger:

- `Prs` — Pressure at outlet, $p_{vol,\,out}$, in Pa
- `Temp` — Temperature at outlet, $T_{vol,\,out}$, in K
- `Enth` — Specific enthalpy at outlet, $h_{vol,\,out}$, in J/kg
- `MassFrac` — Outlet mass fractions, dimensionless.

  Specifically, a bus with these mass fractions:

  - `O2MassFrac` — Oxygen
  - `N2MassFrac` — Nitrogen
  - `UnbrndFuelMassFrac` — Unburned fuel
  - `CO2MassFrac` — Carbon dioxide
  - `H2OMassFrac` — Water
  - `COMassFrac` — Carbon monoxide
  - `NOMassFrac` — Nitric oxide
  - `NO2MassFrac` — Nitrogen dioxide
  - `NOxMassFrac` — Nitric oxide and nitrogen dioxide
  - `PmMassFrac` — Particulate matter
  - `AirMassFrac` — Air
  - `BrndGasMassFrac` — Burned gas

### `Effct` — Heat exchanger effectiveness

`scalar`

Heat exchanger effectiveness, $\varepsilon_{input}$.

#### Dependencies

To create this port, set **Effectiveness model** to `External input`.

### `CoolTemp` — Cooling medium temperature

scalar

Cooling medium temperature, $T_{cool,\,input}$.

#### Dependencies

To create this port, set **Cooling medium temperature input** to `External input`

### Output

#### `Info` — Heat exchanger data

bus

Bus signal containing these block calculations.

| Signal | | | Description | Units |
|---|---|---|---|---|
| `InletTemp` | | | Heat exchanger inlet temperature | K |
| `OutletTemp` | | | Heat exchanger outlet temperature | K |
| `HeatTrnsfrRate` | | | Heat exchanger heat transfer rate | J/s |
| `PwrInfo` | `PwrTrnsfrd` | `PwrHeatFlwIn` | Heat flow rate at port C | W |
| | | `PwrHeatFlwOut` | Heat flow rate at port B | W |
| | `PwrNotTrnsfrd` | `PwrHeatTrnsfr` | Heat transfer rate to cooling medium | W |
| | `PwrStored` | | *Not used* | |

#### `C` — Inlet flow pressure, temperature, enthalpy, mass fractions

two-way connector port

Bus containing the heat exchanger:

- `Prs` — Pressure at inlet, $p_{flw,in}$, in Pa
- `Temp` — Temperature at inlet, $T_{in}$, in K
- `Enth` — Specific enthalpy at inlet, $h_{in}$, in J/kg
- `MassFrac` — Inlet mass fractions, dimensionless.

  Specifically, a bus with these mass fractions:

  - `O2MassFrac` — Oxygen
  - `N2MassFrac` — Nitrogen
  - `UnbrndFuelMassFrac` — Unburned fuel
  - `CO2MassFrac` — Carbon dioxide
  - `H2OMassFrac` — Water
  - `COMassFrac` — Carbon monoxide
  - `NOMassFrac` — Nitric oxide
  - `NO2MassFrac` — Nitrogen dioxide
  - `NOxMassFrac` — Nitric oxide and nitrogen dioxide
  - `PmMassFrac` — Particulate matter
  - `AirMassFrac` — Air
  - `BrndGasMassFrac` — Burned gas

**B — Outlet volume mass flow rate, heat flow rate, temperature, mass fractions**
two-way connector port

Bus containing the heat exchanger:

- `MassFlwRate` — Mass flow rate at outlet, $\dot{m}$, in kg/s
- `HeatFlwRate` — Heat flow rate at outlet, $q_{out}$, in J/s
- `Temp` — Temperature at outlet, $T_{out}$, in K
- `MassFrac` — Outlet mass fractions, dimensionless.

  Specifically, a bus with these mass fractions:

  - `O2MassFrac` — Oxygen
  - `N2MassFrac` — Nitrogen
  - `UnbrndFuelMassFrac` — Unburned fuel
  - `CO2MassFrac` — Carbon dioxide
  - `H2OMassFrac` — Water
  - `COMassFrac` — Carbon monoxide
  - `NOMassFrac` — Nitric oxide
  - `NO2MassFrac` — Nitrogen dioxide
  - `NOxMassFrac` — Nitric oxide and nitrogen dioxide
  - `PmMassFrac` — Particulate matter
  - `AirMassFrac` — Air
  - `BrndGasMassFrac` — Burned gas

## Parameters

#### Block Options

**`Effectiveness model` — Model type for heat effectiveness**
`Constant` (default) | `External input`

Type of model to calculate the heat exchanger effectiveness.

#### Dependencies

Selecting:

- `External input` creates the `Effct` port.
- `Constant` enables the **Heat exchanger effectiveness, ep_cnst** parameter.

**`Cooling medium temperature input` — Specify type**
`Constant` (default) | `External input`

Cooling medium temperature input.

#### Dependencies

Selecting:

- `External input` creates the `CoolTemp` port.
- `Constant` enables the **Cooling medium temperature, T_cool_cnst** parameter.

**`Image type` — Icon color**
`Intercooler` (default) | `EGR cooler hot to cold` | `EGR cooler cold to hot`

Block icon color:

- `Intercooler` for blue, to indicate an intercooler
- `EGR cooler hot to cold` for red to blue, to indicate EGR from hot to cold
- `EGR cooler cold to hot` for blue to red, to indicate EGR from cold to hot

**`Heat exchanger effectiveness, ep_cnst` — Effectiveness**
`0.7` (default) | `scalar`

Constant heat exchanger effectiveness, $\varepsilon_{cnst}$.

**Dependencies**

To enable this parameter, select `Constant` for the **Effectiveness model** parameter.

**`Cooling medium temperature, T_cool_cnst` — Temperature**
`300` (default) | `scalar`

Constant cooling medium temperature, $T_{cool,\,cnst}$, in K.

**Dependencies**

To enable this parameter, select `Constant` for the **Cooling medium temperature input** parameter.

**`Specific heat at constant pressure, cp` — Specific heat**
`1005` (default) | `scalar`

Specific heat at constant pressure, $c_p$, in J/(kg*K).

## References

[1] Eriksson, Lars and Nielsen, Lars. *Modeling and Control of Engines and Drivelines*. Chichester, West Sussex, United Kingdom: John Wiley & Sons Ltd, 2014.

## Extended Capabilities

**C/C++ Code Generation**
Generate C and C++ code using Simulink® Coder™.

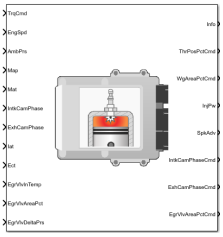
## See Also
Control Volume System | Flow Restriction

**Introduced in R2017a**

# SI Controller

Spark-ignition engine controller that uses the driver torque request

**Library:** Powertrain Blockset / Propulsion / Combustion Engine Controllers



## Description

The SI Controller block implements a spark-ignition (SI) controller that uses the driver torque request to calculate the open-loop air, fuel, and spark actuator commands that are required to meet the driver demand.

You can use the SI Controller block in engine control design or performance, fuel economy, and emission tradeoff studies. The core engine, throttle, and turbocharger wastegate subsystems require the commands that are output from the SI Controller block.

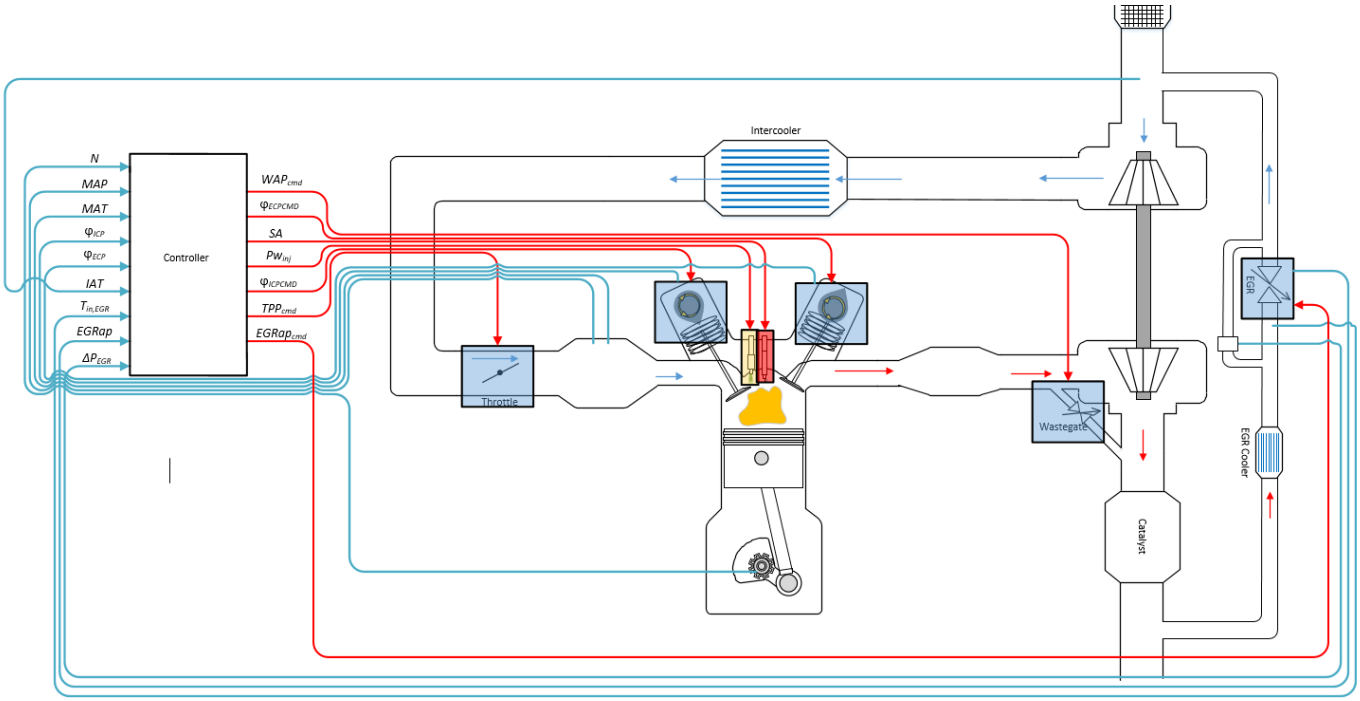
The block uses the commanded torque and engine speed to determine these open-loop actuator commands:

- Throttle position percent
- Wastegate area percent
- Injector pulse-width
- Spark advance
- Intake cam phaser angle
- Exhaust cam phaser angle
- Exhaust gas recirculation (EGR) valve area percent

The SI Controller block has two subsystems:

- The `Controller` subsystem — Determines the commands based on the commanded torque, measured engine speed, and estimated cylinder air mass.
- The `Estimator` subsystem — Determines the estimated air mass flow, torque, and exhaust gas temperature from intake manifold gas pressure, intake manifold gas temperature, engine speed, and cam phaser positions.

The figure illustrates the signal flow.




The figure uses these variables.

| | |
|---|---|
| $N$ | Engine speed |
| $MAP$ | Cycle average intake manifold pressure |
| $IAT$ | Intake air temperature |
| $T_{in,EGR}$ | Temperature at EGR valve inlet |
| $MAT$ | Cycle average intake manifold gas absolute temperature |
| $\varphi_{ICP}$, $\varphi_{ICPCMD}$ | Intake cam phaser angle and intake cam phaser angle command, respectively |
| $\varphi_{ECP}$, $\varphi_{ECPCMD}$ | Exhaust cam phaser angle and exhaust cam phaser angle command, respectively |
| $EGRap$, $EGRap_{cmd}$ | EGR valve area percent and EGR valve area percent command, respectively |
| $\Delta P_{EGR}$ | Pressure difference at EGR valve inlet and outlet |
| $WAP_{cmd}$ | Turbocharger wastegate area percent command |
| $SA$ | Spark advance |
| $Pw_{inj}$ | Fuel injector pulse-width |
| $TPP_{cmd}$ | Throttle position percent command |

The Model-Based Calibration Toolbox was used to develop the tables that are available with the Powertrain Blockset.

### Controller

#### Air

The block determines the commanded engine load (that is, normalized cylinder air mass) from a lookup table that is a function of commanded torque and measured engine speed.

$$L_{cmd} = f_{Lcmd}(T_{cmd}, N)$$

To achieve the commanded load, the controller sets the throttle position percent and turbocharger wastegate area percent using feed forward lookup tables. The lookup tables are functions of the commanded load and measured engine speed.

$$TAP_{cmd} = f_{TAPcmd}(L_{cmd}, N)$$

$$TPP_{cmd} = f_{TPPcmd}(TAP_{cmd})$$

$$WAP_{cmd} = f_{WAPcmd}(L_{cmd}, N)$$

To determine the cam phaser angle commands, the block uses lookup tables that are functions of estimated engine load and measured engine speed.

$$\varphi_{ICPCMD} = f_{ICPCMD}(L_{est}, N)$$

$$\varphi_{ECPCMD} = f_{ECPCMD}(L_{est}, N)$$

The block calculates the desired engine load using this equation.

$$L_{est} = \frac{CpsR_{air}T_{std}\dot{m}_{air,est}}{P_{std}V_dN}$$

The equations use these variables.

| | |
|---|---|
| $L_{est}$ | Estimated engine load |
| $L_{cmd}$ | Commanded engine load |
| $N$ | Engine speed |
| $T_{cmd}$ | Commanded engine torque |
| $TAP_{cmd}$ | Throttle area percent command |
| $TPP_{cmd}$ | Throttle position percent command |
| $WAP_{cmd}$ | Turbocharger wastegate area percent command |
| $Cps$ | Crankshaft revolutions per power stroke |
| $P_{std}$ | Standard pressure |
| $T_{std}$ | Standard temperature |
| $R_{air}$ | Ideal gas constant for air and burned gas mixture |
| $V_d$ | Displaced volume |
| $\dot{m}_{air,est}$ | Estimated engine air mass flow |

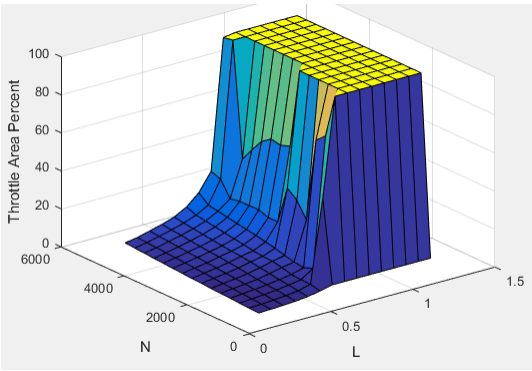
The controller subsystem uses these lookup tables for the air calculations.

- The throttle area percent command lookup table, $f_{TAPcmd}$, is a function of commanded load and engine speed

  $$TAP_{cmd} = f_{TAPcmd}(L_{cmd}, N)$$

  where:

- $TAP_{cmd}$ is throttle area percentage command, in percent.
- $L_{cmd}$=$L$ is commanded engine load, dimensionless.
- $N$ is engine speed, in rpm.



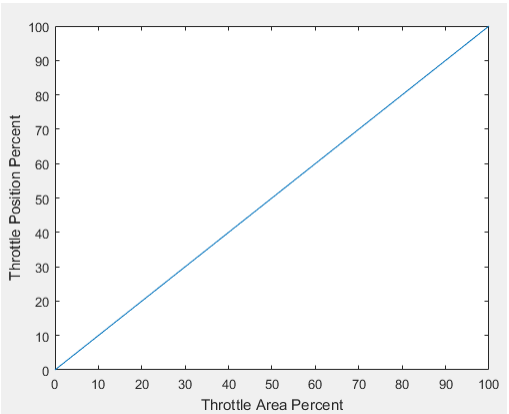

- To account for the non-linearity of the throttle position to throttle area, the throttle position percent lookup table linearizes the open-loop air mass flow control.

  The throttle position percent command lookup table, $f_{TPPcmd}$, is a function of the throttle area percentage command

  $$TPP_{cmd} = f_{TPPcmd}(TAP_{cmd})$$

  where:

  - $TPP_{cmd}$ is throttle position percentage command, in percent.
  - $TAP_{cmd}$ is throttle area percentage command, in percent.



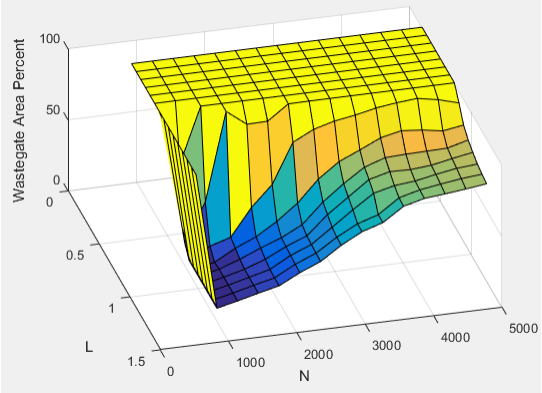

- The wastegate area percent command lookup table, $f_{WAPcmd}$, is a function of the commanded engine load and engine speed

  $$WAP_{cmd} = f_{WAPcmd}(L_{cmd}, N)$$

  where:

  - $WAP_{cmd}$ is wastegate area percentage command, in percent.
  - $L_{cmd}$=$L$ is commanded engine load, dimensionless.

- $N$ is engine speed, in rpm.

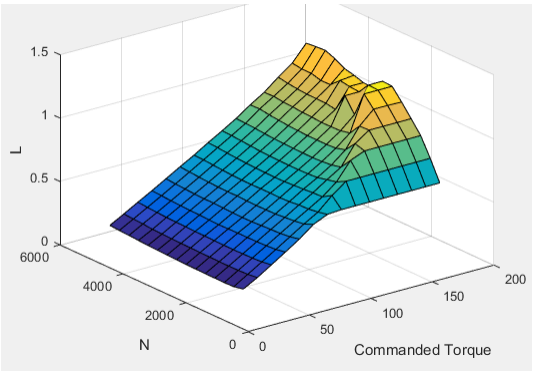



- The commanded engine load lookup table, $f_{Lcmd}$, is a function of the commanded torque and engine speed

  $$L_{cmd} = f_{Lcmd}(T_{cmd}, N)$$

  where:

  - $L_{cmd}$=$L$ is commanded engine load, dimensionless.
  - $T_{cmd}$ is commanded torque, in N·m.
  - $N$ is engine speed, in rpm.

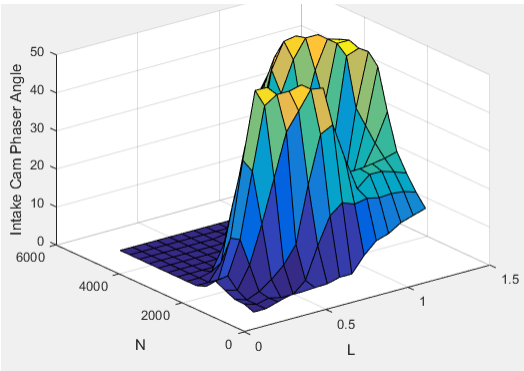



- The intake cam phaser angle command lookup table, $f_{ICPCMD}$, is a function of the engine load and engine speed

  $$\varphi_{ICPCMD} = f_{ICPCMD}(L_{est}, N)$$

  where:

  - $\varphi_{ICPCMD}$ is commanded intake cam phaser angle, in degrees crank advance.
  - $L_{est}$=$L$ is estimated engine load, dimensionless.
  - $N$ is engine speed, in rpm.

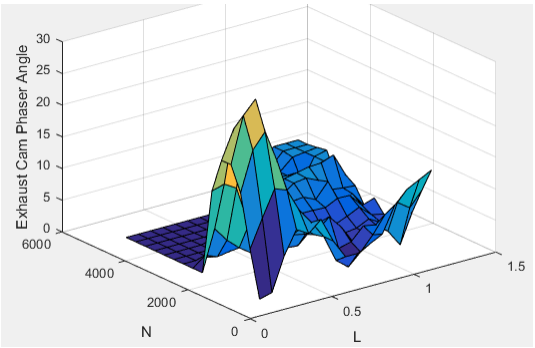



- The exhaust cam phaser angle command lookup table, $f_{ECPCMD}$, is a function of the engine load and engine speed

  $$\varphi_{ECPCMD} = f_{ECPCMD}(L_{est}, N)$$

  where:

  - $\varphi_{ECPCMD}$ is commanded exhaust cam phaser angle, in degrees crank retard.
  - $L_{est}$=$L$ is estimated engine load, dimensionless.
  - $N$ is engine speed, in rpm.



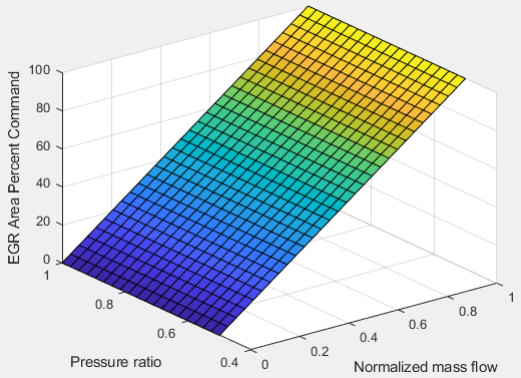

**EGR**

EGR is typically expressed as a percent of total intake port flow.

$$EGR_{pct} = 100\frac{\dot{m}_{EGR}}{\dot{m}_{EGR} + \dot{m}_{air}}$$

To calculate the EGR area percent command, the block uses equations and a lookup table.

| | |
|---|---|
| Equations | $\dot{m}_{EGRstd,\,cmd} = \dot{m}_{EGR,\,cmd}\frac{P_{std}}{P_{in,\,EGR}}\sqrt{\frac{T_{in,\,EGR}}{T_{std}}}$<br>$\dot{m}_{EGRstd,\,max} = f_{EGRstd,\,max}\left(\frac{P_{out,\,EGR}}{P_{in,\,EGR}}\right)$<br>$\dot{m}_{EGR,\,cmd} = EGR_{pct,\,cmd}\dot{m}_{intk,\,est}$ |

| | |
|---|---|
| Lookup table | The EGR area percent command, $EGRap_{cmd}$, lookup table is a function of the normalized mass flow and pressure ratio<br>$$EGRap_{cmd} = f_{EGRap,\,cmd}\left(\frac{\dot{m}_{EGRstd,\,cmd}}{\dot{m}_{EGRstd,\,max}}, \frac{P_{out,\,EGR}}{P_{in,\,EGR}}\right)$$<br>where:<br>• $EGRap_{cmd}$ is commanded EGR area percent, dimensionless.<br>• $\frac{\dot{m}_{EGRstd,\,cmd}}{\dot{m}_{EGRstd,\,max}}$ is the normalized mass flow, dimensionless.<br>• $\frac{P_{out,\,EGR}}{P_{in,\,EGR}}$ is the pressure ratio, dimensionless.<br>  |

The equations and table use these variables.

| | |
|---|---|
| $EGRap$, $EGRap_{cmd}$ | EGR valve area percent and EGR valve area percent command, respectively |
| $EGR_{pct,cmd}$ | EGR percent command |
| $\dot{m}_{EGRstd,\,cmd}$ | Commanded standard mass flow |
| $\dot{m}_{EGRstd,\,max}$ | Maximum standard mass flow |
| $\dot{m}_{EGR,\,cmd}$ | Commanded mass flow |
| $\dot{m}_{intk,\,est}$ | Estimated intake port mass flow |
| $T_{std}$, $P_{std}$ | Standard temperature and pressure |
| $T_{in,EGR}$ | Temperature at EGR valve inlet |
| $P_{out,EGR}$, $P_{in,EGR}$ | Pressure at EGR valve inlet and outlet, respectively |

**Fuel**

The air-fuel ratio (AFR) impacts three-way-catalyst (TWC) conversion efficiency, torque production, and combustion temperature. The engine controller manages AFR by commanding injector pulse-width from a desired relative AFR. The relative AFR, $\lambda_{cmd}$, is the ratio between the commanded AFR and the stoichiometric AFR of the fuel.

$$\lambda_{cmd} = \frac{AFR_{cmd}}{AFR_{stoich}}$$

$$AFR_{cmd} = \frac{\dot{m}_{air,est}}{\dot{m}_{fuel,cmd}}$$

The SI Controller block accounts for the extra fuel delivered to the SI engine during startup. If the engine speed is greater than the startup engine cranking speed, the SI Controller block enriches the optimal AFR, lambda, with an exponentially decaying delta lambda. To initialize the delta lambda, the block uses the engine coolant temperature at startup. The delta lambda exponentially decays to zero based on a time constant that is a function of the engine coolant temperature.

You can configure the block for open-loop and closed-loop AFR control.

| To | Use | Controls > Fuel > Closed-loop feedback Parameter Setting |
|---|---|---|
| • Assess the dynamic and steady-state accuracy of the controller airflow estimation and fuel delivery. | (default) Open-loop control | `off` |
| • Hold the average AFR close to stoichiometric AFR to maintain a high TWC conversion efficiency. | Closed-loop control | `on` |

**Open-Loop Control**

To create an input port for the commanded AFR (lambda), on the **Controls** > **Fuel** > **Open-loop fuel** pane, select **Input lambda**.

You can manually tune the catalyst for maximum efficiency during open-loop AFR control with or without dither. If you want to implement dither during open-loop control, on the **Fuel** tab, on the **Closed-loop fuel** pane, select **Dither**.

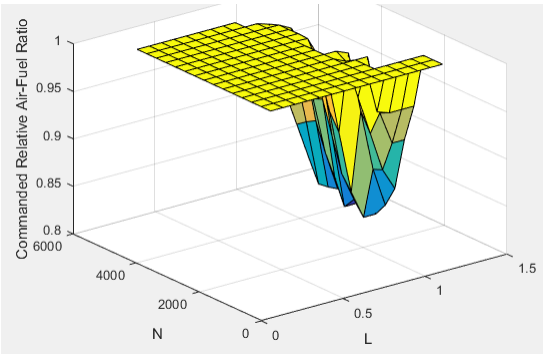
By default, the block is configured to use a lookup table for the commanded AFR.

The commanded lambda, $\lambda_{cmd}$, lookup table is a function of estimated engine load and measured engine speed

$$\lambda_{cmd} = f_{\lambda cmd}(L_{est}, N)$$

where:

- $\lambda_{cmd}$ is commanded relative AFR, dimensionless.
- $L_{est}$=$L$ is estimated engine load, dimensionless.
- $N$ is engine speed, in rpm.




The block calculates the estimated fuel mass flow rate using the commanded lambda, $\lambda_{cmd}$, stoichiometric AFR, and estimated air mass flow rate.

$$\dot{m}_{fuel,cmd} = \frac{\dot{m}_{air,est}}{AFR_{cmd}} = \frac{\dot{m}_{air,est}}{\lambda_{cmd} AFR_{stoich}}$$

The block assumes that the battery voltage and fuel pressure are at nominal settings where pulse-width correction is not necessary. The commanded fuel injector pulse-width is proportional to the fuel mass per injection. The fuel mass per injection is calculated from the commanded fuel mass flow rate, engine speed, and the number of cylinders.

$$Pw_{inj} = \begin{cases} \frac{\dot{m}_{fuel,cmd} Cps \left(\frac{60s}{min}\right)\left(\frac{1000mg}{g}\right)\left(\frac{1000g}{kg}\right)}{NS_{inj} N_{cyl}} & \text{when } Trq_{cmd} > 0 \\ 0 & \text{when } Trq_{cmd} \le 0 \end{cases}$$

### Closed-Loop Control

TWC converters are most efficient when the exhaust AFR is near the stoichiometric AFR, where the air and fuel burn most completely. Around this ideal point, the AFR is within the *catalyst window* in which the catalyst is most efficient at converting carbon monoxide, hydrocarbons, and nitrogen oxides to non-harmful exhaust products. Empirical studies show that oscillating the AFR around stoichiometry at an optimized AFR frequency, amplitude, and bias widens the TWC window, increasing catalyst conversion efficiency in the presence of unavoidable disturbances.

To keep production hardware costs down, AFR control systems include inexpensive switching oxygen sensors positioned in the engine exhaust stream upstream and downstream of the catalyst. The oxygen sensors have a narrow range. Essentially, they switch between too lean (i.e., more air is available than is required to burn the available fuel) and too rich (i.e., more air is available than is required to burn the available fuel).

The block implements a period-based method to control the average AFR at a value within the catalyst window for maximum conversion efficiency. Period-based AFR control is independent of the transport delay across the engine from the fuel injection point to the sensor measurement point. For more information about the method, see Developing a Period-Based Air-Fuel Ratio Controller Using a Low-Cost Switching Sensor.

#### Spark

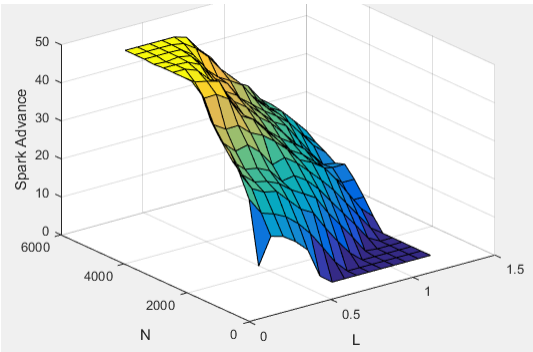
Spark advance is the crank angle before top dead center (BTDC) of the power stroke when the spark is delivered. The spark advance has an impact on engine efficiency, torque, exhaust temperature, knock, and emissions.

The spark advance lookup table is a function of estimated load and engine speed.

$$SA = f_{SA}(L_{est}, N)$$

where:

- $SA$ is spark advance, in crank advance degrees.
- $L_{est}$=$L$ is estimated engine load, dimensionless.
- $N$ is engine speed, in rpm.




The equations use these variables.

| | |
|---|---|
| $L_{est}$ | Estimated engine load, based on normalized cylinder air mass |
| $N$ | Engine speed |
| $f_{SA}$ | Lookup table for spark advance |
| $N$ | Spark advance |

#### Idle Speed

When the commanded torque is below a threshold value, the idle speed controller regulates the engine speed.

| If | Idle Speed Controller |
|---|---|
| $Trq_{cmd,input} < Trq_{idlecmd,enable}$ | Enabled |
| $Trq_{idlecmd,enable} \le Trq_{cmd,input}$ | Not enabled |

The idle speed controller uses a discrete PI controller to regulate the target idle speed by commanding a torque.

The PI controller uses this transfer function:

$$C_{idle}(z) = K_{p,\,idle} + K_{i,\,idle}\frac{t_S}{z-1}$$

The idle speed commanded torque must be less than the maximum commanded torque:

$0 \le Trq_{idlecomd} \le Trq_{idlecmd,max}$

Idle speed control is active under these conditions. If the commanded input torque drops below the threshold for enabling the idle speed controller ($Trq_{cmd,input} < Trq_{idlecmd,enable}$), the commanded engine torque is given by:

$Trq_{cmd} = \max(Trq_{cmd,input}, Trq_{idlecmd})$.

The equations use these variables.

| | |
|---|---|
| $Trq_{cmd}$ | Commanded engine torque |
| $Trq_{cmd,input}$ | Input commanded engine torque |
| $Trq_{idlecmd,enable}$ | Threshold for enabling idle speed controller |
| $Trq_{idlecmd}$ | Idle speed controller commanded torque |
| $Trq_{idlecmd,max}$ | Maximum commanded torque |
| $N_{idle}$ | Base idle speed |
| $K_{p,idle}$ | Idle speed controller proportional gain |
| $K_{i,idle}$ | Idle speed controller integral gain |

#### Speed Limiter

To prevent over revving the engine, the block implements an engine speed limit controller that limits the engine speed to the value specified by the **Rev-limiter speed threshold** parameter on the **Controls** > **Idle Speed** tab.

If the engine speed, $N$, exceeds the engine speed limit, $N_{lim}$, the block sets the commanded engine torque to 0.

To smoothly transition the torque command to 0 as the engine speed approaches the speed limit, the block implements a lookup table multiplier. The lookup table multiplies the torque command by a value that ranges from 0 (engine speed exceeds limit) to 1 (engine speed does not exceed the limit).

### Estimator

The estimator subsystem determines the estimated air mass flow, torque, EGR mass flow, and exhaust temperature based on sensor feedback and calibration parameters.

| | |
|---|---|
| $\dot{m}_{air,est}$ | Estimated engine air mass flow |
| $Trq_{est}$ | Estimated engine torque |
| $T_{exh,est}$ | Estimated engine exhaust temperature |
| $\dot{m}_{EGR,est}$ | Estimated low-pressure EGR mass flow |

#### Air Mass Flow

To calculate engine air mass flow, configure the SI engine to use either of these air mass flow models.

| Air Mass Flow Model | Description |
|---|---|
| “SI Engine Speed-Density Air Mass Flow Model” | Uses the speed-density equation to calculate the engine air mass flow, relating the engine air mass flow to the intake manifold pressure and engine speed. Consider using this air mass flow model in engines with fixed valvetrain designs. |

| Air Mass Flow Model | Description |
|---|---|
| “SI Engine Dual-Independent Cam Phaser Air Mass Flow Model” | To calculate the engine air mass flow, the dual-independent cam phaser model uses:<br>• Empirical calibration parameters developed from engine mapping measurements<br>• Desktop calibration parameters derived from engine computer-aided design (CAD) data<br>In contrast to typical embedded air mass flow calculations based on direct air mass flow measurement with an air mass flow (MAF) sensor, this air mass flow model offers:<br>• Elimination of MAF sensors in dual cam-phased valvetrain applications<br>• Reasonable accuracy with changes in altitude<br>• Semiphysical modeling approach<br>• Bounded behavior<br>• Suitable execution time for electronic control unit (ECU) implementation<br>• Systematic development of a relatively small number of calibration parameters |

To determine the estimated air mass flow, the block uses the intake air mass fraction. The EGR mass fraction at the intake port lags the mass fraction near the EGR valve outlet. To model the lag, the block uses a first order system with a time constant.

$$y_{intk,EGR,est} = \frac{\dot{m}_{EGR,est}}{\dot{m}_{intk,est}} \frac{t_s z}{\tau_{EGR} z + t_s - \tau_{EGR}}$$

The remainder of the gas is air.

$$y_{intk,air,est} = 1 - y_{intk,EGR,est}$$

The equations use these variables.

| | |
|---|---|
| $y_{intk,EGR,est}$ | Estimated intake manifold EGR mass fraction |
| $y_{intk,air,est}$ | Estimated intake manifold air mass fraction |
| $\dot{m}_{EGR,est}$ | Estimated low-pressure EGR mass flow |
| $\dot{m}_{intk,est}$ | Estimated intake port mass flow |
| $\tau_{EGR}$ | EGR time constant |

**Torque**

To calculate the brake torque, configure the SI engine to use either of these torque models.

| Brake Torque Model | Description |
|---|---|
| “SI Engine Torque Structure Model” | For the structured brake torque calculation, the SI engine uses tables for the inner torque, friction torque, optimal spark, spark efficiency, and lambda efficiency.<br><br>If you select **Crank angle pressure and torque** on the block **Torque** tab, you can:<br><br>• Simulate advanced closed-loop engine controls in desktop simulations and on HIL bench, based on cylinder pressure recorded from a model or laboratory test as a function of crank angle.<br>• Simulate driveline vibrations downstream of the engine due to high-frequency crankshaft torsionals.<br>• Simulate engine misfires due to lean operation or spark plug fouling by using the injector pulse width input.<br>• Simulate cylinder deactivation effect (closed intake and exhaust valves, no injected fuel) on individual cylinder pressures, mean-value airflow, mean-value torque, and crank-angle-based torque.<br>• Simulate the fuel-cut effect on individual cylinder pressure, mean-value torque, and crank-angle-based torque. |
| “SI Engine Simple Torque Model” | For the simple brake torque calculation, the SI engine block uses a torque lookup table map that is a function of engine speed and load. |

#### EGR

The controller estimates low-pressure mass flow, EGR valve inlet pressure, and EGR valve outlet pressure using an algorithm developed by F. Liu and J. Pfeiffer. The estimator requires measured EGR valve differential pressure, EGR valve area percent, intake air temperature, and EGR valve inlet temperature.

To estimate the EGR valve commands, the block uses:

- Equations

  $$\dot{m}_{air,std} = \dot{m}_{air,est}\frac{P_{std}}{P_{amb}}\sqrt{\frac{IAT}{T_{std}}}$$

  $$P_{in,EGR} = P_{out,EGR} + \Delta P_{EGR}$$

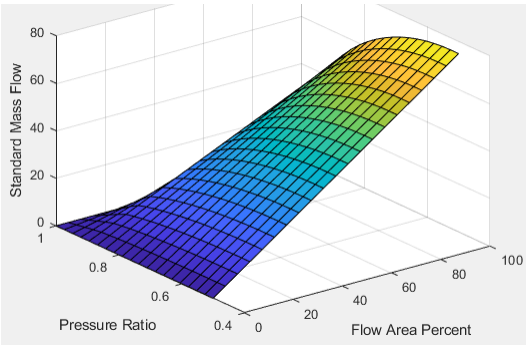
  $$\dot{m}_{EGR,est} = \dot{m}_{EGR,std}\frac{P_{in,EGR}}{P_{std}}\sqrt{\frac{T_{std}}{T_{in,EGR}}}$$

- Tables
  - The EGR valve standard mass flow lookup table is a function of EGR valve area percent and the pressure ratio

$$\dot{m}_{EGR,\,std} = f_{EGR,\,std}\left(EGRap, \frac{P_{out,\,EGR}}{P_{in,\,EGR}}\right)$$

where:

- $\dot{m}_{EGR,\,std}$ is EGR valve standard mass flow, dimensionless.
- $EGRap$ is EGR valve flow area percent, in percent.
- $\frac{P_{out,\,EGR}}{P_{in,\,EGR}}$ is the pressure ratio, dimensionless.

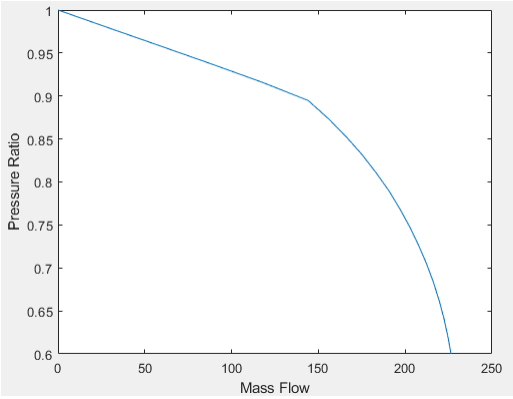



- The pressure ratio is a function of the standard mass flow

$$\frac{P_{out,\,EGR}}{P_{amb}} = f_{intksys,\,pr}(\dot{m}_{air,\,std})$$

where:

- $\dot{m}_{air,\,std}$ is standard mass flow, in g/s.
- $\frac{P_{out,\,EGR}}{P_{amb}}$ is pressure ratio, dimensionless.




The equations use these variables.

| | |
|---|---|
| *EGRap* | EGR valve area percent command |
| *IAT* | Intake air temperature |

| | |
|---|---|
| $\dot{m}_{air,std}$, $\dot{m}_{EGR,std}$ | Standard air and EGR valve mass flow, respectively |
| $\dot{m}_{air,est}$,$\dot{m}_{EGR,est}$ | Estimated air and EGR valve mass flow, respectively |
| $T_{std}$, $P_{std}$ | Standard temperature and pressure |
| $T_{amb}$, $P_{amb}$ | Ambient temperature and pressure |
| $\Delta P_{EGR}$ | Pressure difference at EGR valve inlet and outlet |
| $T_{in,EGR}$, $T_{out,EGR}$ | Temperature at EGR valve inlet and outlet, respectively |
| $P_{in,EGR}$, $P_{out,EGR}$ | Pressure at EGR valve inlet and outlet, respectively |

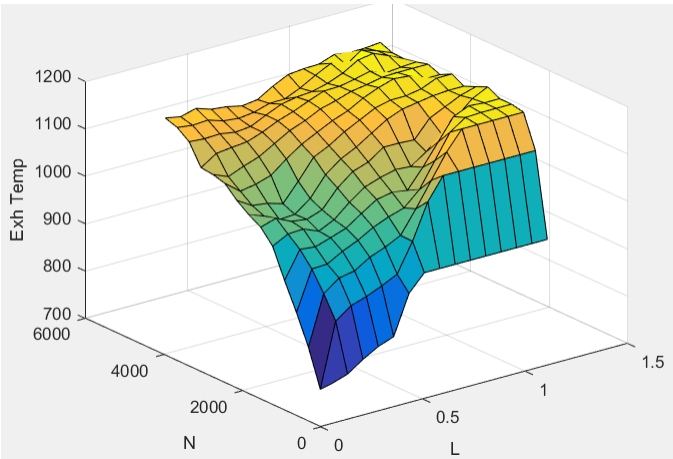
#### Exhaust Temperature

The exhaust temperature lookup table, $f_{Texh}$, is a function of engine load and engine speed

$$T_{exh} = f_{Texh}(L, N)$$

where:

- $T_{exh}$ is engine exhaust temperature, in K.
- $L$ is normalized cylinder air mass or engine load, dimensionless.
- $N$ is engine speed, in rpm.




## Ports

### Input

#### `TrqCmd` — Commanded engine torque

`scalar`

Commanded engine torque, $Trq_{cmd,input}$, in N·m.

#### `EngSpd` — Measured engine speed

`scalar`

Measured engine speed, $N$, in rpm.

**`AmbPrs` — Measured absolute ambient pressure**
`scalar`

Measured ambient pressure, $P_{Amb}$, in Pa.

**`Map` — Measured intake manifold absolute pressure**
`scalar`

Measured intake manifold absolute pressure$MAP$, in Pa.

**`Mat` — Measured intake manifold absolute temperature**
`scalar`

Measured intake manifold absolute temperature, $MAT$, in K.

**`IntkCamPhase` — Intake cam phaser angle**
`scalar`

Intake cam phaser angle, $\varphi_{ICP}$, in degCrkAdv, or degrees crank advance.

**`ExhCamPhase` — Exhaust cam phaser angle**
`scalar`

Exhaust cam phaser angle, $\varphi_{ECP}$, in degCrkRet, or degrees crank retard.

**`Iat` — Intake air temperature**
`scalar`

Intake air temperature, $IAT$, in K.

**`Ect` — Engine cooling temperature**
`scalar`

Engine cooling temperature, $T_{coolant}$, in K.

**`EgrVlvInTemp` — EGR valve inlet temperature**
`scalar`

EGR valve inlet temperature, $T_{in,EGR}$, in K.

**`EgrVlvAreaPct` — EGR valve area percent**
`scalar`

EGR valve area percent, $EGRap$, in %.

**`EgrVlvDeltaPrs` — EGR valve delta pressure**
`scalar`

EGR valve delta pressure, $\Delta P_{EGR}$, in Pa.

**`O2VoltSen` — Oxygen sensor voltage**
`scalar`

Oxygen sensor voltage for closed-loop air-fuel-ratio (lambda) control, in mV.

To configure the block to use closed-loop air-fuel-ratio control, on the **Fuel** tab, on the **Closed-loop fuel** pane, select **Closed-loop feedback**.

#### `LambdaCmd` — Commanded AFR, lambda

scalar

Commanded air-fuel-ratio (lambda), $\lambda_{cmd}$, dimensionless.

##### Dependencies

To create this port, on the **Fuel** tab, on the **Open-loop fuel** pane, select **Input lambda**.

### Output

#### `Info` — Bus signal

bus

Bus signal containing these block calculations.

| Signal | Description | Variable | Units |
|---|---|---|---|
| `TrqCmd` | Engine torque | $Trq_{cmd}$ | N·m |
| `LdCmd` | Commanded load | $L_{cmd}$ | N/A |
| `ThrPosCmd` | Throttle area percent command | $TAP_{cmd}$ | % |
| `WgAreaPctCmd` | Wastegate area percent command | $WAP_{cmd}$ | % |
| `InjPw` | Fuel injector pulse-width | $Pw_{inj}$ | ms |
| `SpkAdv` | Spark advance | $SA$ | degBTDC |
| `IntkCamPhaseCmd` | Intake cam phaser angle command | $\varphi_{ICPCMD}$ | degCrkAdv |
| `ExhCamPhaseCmd` | Exhaust cam phaser angle command | $\varphi_{ECPCMD}$ | degCrkRet |
| `EgrVlvAreaPctCmd` | Exhaust cam phaser angle command | $EGRap_{cmd}$ | % |
| `FuelMassFlwCmd` | EGR valve area percent command | $\dot{m}_{fuel,\,cmd}$ | kg/s |
| `AfrCmd` | Commanded air-fuel ratio | $AFR_{cmd}$ | N/A |
| `EstEngTrq` | Estimated engine torque | $Trq_{est}$ | N·m |
| `EstNrmlzdAirCharg` | Estimated normalized cylinder air mass | N/A | N/A |
| `EstIntkPortMassFlw` | Estimated intake port air mass flow rate | $\dot{m}_{intk,\,est}$ | kg/s |
| `EstIntkAirMassFlw` | Estimated air mass flow rate | $\dot{m}_{air,\,est}$ | kg/s |
| `EstEgrMassFlw` | Estimated low-pressure EGR mass flow rate | $\dot{m}_{EGR,\,est}$ | kg/s |
| `EstExhManGasTemp` | Estimated exhaust manifold gas temperature | $T_{exh,est}$ | K |
| `EngRevLimAct` | Flag that indicates if rev-limiter control is active | N/A | N/A |
| `ClsdLpFuelMult` | Fuel injector pulse-width multiplier for closed-loop AFR control | $Pw_{inj_mult}$ | N/A |

#### `ThrPosPctCmd` — Throttle area percent command

scalar

Throttle area percent command, $TAP_{cmd}$.

**`WgAreaPctCmd` — Wastegate area percent command**
`scalar`

Wastegate area percent command, $WAP_{cmd}$.

**`InjPw` — Fuel injector pulse-width**
`scalar`

Fuel injector pulse-width, $Pw_{inj}$, in ms.

**`SpkAdv` — Spark advance**
`scalar`

Spark advance, $SA$, in degrees crank angle before top dead center (degBTDC).

**`IntkCamPhaseCmd` — Intake cam phaser angle command**
`scalar`

Intake cam phaser angle command, $\varphi_{ICPCMD}$.

**`ExhCamPhaseCmd` — Exhaust cam phaser angle command**
`scalar`

Exhaust cam phaser angle command, $\varphi_{ECPCMD}$.

**`EgrVlvAreaPctCmd` — EGR valve area percent command**
`scalar`

EGR valve area percent command, $EGRap_{cmd}$, in %.

## Parameters

### Configuration

**`Air mass flow estimation model` — Select air mass flow estimation model**
`Dual Variable Cam Phasing` (default) | `Simple Speed-Density`

To calculate engine air mass flow, configure the SI engine to use either of these air mass flow models.

| Air Mass Flow Model | Description |
|---|---|
| “SI Engine Speed-Density Air Mass Flow Model” | Uses the speed-density equation to calculate the engine air mass flow, relating the engine air mass flow to the intake manifold pressure and engine speed. Consider using this air mass flow model in engines with fixed valvetrain designs. |

| Air Mass Flow Model | Description |
|---|---|
| “SI Engine Dual-Independent Cam Phaser Air Mass Flow Model” | To calculate the engine air mass flow, the dual-independent cam phaser model uses:<br>• Empirical calibration parameters developed from engine mapping measurements<br>• Desktop calibration parameters derived from engine computer-aided design (CAD) data<br>In contrast to typical embedded air mass flow calculations based on direct air mass flow measurement with an air mass flow (MAF) sensor, this air mass flow model offers:<br>• Elimination of MAF sensors in dual cam-phased valvetrain applications<br>• Reasonable accuracy with changes in altitude<br>• Semiphysical modeling approach<br>• Bounded behavior<br>• Suitable execution time for electronic control unit (ECU) implementation<br>• Systematic development of a relatively small number of calibration parameters |

#### Dependencies

The table summarizes the parameter dependencies.

| Air Mass Flow Estimation Model | Enables Parameters on Estimation > Air Tab |
|---|---|
| `Dual Variable Cam Phasing` | **Cylinder volume at intake valve close table, f_vivc**<br>**Cylinder volume intake cam phase breakpoints, f_vivc_icp_bpt**<br>**Cylinder trapped mass correction factor, f_tm_corr**<br>**Normalized density breakpoints, f_tm_corr_nd_bpt**<br>**Engine speed breakpoints, f_tm_corr_n_bpt**<br>**Air mass flow, f_mdot_air**<br>**Exhaust cam phase breakpoints, f_mdot_air_ecp_bpt**<br>**Trapped mass flow breakpoints, f_mdot_trpd_bpt**<br>**Air mass flow correction factor, f_mdot_air_corr**<br>**Engine load breakpoints for air mass flow correction, f_mdot_air_corr_ld_bpt**<br>**Engine speed breakpoints for air mass flow correction, f_mdot_air_n_bpt** |

| Air Mass Flow Estimation Model | Enables Parameters on Estimation > Air Tab |
|---|---|
| `Simple Speed-Density` | **Speed-density volumetric efficiency, f_nv**<br><br>**Speed-density intake manifold pressure breakpoints, f_nv_prs_bpt**<br><br>**Speed-density engine speed breakpoints, f_nv_n_bpt** |

**`Torque estimation model` — Select torque estimation model**
`Torque Structure` (default) | `Simple Torque Lookup`

To calculate the brake torque, configure the SI engine to use either of these torque models.

| Brake Torque Model | Description |
|---|---|
| “SI Engine Torque Structure Model” | For the structured brake torque calculation, the SI engine uses tables for the inner torque, friction torque, optimal spark, spark efficiency, and lambda efficiency.<br><br>If you select **Crank angle pressure and torque** on the block **Torque** tab, you can:<br><br>• Simulate advanced closed-loop engine controls in desktop simulations and on HIL bench, based on cylinder pressure recorded from a model or laboratory test as a function of crank angle.<br>• Simulate driveline vibrations downstream of the engine due to high-frequency crankshaft torsionals.<br>• Simulate engine misfires due to lean operation or spark plug fouling by using the injector pulse width input.<br>• Simulate cylinder deactivation effect (closed intake and exhaust valves, no injected fuel) on individual cylinder pressures, mean-value airflow, mean-value torque, and crank-angle-based torque.<br>• Simulate the fuel-cut effect on individual cylinder pressure, mean-value torque, and crank-angle-based torque. |
| “SI Engine Simple Torque Model” | For the simple brake torque calculation, the SI engine block uses a torque lookup table map that is a function of engine speed and load. |

**Dependencies**

The table summarizes the parameter dependencies.

| Torque Estimation Model | Enables Parameters on Estimation > Torque Tab |
|---|---|
| `Torque Structure` | **Inner torque table, f_tq_inr**<br>**Friction torque table, f_tq_fric**<br>**Engine temperature modifier on friction torque, f_fric_temp_mod**<br>**Engine temperature modifier breakpoints, f_fric_temp_bpt**<br>**Pumping torque table, f_tq_pump**<br>**Optimal spark table, f_sa_opt**<br>**Inner torque load breakpoints, f_tq_inr_l_bpt**<br>**Inner torque speed breakpoints, f_tq_inr_n_bpt**<br>**Spark efficiency table, f_m_sa**<br>**Spark retard from optimal, f_del_sa_bpt**<br>**Lambda efficiency, f_m_lam**<br>**Lambda breakpoints, f_m_lam_bpt** |
| `Simple Torque Lookup` | **Torque table, f_tq_nl**<br>**Torque table load breakpoints, f_tq_nl_l_bpt**<br>**Torque table speed breakpoints, f_tq_nl_n_bpt** |

### Controls

#### Air

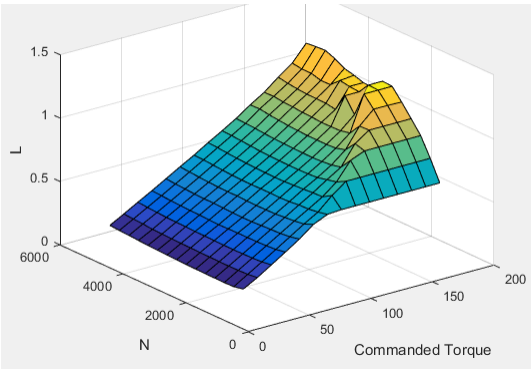
**`Engine commanded load table, f_lcmd` — Lookup table**
`array`

The commanded engine load lookup table, $f_{Lcmd}$, is a function of the commanded torque and engine speed

$$L_{cmd} = f_{Lcmd}(T_{cmd}, N)$$

where:

- $L_{cmd}$=$L$ is commanded engine load, dimensionless.
- $T_{cmd}$ is commanded torque, in N·m.
- $N$ is engine speed, in rpm.




**`Torque command breakpoints, f_lcmd_tq_bpt` — Breakpoints**
`[15 26.43 37.86 49.29 60.71 72.14 83.57 95 106.4 117.9 129.3 140.7 152.1 163.6 175]` (default) | `vector`

Torque command breakpoints, in N·m.

**`Speed breakpoints, f_lcmd_n_bpt` — Breakpoints**
`[750 1054 1357 1661 1964 2268 2571 2875 3179 3482 3786 4089 4393 4696 5000]` (default) | `vector`

Speed breakpoints, in rpm.

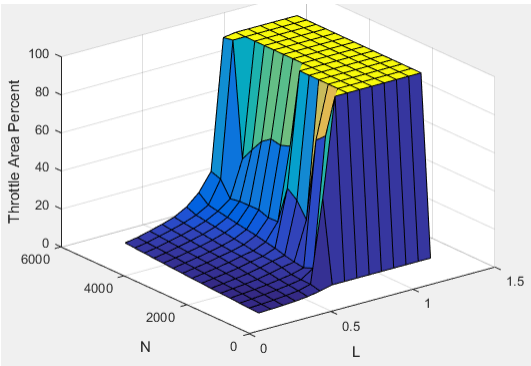
**`Throttle area percent, f_tap` — Lookup table, %**
`array`

The throttle area percent command lookup table, $f_{TAPcmd}$, is a function of commanded load and engine speed

$$TAP_{cmd} = f_{TAPcmd}(L_{cmd}, N)$$

where:

- $TAP_{cmd}$ is throttle area percentage command, in percent.
- $L_{cmd}$=$L$ is commanded engine load, dimensionless.
- $N$ is engine speed, in rpm.



**Throttle area percent load breakpoints, f_tap_ld_bpt — Breakpoints**
`[0.2 0.275 0.35 0.425 0.5 0.575 0.65 0.725 0.8 0.875 0.95 1.025 1.1 1.175 1.25]` (default) | `vector`

Throttle area percent load breakpoints, dimensionless.

**Throttle area percent speed breakpoints, f_tap_n_bpt — Breakpoints**
`[750 1054 1357 1661 1964 2268 2571 2875 3179 3482 3786 4089 4393 4696 5000]` (default) | `vector`

Throttle area percent speed breakpoints, in rpm.

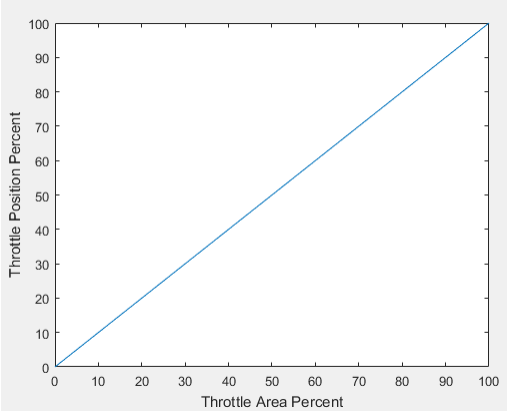
**Throttle area percent to position percent table, f_tpp — Lookup table**
`[0 100]` (default) | `vector`

The throttle position percent command lookup table, $f_{TPPcmd}$, is a function of the throttle area percentage command

$$TPP_{cmd} = f_{TPPcmd}(TAP_{cmd})$$

where:

- $TPP_{cmd}$ is throttle position percentage command, in percent.
- $TAP_{cmd}$ is throttle area percentage command, in percent.




**Throttle area percent to position percent area breakpoints, f_tpp_tap_bpt — Breakpoints**
`[0 100]` (default) | `vector`

Throttle area percent to position percent area breakpoints, dimensionless.

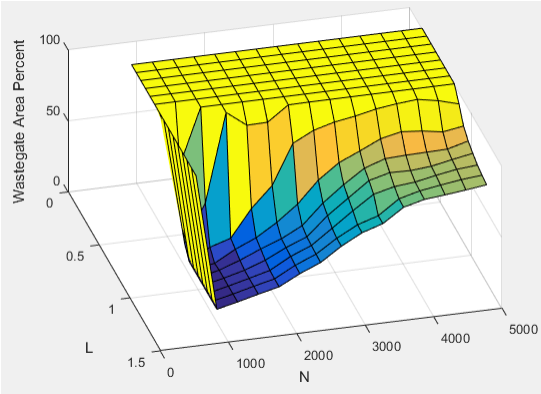
**Wastegate area percent, f_wap — Lookup table, %**
`array`

The wastegate area percent command lookup table, $f_{WAPcmd}$, is a function of the commanded engine load and engine speed

$$WAP_{cmd} = f_{WAPcmd}(L_{cmd}, N)$$

where:

- $WAP_{cmd}$ is wastegate area percentage command, in percent.
- $L_{cmd}$=$L$ is commanded engine load, dimensionless.
- $N$ is engine speed, in rpm.




**`Load breakpoints, f_wap_ld_bpt` — Breakpoints**
`[0.2 0.275 0.35 0.425 0.5 0.575 0.65 0.725 0.8 0.875 0.95 1.025 1.1 1.175 1.25]` (default) | `vector`

Load breakpoints, dimensionless.

**`Speed breakpoints, f_wap_n_bpt` — Breakpoints, rpm**
`[750 1054 1357 1661 1964 2268 2571 2875 3179 3482 3786 4089 4393 4696 5000]` (default) | `vector`

Speed breakpoints, in rpm.

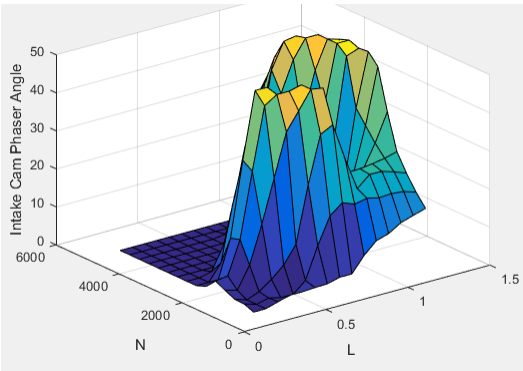
**`Intake cam phaser angle, f_icp` — Lookup table**
`array`

The intake cam phaser angle command lookup table, $f_{ICPCMD}$, is a function of the engine load and engine speed

$$\varphi_{ICPCMD} = f_{ICPCMD}(L_{est}, N)$$

where:

- $\varphi_{ICPCMD}$ is commanded intake cam phaser angle, in degrees crank advance.
- $L_{est}$=$L$ is estimated engine load, dimensionless.
- $N$ is engine speed, in rpm.



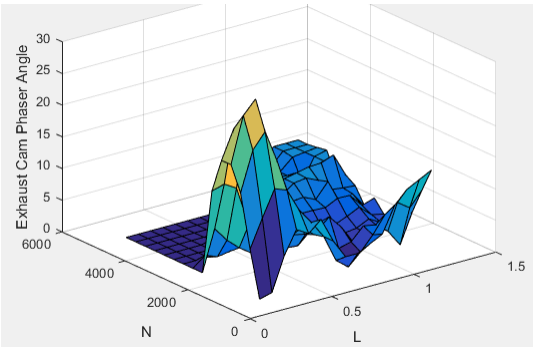

**`Exhaust cam phaser angle, f_ecp` — Lookup table**
`array`

The exhaust cam phaser angle command lookup table, $f_{ECPCMD}$, is a function of the engine load and engine speed

$$\varphi_{ECPCMD} = f_{ECPCMD}(L_{est}, N)$$

where:

- $\varphi_{ECPCMD}$ is commanded exhaust cam phaser angle, in degrees crank retard.
- $L_{est}$=$L$ is estimated engine load, dimensionless.
- $N$ is engine speed, in rpm.




**`Load breakpoints, f_cp_ld_bpt` — Breakpoints**
`[0.2 0.275 0.35 0.425 0.5 0.575 0.65 0.725 0.8 0.875 0.95 1.025 1.1 1.175 1.25]` (default) | `vector`

Load breakpoints, dimensionless.

**`Speed breakpoints, f_cp_n_bpt` — Breakpoints**
`[750 1054 1357 1661 1964 2268 2571 2875 3179 3482 3786 4089 4393 4696 5000]` (default) | `vector`

Speed breakpoints, in rpm.

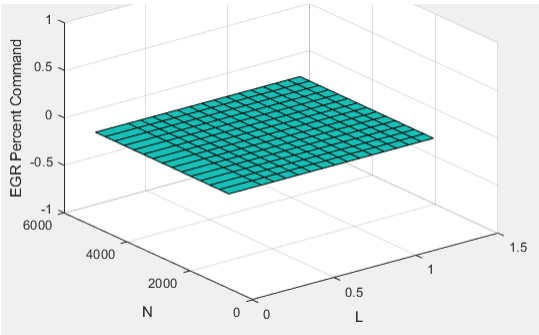
**`Commanded EGR percent, f_egrpct_cmd` — Lookup table**
`array`

The EGR percent command, $EGR_{pct,cmd}$, lookup table is a function of estimated engine load and engine speed

$$EGR_{pct,\,cmd} = f_{EGRpct,\,cmd}(L_{est}, N)$$

where:

- $EGR_{pct,cmd}$ is commanded EGR percent, dimensionless.
- $L_{est}$=$L$ is estimated engine load, dimensionless.
- $N$ is engine speed, in rpm.




**`Load breakpoints, f_egrpct_ld_bpt` — Breakpoints**
`[0 0.2 0.275 0.35 0.425 0.5 0.575 0.65 0.725 0.8 0.875 0.95 1.025 1.1 1.175 1.25]` (default) | `vector`

Engine load breakpoints, $L$, dimensionless.

**`Speed breakpoints, f_egrpct_n_bpt` — Breakpoints**
`[750 1054 1357 1661 1964 2268 2571 2875 3179 3482 3786 4089 4393 4696 5000]` (default) | `vector`

Engine speed breakpoints, $N$, in rpm.

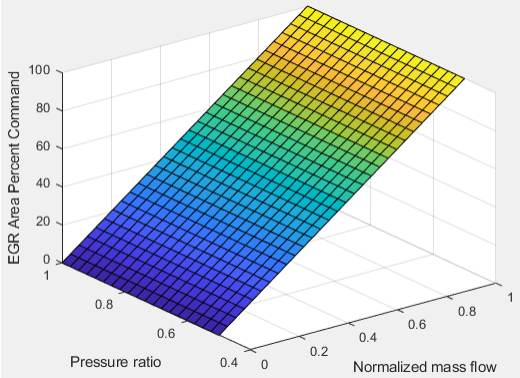
**`EGR valve area percent, f_egr_areapct_cmd` — Lookup table**
`array`

The EGR area percent command, $EGRap_{cmd}$, lookup table is a function of the normalized mass flow and pressure ratio

$$EGRap_{cmd} = f_{EGRap,\,cmd}\left(\frac{\dot{m}_{EGRstd,\,cmd}}{\dot{m}_{EGRstd,\,max}}, \frac{P_{out,\,EGR}}{P_{in,\,EGR}}\right)$$

where:

- $EGRap_{cmd}$ is commanded EGR area percent, dimensionless.
- $\frac{\dot{m}_{EGRstd,\,cmd}}{\dot{m}_{EGRstd,\,max}}$ is the normalized mass flow, dimensionless.
- $\frac{P_{out,\,EGR}}{P_{in,\,EGR}}$ is the pressure ratio, dimensionless.




**Open EGR valve standard flow, f_egr_max_stdflow — Breakpoints**
`[74.87 74.87 74.74 74.39 73.81 72.98 71.91 70.58 68.97 67.06 64.84 62.25 59.27 55.81 51.79 47.07 36.33 24.22 12.11 0]` (default) | `vector`

Maximum standard EGR valve mass flow breakpoints, $\dot{m}_{EGRstd,\,max}$, in N·m.

**Normalized EGR valve standard flow breakpoints, f_egr_areapct_nrmlzdflow_bpt — Breakpoints**
`[0 0.03448 0.06897 0.1034 0.1379 0.1724 0.2069 0.2414 0.2759 0.3103 0.3448 0.3793 0.4138 0.4483 0.4828 0.5172 0.5517 0.5862 0.6207 0.6552 0.6897 0.7241 0.7586 0.7931 0.8276 0.8621 0.8966 0.931 0.9655 1]` (default) | `vector`

Normalized mass flow breakpoints, $\frac{\dot{m}_{EGRstd,\,cmd}}{\dot{m}_{EGRstd,\,max}}$, dimensionless.

**EGR valve pressure ratio breakpoints, f_egr_areapct_pr_bpt — Breakpoints**
`vector`

Pressure ratio breakpoints, $\frac{P_{out,\,EGR}}{P_{in,\,EGR}}$, dimensionless.

#### Fuel

**Injector slope, Sinj — Slope**
`6.452` (default) | `scalar`

Fuel injector slope, $S_{inj}$, in mg/ms.

**Stoichiometric air-fuel ratio, afr_stoich — Ratio**
`14.6` (default) | `scalar`

Stoichiometric air-fuel ratio, $AFR_{stoich}$.

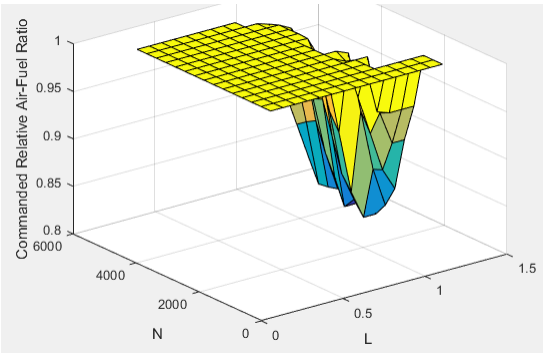
**Relative air-fuel ratio lambda, f_lamcmd — Air-fuel-ratio (AFR) lookup table**
`array`

The commanded lambda, $\lambda_{cmd}$, lookup table is a function of estimated engine load and measured engine speed

$$\lambda_{cmd} = f_{\lambda cmd}(L_{est}, N)$$

where:

- $\lambda_{cmd}$ is commanded relative AFR, dimensionless.
- $L_{est}$=$L$ is estimated engine load, dimensionless.
- $N$ is engine speed, in rpm.




**Dependencies**

To create this parameter, on the **Fuel** tab, on the **Open-loop fuel** pane, clear **Input lambda**.

**`Load breakpoints, f_lamcmd_ld_bpt` — Breakpoints**
`[0.2 0.275 0.35 0.425 0.5 0.575 0.65 0.725 0.8 0.875 0.95 1.025 1.1 1.175 1.25]` (default) | `vector`

Load breakpoints, dimensionless.

**Dependencies**

To create this parameter, on the **Fuel** tab, on the **Open-loop fuel** pane, clear **Input lambda**.

**`Speed breakpoints, f_lamcmd_n_bpt` — Breakpoints**
`[750 1054 1357 1661 1964 2268 2571 2875 3179 3482 3786 4089 4393 4696 5000]` (default) | `vector`

Speed breakpoints, in rpm.

**Dependencies**

To create this parameter, on the **Fuel** tab, on the **Open-loop fuel** pane, clear **Input lambda**.

**`Engine startup lambda enrichment delta vs coolant temperature, f_startup_lambda_delta` — Lookup table**
`[0.5 0.3 0.2 0]` (default) | `vector`

Engine startup lambda enrichment delta as a function of coolant temperature, dimensionless.

The SI Controller block uses this parameter to account for the extra fuel delivered to the spark-ignition (SI) engine during startup. If the engine speed is greater than the **Engine cranking speed** parameter, the SI Controller block enriches the optimal relative air-fuel ratio (lambda) with an exponentially decaying delta lambda. To initialize the delta lambda, the block uses the **Engine startup lambda enrichment delta vs coolant temperature** parameter to create a lambda enrichment table that is a function of the engine coolant temperature. The delta lambda exponentially

decays to zero based on a time constant specified with the **Engine startup lambda enrichment delta time constant vs coolant temperature** parameter.

**Dependencies**

To create this parameter, on the **Fuel** tab, on the **Open-loop fuel** pane, clear **Input lambda**.

**`Engine startup lambda enrichment delta time constant vs coolant temperature, f_startup_lambda_delta_timecnst` — Lambda time constant**
`[90 40 12 0]` (default) | `vector`

Engine startup lambda enrichment delta time constant versus coolant temperature, in s.

The SI Controller block uses this parameter to account for the extra fuel delivered to the spark-ignition (SI) engine during startup. If the engine speed is greater than the **Engine cranking speed** parameter, the SI Controller block enriches the optimal relative air-fuel ratio (lambda) with an exponentially decaying delta lambda. To initialize the delta lambda, the block uses the **Engine startup lambda enrichment delta vs coolant temperature** parameter to create a lambda enrichment table that is a function of the engine coolant temperature. The delta lambda exponentially decays to zero based on a time constant specified with the **Engine startup lambda enrichment delta time constant vs coolant temperature** parameter.

**Dependencies**

To create this parameter, on the **Fuel** tab, on the **Open-loop fuel** pane, clear **Input lambda**.

**`Engine startup coolant temperature breakpoints, f_startup_ect_bpt` — Breakpoints**
`[-40 0 20 50]` (default) | `vector`

Engine startup coolant temperature breakpoints, in C.

The SI Controller block uses this parameter to account for the extra fuel delivered to the spark-ignition (SI) engine during startup. If the engine speed is greater than the **Engine cranking speed** parameter, the SI Controller block enriches the optimal relative air-fuel ratio (lambda) with an exponentially decaying delta lambda. To initialize the delta lambda, the block uses the **Engine startup lambda enrichment delta vs coolant temperature** parameter to create a lambda enrichment table that is a function of the engine coolant temperature. The delta lambda exponentially decays to zero based on a time constant specified with the **Engine startup lambda enrichment delta time constant vs coolant temperature** parameter.

**Dependencies**

To create this parameter, on the **Fuel** tab, on the **Open-loop fuel** pane, clear **Input lambda**.

**`Closed-loop feedback` — Minimize commanded AFR error**
`off` (default) | `on`

Select option to minimize the commanded air-fuel-ratio (lambda), $\lambda_{cmd}$ error.

**Dependencies**

Selecting this parameter enables these parameters:

- **Closed-loop fuel proportional gain, ClsdLpFuelPGain**
- **Closed-loop fuel integral gain, ClsdLpFuelIGain**

- **Closed-loop fuel integrator limit, ClsdLpFuelIntgLmt**
- **Lambda dither amplitude, LambdaDitherAmp**
- **Lambda dither frequency, LambdaDitherFrq**
- **Oxygen sensor stoichiometric reset voltage, O2ResetStoichVoltSen**
- **Oxygen sensor minimum voltage reset, O2ResetMinVoltSen**
- **Oxygen sensor maximum voltage reset, O2ResetMaxVoltSen**
- **Oxygen sensor voltage learn update period, O2LearnUpdatePerSen**
- **Oxygen sensor voltage amplitude minimum, O2AmpMinVoltSen**
- **Oxygen sensor ready voltage, O2ReadyVoltSen**
- **Oxygen sensor not ready voltage, O2NotReadyVoltSen**

### `Dither` — Model catalytic conversion efficiency

`off` (default) | `on`

Configure the block to model dither. For open-loop analysis, select this option to tune for maximum catalytic conversion efficiency.

#### Dependencies

By default, selecting **Closed-loop feedback** configures the block to model dither.

To enable this parameter for open-loop air-fuel-ratio (lambda) commands, clear **Closed-loop feedback**.

Selecting this parameter enables these parameters:

- **Lambda dither amplitude, LambdaDitherAmp**
- **Lambda dither frequency, LambdaDitherFrq**

### `Closed-loop fuel proportional gain, ClsdLpFuelPGain` — Proportional gain

`0.005` (default) | `scalar`

Closed-loop fuel proportional gain, dimensionless.

#### Dependencies

To enable this parameter, on the **Fuel** tab, on the **Closed-loop fuel** pane, select **Closed-loop feedback**.

### `Closed-loop fuel integral gain, ClsdLpFuelIGain` — Integral gain

`0.05` (default) | `scalar`

Closed-loop fuel integral gain, dimensionless.

#### Dependencies

To enable this parameter, on the **Fuel** tab, on the **Closed-loop fuel** pane, select **Closed-loop feedback**.

### `Closed-loop fuel integrator limit, ClsdLpFuelIntgLmt` — Integrator limit

`0.2` (default) | `scalar`

Closed-loop fuel integrator limit, dimensionless.

**Dependencies**

To enable this parameter, on the **Fuel** tab, on the **Closed-loop fuel** pane, select **Closed-loop feedback**.

**`Lambda dither amplitude, LambdaDitherAmp` — Amplitude**
`0.03` (default) | `scalar`

Lambda dither amplitude, dimensionless.

**Dependencies**

To enable this parameter, on the **Fuel** tab, on the **Closed-loop fuel** pane, select either **Closed-loop feedback** or **Dither**.

**`Lambda dither frequency, LambdaDitherFrq` — Frequency**
`0.75` (default) | `scalar`

Lambda dither frequency, in Hz.

**Dependencies**

To enable this parameter, on the **Fuel** tab, on the **Closed-loop fuel** pane, select either **Closed-loop feedback** or **Dither**.

**`Oxygen sensor stoichiometric reset voltage, O2ResetStoichVoltSen` — Closed-loop AFR control**
`2500` (default) | `scalar`

Oxygen sensor stoichiometric reset voltage, O2ResetStoichVoltSen, in mV.

**Dependencies**

To enable this parameter, on the **Fuel** tab, on the **Closed-loop fuel** pane, select **Closed-loop feedback**.

**`Oxygen sensor minimum voltage reset, O2ResetMinVoltSen` — Closed-loop AFR control**
`0` (default) | `scalar`

Oxygen sensor minimum voltage reset, O2ResetMinVoltSen, in mV.

**Dependencies**

To enable this parameter, on the **Fuel** tab, on the **Closed-loop fuel** pane, select **Closed-loop feedback**.

**`Oxygen sensor maximum voltage reset, O2ResetMaxVoltSen` — Closed-loop AFR control**
`5000` (default) | `scalar`

Oxygen sensor maximum voltage reset, O2ResetMaxVoltSen, in mV.

**Dependencies**

To enable this parameter, on the **Fuel** tab, on the **Closed-loop fuel** pane, select **Closed-loop feedback**.

**`Oxygen sensor voltage learn update period, O2LearnUpdatePerSen` — Closed-loop AFR control**
`4` (default) | `scalar`

Oxygen sensor voltage learn update period, O2LearnUpdatePerSen, in mV.

**Dependencies**

To enable this parameter, on the **Fuel** tab, on the **Closed-loop fuel** pane, select **Closed-loop feedback**.

**`Oxygen sensor voltage amplitude minimum, O2AmpMinVoltSen` — Closed-loop AFR control**
`250` (default) | `scalar`

Oxygen sensor voltage amplitude minimum, O2AmpMinVoltSen, in mV.

**Dependencies**

To enable this parameter, on the **Fuel** tab, on the **Closed-loop fuel** pane, select **Closed-loop feedback**.

**`Oxygen sensor ready voltage, O2ReadyVoltSen` — Closed-loop AFR control**
`1150` (default) | `scalar`

Oxygen sensor ready voltage, O2ReadyVoltSen, in mV.

**Dependencies**

To enable this parameter, on the **Fuel** tab, on the **Closed-loop fuel** pane, select **Closed-loop feedback**.

**`Oxygen sensor not ready voltage, O2NotReadyVoltSen` — Closed-loop AFR control**
`1950` (default) | `scalar`

Oxygen sensor not ready voltage, O2NotReadyVoltSen, in mV.

**Dependencies**

To enable this parameter, on the **Fuel** tab, on the **Closed-loop fuel** pane, select **Closed-loop feedback**.

**Spark**

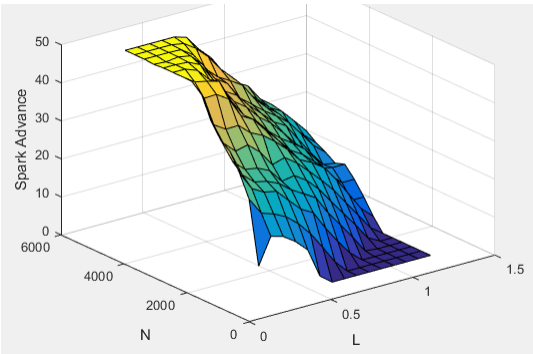
**`Spark advance table, f_sa` — Lookup table**
`array`

The spark advance lookup table is a function of estimated load and engine speed.

$$SA = f_{SA}(L_{est}, N)$$

where:

- $SA$ is spark advance, in crank advance degrees.
- $L_{est}$=$L$ is estimated engine load, dimensionless.
- $N$ is engine speed, in rpm.




**`Load breakpoints, f_sa_ld_bpt` — Breakpoints**
`[0.2 0.275 0.35 0.425 0.5 0.575 0.65 0.725 0.8 0.875 0.95 1.025 1.1 1.175 1.25]` (default) | `vector`

Load breakpoints, dimensionless.

**`Speed breakpoints, f_sa_n_bpt` — Breakpoints**
`[750 1054 1357 1661 1964 2268 2571 2875 3179 3482 3786 4089 4393 4696 5000]` (default) | `vector`

Speed breakpoints, in rpm.

#### Idle Speed

**`Target idle speed, N_idle` — Speed**
`750` (default) | `scalar`

Target idle speed, $N_{idle}$, in rpm.

**`Enable torque command limit, Trq_idlecmd_enable` — Torque**
`1` (default) | `scalar`

Torque to enable the idle speed controller, $Trq_{idlecmd,enable}$, in N·m.

**`Maximum torque command, Trq_idlecmd_max` — Torque**
`50` (default) | `scalar`

Maximum idle controller commanded torque, $Trq_{idlecmd,max}$, in N·m.

**`Proportional gain, Kp_idle` — PI Controller**
`0.05` (default) | `scalar`

Proportional gain for idle speed control, $K_{p,idle}$, in N·m/rpm.

**`Integral gain, Ki_idle` — PI Controller**
`0.2` (default) | `scalar`

Integral gain for idle speed control, $K_{i,idle}$, in N·m/(rpm·s).

**`Rev-limiter speed threshold` — Engine speed limit**
`scalar`

Engine speed limit, $N_{lim}$, in rpm.

If the engine speed, $N$, exceeds the engine speed limit, $N_{lim}$, the block sets the commanded engine torque to 0.

To smoothly transition the torque command to 0 as the engine speed approaches the speed limit, the block implements a lookup table multiplier. The lookup table multiplies the torque command by a value that ranges from 0 (engine speed exceeds limit) to 1 (engine speed does not exceed the limit).

**`Engine cranking speed, CrankSpeed` — Engine speed**
`150` (default) | `scalar`

Engine cranking speed, in rpm.

**Estimation**

**Air**

**`Number of cylinders, NCyl` — Engine cylinders**
`4` (default) | `scalar`

Number of engine cylinders, $N_{cyl}$.

**`Crank revolutions per power stroke, Cps` — Revolutions per stroke**
`2` (default) | `scalar`

Crankshaft revolutions per power stroke, *Cps*, in rev/stroke.

**`Total displaced volume, Vd` — Volume**
`0.0015` (default) | `scalar`

Displaced volume, $V_d$, in m^3.

**`Ideal gas constant air, Rair` — Constant**
`287` (default) | `scalar`

Ideal gas constant, $R_{air}$, in J/(kg·K).

**`Air standard pressure, Pstd` — Pressure**
`101325` (default) | `scalar`

Standard air pressure, $P_{std}$, in Pa.

**`Air standard temperature, Tstd` — Temperature**
`293.15` (default) | `scalar`

Standard air temperature, $T_{std}$, in K.

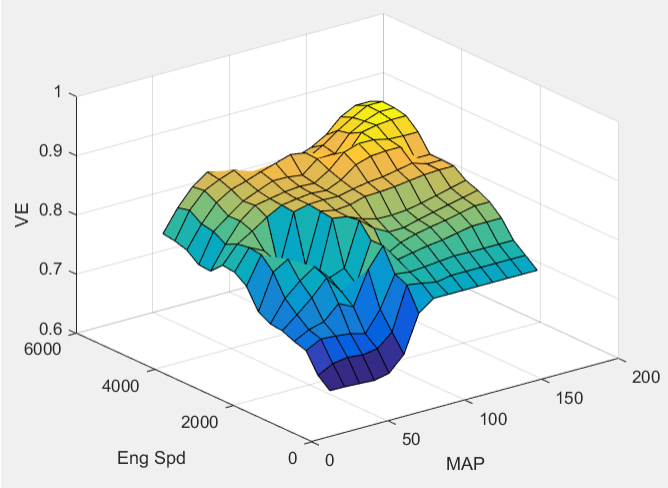
**`Speed-density volumetric efficiency, f_nv` — Lookup table**
`array`

The engine volumetric efficiency lookup table, $f_{\eta_v}$, is a function of intake manifold absolute pressure and engine speed

$$\eta_v = f_{\eta_v}(MAP, N)$$

where:

- $\eta_v$ is engine volumetric efficiency, dimensionless.
- $MAP$ is intake manifold absolute pressure, in KPa.
- $N$ is engine speed, in rpm.




**Dependencies**

To enable this parameter, for the **Air mass flow estimation model** parameter, select `Simple Speed-Density`.

**`Speed-density intake manifold pressure breakpoints, f_nv_prs_bpt` — Breakpoints**
`[31 40.64 50.29 59.93 69.57 79.21 88.86 98.5 108.1 117.8 127.4 137.1 146.7 156.4 166]` (default) | `vector`

Intake manifold pressure breakpoints for speed-density volumetric efficiency lookup table, in KPa.

**Dependencies**

To enable this parameter, for the **Air mass flow estimation model** parameter, select `Simple Speed-Density`.

**`Speed-density engine speed breakpoints, f_nv_n_bpt` — Breakpoints**
`[750 1054 1357 1661 1964 2268 2571 2875 3179 3482 3786 4089 4393 4696 5000]` (default) | `vector`

Engine speed breakpoints for speed-density volumetric efficiency lookup table, in rpm.

**Dependencies**

To enable this parameter, for the **Air mass flow estimation model** parameter, select `Simple Speed-Density`.

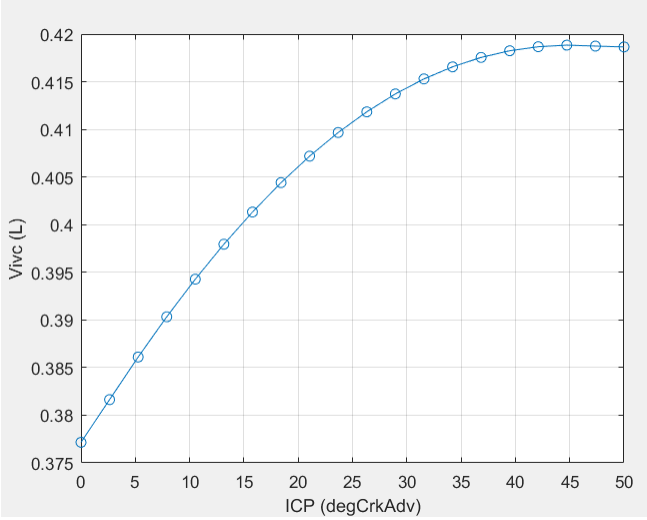
**`Cylinder volume at intake valve close table, f_vivc` — 2-D lookup table**
`array`

The cylinder volume at intake valve close table (IVC), $f_{Vivc}$ is a function of the intake cam phaser angle

$$V_{IVC} = f_{Vivc}(\varphi_{ICP})$$

where:

- $V_{IVC}$ is cylinder volume at IVC, in L.
- $\varphi_{ICP}$ is intake cam phaser angle, in crank advance degrees.




**Dependencies**

To enable this parameter, for the **Air mass flow estimation model** parameter, select `Dual Variable Cam Phasing`.

**`Engine speed breakpoints, f_tm_corr_n_bpt` — Breakpoints**
`[750 973.7 1197 1421 1645 1868 2092 2316 2539 2763 2987 3211 3434 3658 3882 4105 4329 4553 4776 5000]` (default) | `vector`

Engine speed breakpoints, in rpm.

**Dependencies**

To enable this parameter, for the **Air mass flow estimation model** parameter, select `Dual Variable Cam Phasing`.

**`Cylinder volume intake cam phase breakpoints, f_vivc_icp_bpt` — Breakpoints**
`[0 2.632 5.263 7.895 10.53 13.16 15.79 18.42 21.05 23.68 26.32 28.95 31.58 34.21 36.84 39.47 42.11 44.74 47.37 50]` (default) | `vector`

Cylinder volume at intake valve close table breakpoints.

**Dependencies**

To enable this parameter, for the **Air mass flow estimation model** parameter, select `Dual Variable Cam Phasing`.

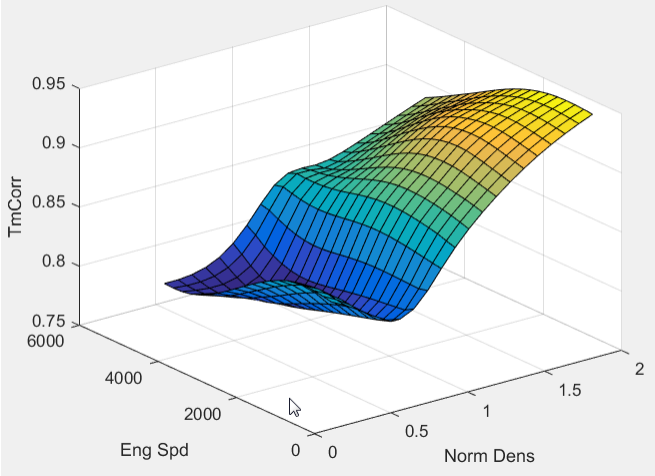
**`Cylinder trapped mass correction factor, f_tm_corr` — Lookup table**
`array`

The trapped mass correction factor table, $f_{TMcorr}$, is a function of the normalized density and engine speed

$$TM_{corr} = f_{TMcorr}(\rho_{norm}, \quad N)$$

where:

- $TM_{corr}$, is trapped mass correction multiplier, dimensionless.
- $\rho_{norm}$ is normalized density, dimensionless.
- $N$ is engine speed, in rpm.




**Dependencies**

To enable this parameter, for the **Air mass flow estimation model** parameter, select `Dual Variable Cam Phasing`.

**`Normalized density breakpoints, f_tm_corr_nd_bpt` — Breakpoints**
`[0.3 0.3895 0.4789 0.5684 0.6579 0.7474 0.8368 0.9263 1.016 1.105 1.195 1.284 1.374 1.463 1.553 1.642 1.732 1.821 1.911 2]` (default) | `vector`

Normalized density breakpoints.

**Dependencies**

To enable this parameter, for the **Air mass flow estimation model** parameter, select `Dual Variable Cam Phasing`.

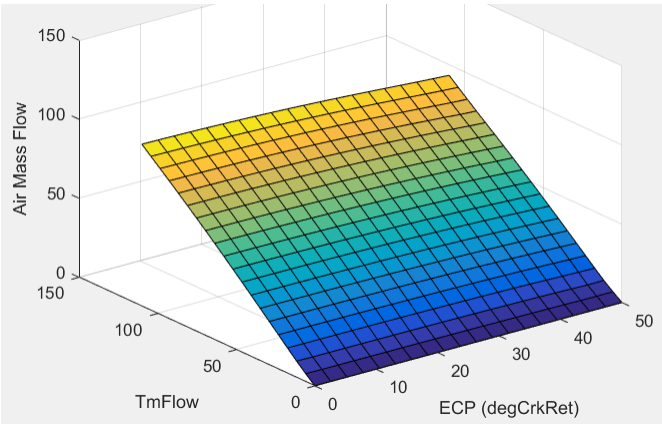
**`Intake mass flow, f_mdot_intk` — Lookup table**
`array`

The phaser intake mass flow model lookup table is a function of exhaust cam phaser angles and trapped air mass flow

$$\dot{m}_{intkideal} = f_{intkideal}(\varphi_{ECP}, TM_{flow})$$

where:

- $\dot{m}_{intkideal}$ is engine intake port mass flow at arbitrary cam phaser angles, in g/s.
- $\varphi_{ECP}$ is exhaust cam phaser angle, in degrees crank retard.
- $TM_{flow}$ is flow rate equivalent to corrected trapped mass at the current engine speed, in g/s.




**Dependencies**

To enable this parameter, for the **Air mass flow estimation model** parameter, select `Dual Variable Cam Phasing`.

**`Exhaust cam phase breakpoints, f_mdot_air_ecp_bpt` — Breakpoints**
`[0 2.632 5.263 7.895 10.53 13.16 15.79 18.42 21.05 23.68 26.32 28.95 31.58 34.21 36.84 39.47 42.11 44.74 47.37 50]` (default) | `vector`

Exhaust cam phaser breakpoints for air mass flow lookup table.

**Dependencies**

To enable this parameter, for the **Air mass flow estimation model** parameter, select `Dual Variable Cam Phasing`.

**`Trapped mass flow breakpoints, f_mdot_trpd_bpt` — Breakpoints**
`[0 5.79 11.58 17.37 23.16 28.95 34.74 40.53 46.32 52.11 57.89 63.68 69.47 75.26 81.05 86.84 92.63 98.42 104.2 110]` (default) | `vector`

Trapped mass flow breakpoints for air mass flow lookup table.

**Dependencies**

To enable this parameter, for the **Air mass flow estimation model** parameter, select `Dual Variable Cam Phasing`.

**`Air mass flow correction factor, f_mdot_air_corr` — Lookup table**
`array`

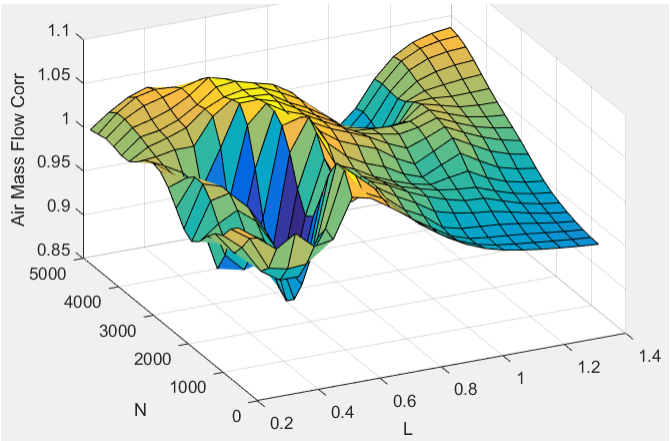
The intake air mass flow correction lookup table, $f_{aircorr}$, is a function of ideal load and engine speed

$$\dot{m}_{air} = \dot{m}_{intkideal} f_{aircorr}(L_{ideal}, N)$$

where:

- $L_{ideal}$ is engine load (normalized cylinder air mass) at arbitrary cam phaser angles, uncorrected for final steady-state cam phaser angles, dimensionless.
- $N$ is engine speed, in rpm.
- $\dot{m}_{air}$ is engine intake air mass flow final correction at steady-state cam phaser angles, in g/s.

- $\dot{m}_{intkideal}$ is engine intake port mass flow at arbitrary cam phaser angles, in g/s.




**Dependencies**

To enable this parameter, for the **Air mass flow estimation model** parameter, select `Dual Variable Cam Phasing`.

**`Engine load breakpoints for air mass flow correction, f_mdot_air_corr_ld_bpt` — Breakpoints**
`vector`

Engine load breakpoints for air mass flow final correction.

**Dependencies**

To enable this parameter, for the **Air mass flow estimation model** parameter, select `Dual Variable Cam Phasing`.

**`Engine speed breakpoints for air mass flow correction, f_mdot_air_n_bpt` — Breakpoints**
`[750 973.7 1197 1421 1645 1868 2092 2316 2539 2763 2987 3211 3434 3658 3882 4105 4329 4553 4776 5000]` (default) | `vector`

Engine speed breakpoints for air mass flow final correction.

**Dependencies**

To enable this parameter, for the **Air mass flow estimation model** parameter, select `Dual Variable Cam Phasing`.

**`EGR flow time constant, tau_egr` — Constant**
`0.2` (default) | `scalar`

EGR flow time constant, $\tau_{EGR}$, in s.

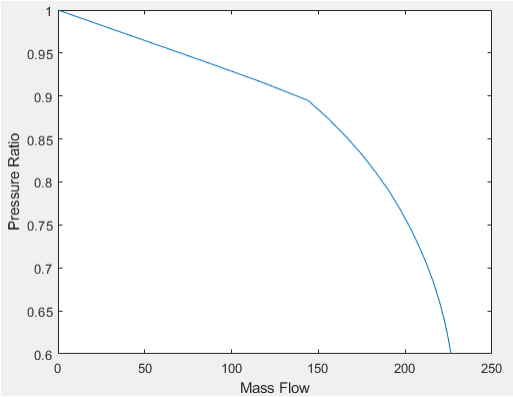
**`Intake system pressure ratio table, f_intksys_stdflow_pr` — Table**
`array`

The pressure ratio is a function of the standard mass flow

$$\frac{P_{out,EGR}}{P_{amb}} = f_{intksys,pr}(\dot{m}_{air,std})$$

where:

- $\dot{m}_{air,\,std}$ is standard mass flow, in g/s.
- $\frac{P_{out,\,EGR}}{P_{amb}}$ is pressure ratio, dimensionless.




**`Standard mass flow rate breakpoints for intake pressure ratio, f_intksys_stdflow_bpt` — Breakpoints**
`[0 29.67 59.34 89.01 117.8 144.1 155.7 166 175.2 183.3 190.7 197 202.7 207.7 212.2 216.1 219.4 222.2 224.5 226.4]` (default) | `vector`

Standard mass flow, $\dot{m}_{air,\,std}$, in g/s.

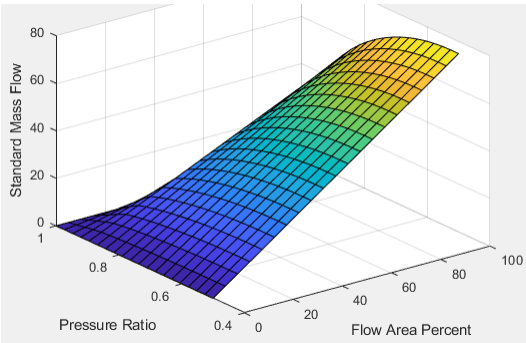
**`EGR valve standard mass flow rate, f_egr_stdflow` — Table**
`array`

The EGR valve standard mass flow lookup table is a function of EGR valve area percent and the pressure ratio

$$\dot{m}_{EGR,\,std} = f_{EGR,\,std}\left(EGRap,\,\frac{P_{out,\,EGR}}{P_{in,\,EGR}}\right)$$

where:

- $\dot{m}_{EGR,\,std}$ is EGR valve standard mass flow, dimensionless.
- $EGRap$ is EGR valve flow area percent, in percent.
- $\frac{P_{out,\,EGR}}{P_{in,\,EGR}}$ is the pressure ratio, dimensionless.




**`EGR valve standard flow pressure ratio breakpoints, f_egr_stdflow_pr_bpt` — Breakpoints**
`vector`

EGR valve standard flow pressure ratio, $\frac{P_{out,EGR}}{P_{in,EGR}}$, dimensionless.

**`EGR valve standard flow area percent breakpoints, f_egr_stdflow_egrap_bpt` — Breakpoints**
`[0;5;10;15;20;25;30;35;40;45;50;55;60;65;70;75;80;85;90;95;100]` (default) | `vector`

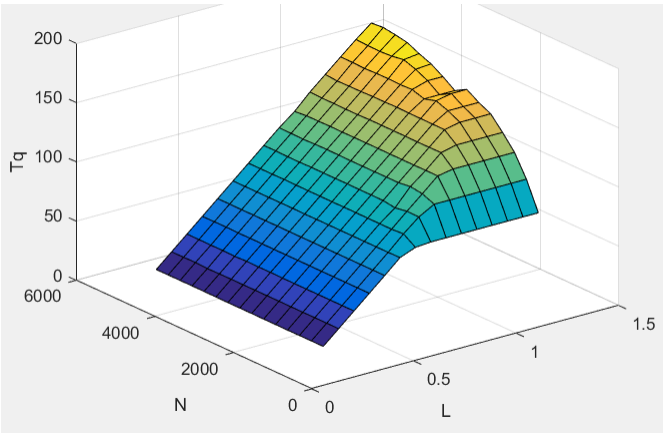
EGR valve flow area percent, *EGRap*, in percent.

**Torque**

**`Torque table, f_tq_nl` — Lookup table**
`[L x N] array`

For the simple torque lookup table model, the SI engine uses a lookup table map that is a function of engine speed and load, $T_{brake} = f_{TnL}(L, N)$, where:

- $T_{brake}$ is engine brake torque after accounting for spark advance, AFR, and friction effects, in N·m.
- $L$ is engine load, as a normalized cylinder air mass, dimensionless.
- $N$ is engine speed, in rpm.



The simple torque lookup model assumes that the calibration has negative torque values to indicate the non-firing engine load (L) versus speed (N) condition. The calibrated table (L-by-N) contains the non-firing data in the first table row (1-by-N). When the fuel delivered to the engine is zero, the model uses the data in the first table row (1-by-N) at or above 100 AFR. 100 AFR results from fuel cutoff or very lean operation where combustion cannot occur.

**Dependencies**

To enable this parameter, for the **Torque model** parameter, select `Simple Torque Lookup`.

**`Torque table load breakpoints, f_tq_nl_l_bpt` — Breakpoints**
`[0.2 0.275 0.35 0.425 0.5 0.575 0.65 0.725 0.8 0.875 0.95 1.025 1.1 1.175 1.25]` (default) | `vector` | `[1 x L] vector`

Engine load breakpoints, *L*, dimensionless.

**Dependencies**

To enable this parameter, for the **Torque model** parameter, select `Simple Torque Lookup`.

**`Torque table speed breakpoints, f_tq_nl_n_bpt` — Breakpoints**
`[750 1053.57142857143 1357.14285714286 1660.71428571429 1964.28571428571 2267.85714285714 2571.42857142857 2875 3178.57142857143 3482.14285714286 3785.71428571429 4089.28571428571 4392.85714285714 4696.42857142857 5000]` (default) | `vector` | `[1 x N] vector`

Engine speed breakpoints, *N*, in rpm.

**Dependencies**

To enable this parameter, for the **Torque model** parameter, select `Simple Torque Lookup`.

**`Crank angle pressure and torque` — Enable Crank angle signals**
`off` (default) | `on`

If you select **Crank angle pressure and torque** on the block **Torque** tab, you can:

- Simulate advanced closed-loop engine controls in desktop simulations and on HIL bench, based on cylinder pressure recorded from a model or laboratory test as a function of crank angle.
- Simulate driveline vibrations downstream of the engine due to high-frequency crankshaft torsionals.
- Simulate engine misfires due to lean operation or spark plug fouling by using the injector pulse width input.
- Simulate cylinder deactivation effect (closed intake and exhaust valves, no injected fuel) on individual cylinder pressures, mean-value airflow, mean-value torque, and crank-angle-based torque.
- Simulate the fuel-cut effect on individual cylinder pressure, mean-value torque, and crank-angle-based torque.

**Dependencies**

To enable this parameter, set **Torque model** to `Torque Structure`.

**`Cylinder pressure, f_crk_prs` — Cylinder pressure table**
`L x M x N` array

Cylinder pressure table *Prs*, as a function of speed *N*, load *L*, and crank angle *M*, in Pa.

**Dependencies**

To enable this parameter, for the **Torque model** parameter, select `Torque Structure`. Select **Crank angle pressure and torque**.

**`Brake torque, f_crk_btq` — Brake torque table**
`L x M x N` array

Brake torque table $T_{brake}$, as a function of speed *N*, load *L*, and crank angle *M*, in N·m.

**Dependencies**

To enable this parameter, for the **Torque model** parameter, select `Torque Structure`. Select **Crank angle pressure and torque**.

**`Speed breakpoints, f_crk_n_bpt` — Speed breakpoints**
`[750 5000]` (default) | `1 x N` vector

Speed breakpoints, *N*, in rpm.

**Dependencies**

To enable this parameter, for the **Torque model** parameter, select `Torque Structure`. Select **Crank angle pressure and torque**.

**`Load breakpoints, f_crk_l_bpt` — Load breakpoints**
`[0.2 1.4]` (default) | `1 x L` vector

Load breakpoints, *L*. No dimension.

**Dependencies**

To enable this parameter, for the **Torque model** parameter, select `Torque Structure`. Select **Crank angle pressure and torque**.

**`Crank angle breakpoints, f_crk_ang_bpt` — Crank angle breakpoints**
`[60 660]` (default) | `1 x M` vector

Crank angle breakpoints, *M*, in deg.

**Dependencies**

To enable this parameter, for the **Torque model** parameter, select `Torque Structure`. Select **Crank angle pressure and torque**.

**`TDC compression angles by cylinder, f_crk_tdc_ang` — TDC compression angles by cylinder**
`[0 540 180 360]` (default) | vector

Top dead center (TDC) compression angles by cylinder, in deg.

**Dependencies**

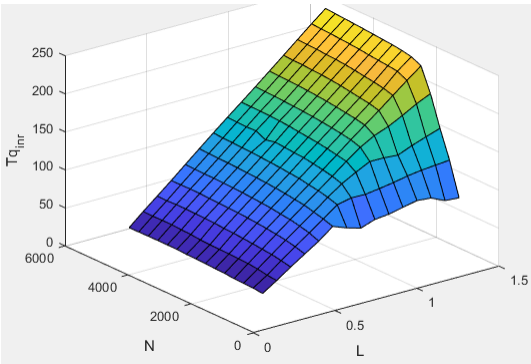
To enable this parameter, for the **Torque model** parameter, select `Torque Structure`. Select **Crank angle pressure and torque**.

### `Inner torque table, f_tq_inr` — Lookup table

`array`

The inner torque lookup table, $f_{Tqinr}$, is a function of engine speed and engine load, $Tq_{inr} = f_{Tqinr}(L, N)$, where:

- $Tq_{inr}$ is inner torque based on gross indicated mean effective pressure, in N·m.
- $L$ is engine load at arbitrary cam phaser angles, corrected for final steady-state cam phaser angles, dimensionless.
- $N$ is engine speed, in rpm.




#### Dependencies

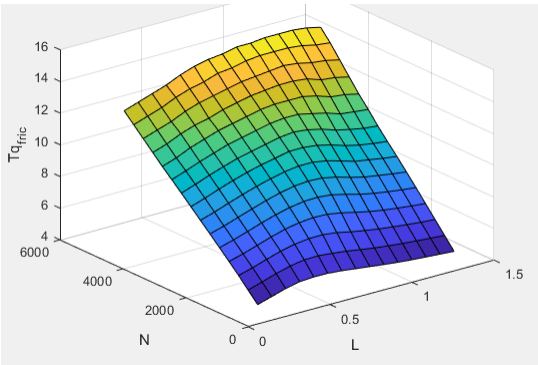
To enable this parameter, for the **Torque model** parameter, select `Torque Structure`.

### `Friction torque table, f_tq_fric` — Lookup table

`array`

The friction torque lookup table, $f_{Tfric}$, is a function of engine speed and engine load, $T_{fric} = f_{Tfric}(L, N)$, where:

- $T_{fric}$ is friction torque offset to inner torque, in N·m.
- $L$ is engine load at arbitrary cam phaser angles, corrected for final steady-state cam phaser angles, dimensionless.
- $N$ is engine speed, in rpm.



**Dependencies**

To enable this parameter, for the **Torque model** parameter, select `Torque Structure`.

**`Engine temperature modifier on friction torque, f_fric_temp_mod` — Lookup table**
`[3.96 3.22 2.56 2.26 2.11 2 1.9 1.83 1.76 1.7 1.65 1.6 1.55 1.49 1.44 1.41 1.38 1.35 1.32 1.3 1.27 1.25 1.24 1.21 1.2 1.18 1.16 1.15 1.13 1.12 1.11 1.1 1.09 1.08 1.07 1.06 1.05 1.05 1.04 1.03 1.02 1.02 1.01 1.01 1 1 1 0.999 0.997 0.995 0.993 0.991 0.989 0.987]` (default) | `vector` | `vector`

Engine temperature modifier on friction torque, $f_{fric,temp}$, dimensionless.

**Dependencies**

To enable this parameter, for the **Torque model** parameter, select `Torque Structure`.

**`Engine temperature modifier breakpoints, f_fric_temp_bpt` — Breakpoints**
`[274 276 278 280 282 284 286 288 290 292 294 296 298 300 302 304 306 308 310 312 314 316 318 320 322 324 326 328 330 332 334 336 338 340 342 344 346 348 350 352 354 356 358 360 362 364 366 368 370 372 374 376 378 380]` (default) | `vector` | `vector`

Engine temperature modifier breakpoints, in K.

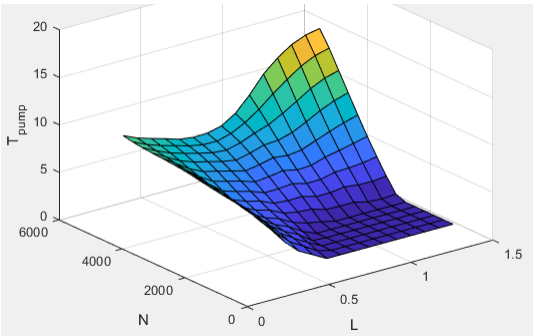
**Dependencies**

To enable this parameter, for the **Torque model** parameter, select `Torque Structure`.

**`Pumping work table, f_tq_pump` — Lookup table**
`array`

The pumping work lookup table, $f_{Tpump}$, is a function of engine load and engine speed, $T_{pump}$=`f`$_{\texttt{Tpump}}$`(L,N)`, where:

- $T_{pump}$ is pumping work, in N·m.
- $L$ is engine load, as a normalized cylinder air mass, dimensionless.
- $N$ is engine speed, in rpm.



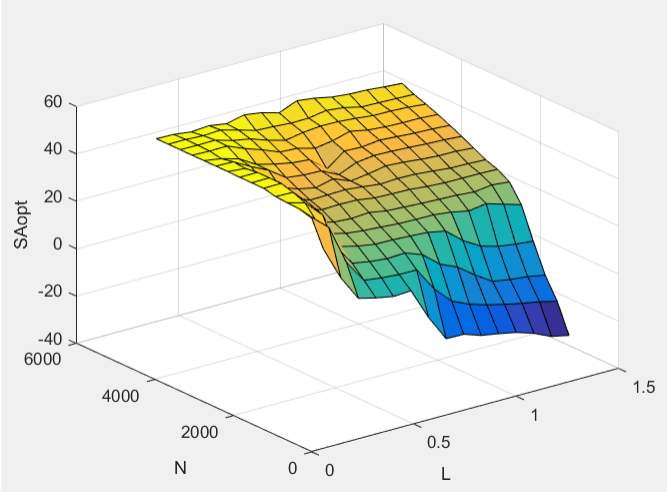

**Dependencies**

To enable this parameter, for the **Torque model** parameter, select `Torque Structure`.

**`Optimal spark table, f_sa_opt` — Lookup table**
`array`

The optimal spark lookup table, $f_{SAopt}$, is a function of engine speed and engine load, $SA_{opt} = f_{SAopt}(L, N)$, where:

- $SA_{opt}$ is optimal spark advance timing for maximum inner torque at stoichiometric air-fuel ratio (AFR), in deg.
- $L$ is engine load at arbitrary cam phaser angles, corrected for final steady-state cam phaser angles, dimensionless.
- $N$ is engine speed, in rpm.




**Dependencies**

To enable this parameter, for the **Torque model** parameter, select `Torque Structure`.

**`Inner torque load breakpoints, f_tq_inr_l_bpt` — Breakpoints**
`[0.2 0.28571 0.37143 0.45714 0.54286 0.62857 0.71429 0.8 0.88571 0.97143 1.0571 1.1429 1.2286 1.3143 1.4]` (default) | `vector`

Inner torque load breakpoints, dimensionless.

**Dependencies**

To enable this parameter, for the **Torque model** parameter, select `Torque Structure`.

**`Inner torque speed breakpoints, f_tq_inr_n_bpt` — Breakpoints**
`[750 1053.5714 1357.1429 1660.7143 1964.2857 2267.8571 2571.4286 2875 3178.5714 3482.1429 3785.7143 4089.2857 4392.8571 4696.4286 5000]` (default) | `vector`

Inner torque speed breakpoints, in rpm.

**Dependencies**

To enable this parameter, for the **Torque model** parameter, select `Torque Structure`.

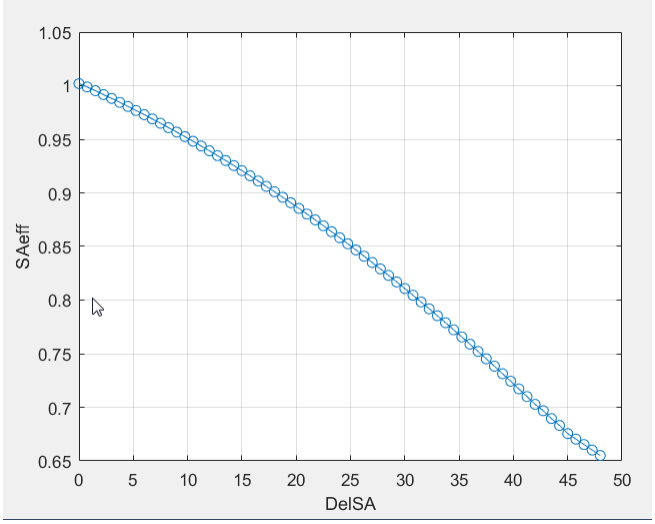
**`Spark efficiency table, f_m_sa` — Lookup table**
`array`

The spark efficiency lookup table, $f_{Msa}$, is a function of the spark retard from optimal

$$M_{sa} = f_{Msa}(\Delta SA)$$
$$\Delta SA = SA_{opt} - SA$$

where:

- $M_{sa}$ is the spark retard efficiency multiplier, dimensionless.
- $\Delta SA$is the spark retard timing distance from optimal spark advance, in deg.




**Dependencies**

To enable this parameter, for the **Torque model** parameter, select `Torque Structure`.

**`Spark retard from optimal, f_del_sa_bpt` — Breakpoints**
`[0 0.75 1.5 2.25 3 3.75 4.5 5.25 6 6.75 7.5 8.25 9 9.75 10.5 11.25 12 12.75 13.5 14.25 15 15.75 16.5 17.25 18 18.75 19.5 20.25 21 21.75 22.5 23.25 24 24.75 25.5 26.25 27 27.75 28.5 29.25 30 30.75 31.5 32.25 33 33.75 34.5 35.25 36 36.75 37.5 38.25 39 39.75 40.5 41.25 42 42.75 43.5 44.25 45 45.75 46.5 47.25 48]` (default) | `vector`

Spark retard from optimal inner torque timing breakpoints, in deg.

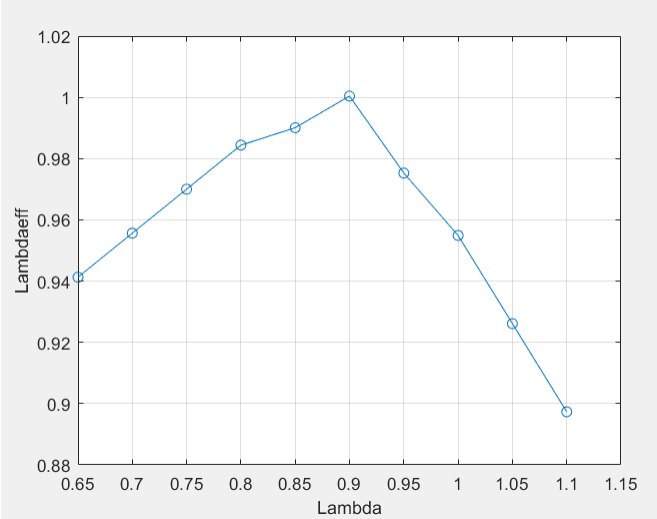
**Dependencies**

To enable this parameter, for the **Torque model** parameter, select `Torque Structure`.

**`Lambda efficiency, f_m_lam` — Lookup table**
`array`

The lambda efficiency lookup table, $f_{M\lambda}$, is a function of lambda, $M_\lambda = f_{M\lambda}(\lambda)$, where:

- $M_\lambda$ is the lambda multiplier on inner torque to account for the air-fuel ratio (AFR) effect, dimensionless.
- $\lambda$ is lambda, AFR normalized to stoichiometric fuel AFR, dimensionless.




**Dependencies**

To enable this parameter, for the **Torque model** parameter, select `Torque Structure`.

**`Lambda breakpoints, f_m_lam_bpt` — Breakpoints**
`[0.65 0.7 0.75 0.8 0.85 0.9 0.95 1 1.05 1.1]` (default) | `vector`

Lambda effect on inner torque lambda breakpoints, dimensionless.

**Dependencies**

To enable this parameter, for the **Torque model** parameter, select `Torque Structure`.

**Exhaust**

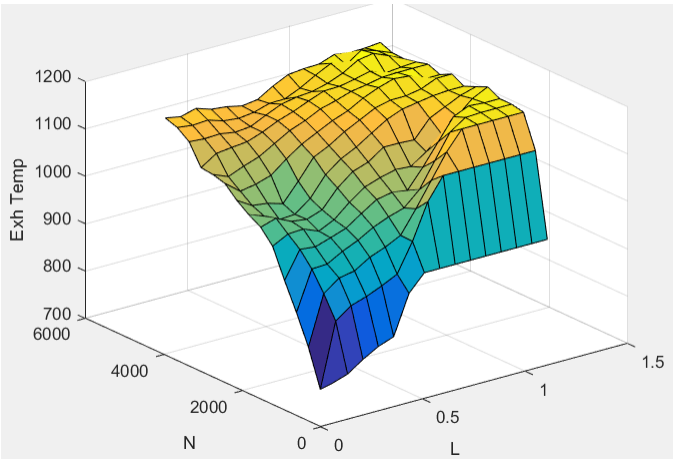
**`Exhaust temperature table, f_t_exh` — Lookup table**
`array`

The exhaust temperature lookup table, $f_{Texh}$, is a function of engine load and engine speed

$$T_{exh} = f_{Texh}(L, N)$$

where:

- $T_{exh}$ is engine exhaust temperature, in K.
- $L$ is normalized cylinder air mass or engine load, dimensionless.
- $N$ is engine speed, in rpm.




**`Load breakpoints, f_t_exh_l_bpt` — Breakpoints**
`[0.2 0.275 0.35 0.425 0.5 0.575 0.65 0.725 0.8 0.875 0.95 1.025 1.1 1.175 1.25]` (default) | `vector`

Engine load breakpoints used for exhaust temperature lookup table.

**`Speed breakpoints, f_t_exh_n_bpt` — Breakpoints**
`[750 1054 1357 1661 1964 2268 2571 2875 3179 3482 3786 4089 4393 4696 5000]` (default) | `vector`

Engine speed breakpoints used for exhaust temperature lookup table, in rpm.

## References

[1] Gerhardt, J., Hönninger, H., and Bischof, H., *A New Approach to Functional and Software Structure for Engine Management Systems — BOSCH ME7*. SAE Technical Paper 980801, 1998.

[2] Heywood, John B. *Internal Combustion Engine Fundamentals*. New York: McGraw-Hill, 1988.

[3] Leone, T. Christenson, E., Stein, R., *Comparison of Variable Camshaft Timing Strategies at Part Load*. SAE Technical Paper 960584, 1996, doi:10.4271/960584.

[4] Liu, F. and Pfeiffer, J., *Estimation Algorithms for Low Pressure Cooled EGR in Spark-Ignition Engines*. SAE Int. J. Engines 8(4):2015, doi:10.4271/2015-01-1620.

## Extended Capabilities

### C/C++ Code Generation
Generate C and C++ code using Simulink® Coder™.

## See Also
Mapped SI Engine | SI Core Engine

### Topics
“Engine Calibration Maps”

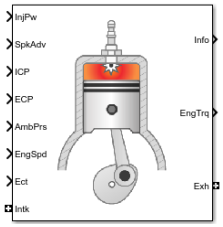
**External Websites**
Developing a Period-Based Air-Fuel Ratio Controller Using a Low-Cost Switching Sensor

**Introduced in R2017a**

# SI Core Engine

Spark-ignition engine from intake to exhaust port

**Library:** Powertrain Blockset / Propulsion / Combustion Engine Components / Core Engine




## Description

The SI Core Engine block implements a spark-ignition (SI) engine from intake to exhaust port. You can use the block in larger vehicle models, hardware-in-the-loop (HIL) engine control design, or vehicle-level fuel economy and performance simulations.

The SI Core Engine block calculates:

- Brake torque
- Fuel flow
- Port gas mass flow, including exhaust gas recirculation (EGR)
- Air-fuel ratio (AFR)
- Exhaust temperature and exhaust mass flow rate
- Engine-out (EO) exhaust emissions
  - Hydrocarbon (HC)
  - Carbon monoxide (CO)
  - Nitric oxide and nitrogen dioxide (NOx)
  - Carbon dioxide ($CO_2$)
  - Particulate matter (PM)

### Air Mass Flow

To calculate engine air mass flow, configure the SI engine to use either of these air mass flow models.

| Air Mass Flow Model | Description |
|---|---|
| “SI Engine Speed-Density Air Mass Flow Model” | Uses the speed-density equation to calculate the engine air mass flow, relating the engine air mass flow to the intake manifold pressure and engine speed. Consider using this air mass flow model in engines with fixed valvetrain designs. |

| Air Mass Flow Model | Description |
|---|---|
| “SI Engine Dual-Independent Cam Phaser Air Mass Flow Model” | To calculate the engine air mass flow, the dual-independent cam phaser model uses:<br>• Empirical calibration parameters developed from engine mapping measurements<br>• Desktop calibration parameters derived from engine computer-aided design (CAD) data<br>In contrast to typical embedded air mass flow calculations based on direct air mass flow measurement with an air mass flow (MAF) sensor, this air mass flow model offers:<br>• Elimination of MAF sensors in dual cam-phased valvetrain applications<br>• Reasonable accuracy with changes in altitude<br>• Semiphysical modeling approach<br>• Bounded behavior<br>• Suitable execution time for electronic control unit (ECU) implementation<br>• Systematic development of a relatively small number of calibration parameters |

### Brake Torque

To calculate the brake torque, configure the SI engine to use either of these torque models.

| Brake Torque Model | Description |
|---|---|
| “SI Engine Torque Structure Model” | For the structured brake torque calculation, the SI engine uses tables for the inner torque, friction torque, optimal spark, spark efficiency, and lambda efficiency.<br><br>If you select **Crank angle pressure and torque** on the block **Torque** tab, you can:<br><br>• Simulate advanced closed-loop engine controls in desktop simulations and on HIL bench, based on cylinder pressure recorded from a model or laboratory test as a function of crank angle.<br>• Simulate driveline vibrations downstream of the engine due to high-frequency crankshaft torsionals.<br>• Simulate engine misfires due to lean operation or spark plug fouling by using the injector pulse width input.<br>• Simulate cylinder deactivation effect (closed intake and exhaust valves, no injected fuel) on individual cylinder pressures, mean-value airflow, mean-value torque, and crank-angle-based torque.<br>• Simulate the fuel-cut effect on individual cylinder pressure, mean-value torque, and crank-angle-based torque. |
| “SI Engine Simple Torque Model” | For the simple brake torque calculation, the SI engine block uses a torque lookup table map that is a function of engine speed and load. |

### Fuel Flow

To calculate the fuel flow, the SI Core Engine block uses fuel injector characteristics and fuel injector pulse-width.

$$\dot{m}_{fuel} = \frac{NS_{inj}Pw_{inj}N_{cyl}}{Cps\left(\frac{60s}{\min}\right)\left(\frac{1000mg}{g}\right)}$$

To calculate the fuel economy for high-fidelity models, the block uses the volumetric fuel flow.

$$Q_{fuel} = \frac{\dot{m}_{fuel}}{\left(\frac{1000kg}{m^3}\right)Sg_{fuel}}$$

The equation uses these variables.

| | |
|---|---|
| $\dot{m}_{fuel}$ | Fuel mass flow, g/s |
| $\omega$ | Engine rotational speed, rad/s |
| $Cps$ | Crankshaft revolutions per power stroke, rev/stroke |
| $S_{inj}$ | Fuel injector slope, mg/ms |
| $Pw_{inj}$ | Fuel injector pulse-width, ms |

| | |
|---|---|
| $N_{cyl}$ | Number of engine cylinders |
| $N$ | Engine speed, rpm |
| $Sg_{fuel}$ | Specific gravity of fuel |
| $Q_{fuel}$ | Volumetric fuel flow |

The block uses the internal signal `FlwDir` to track the direction of the flow.

### Air-Fuel Ratio

To calculate the air-fuel (AFR) ratio, the CI Core Engine and SI Core Engine blocks implement this equation.

$$AFR = \frac{\dot{m}_{air}}{\dot{m}_{fuel}}$$

The CI Core Engine uses this equation to calculate the relative AFR.

$$\lambda = \frac{AFR}{AFR_S}$$

To calculate the exhaust gas recirculation (EGR), the blocks implement this equation. The calculation expresses the EGR as a percent of the total intake port flow.

$$EGR_{pct} = 100\frac{\dot{m}_{intk,b}}{\dot{m}_{intk}} = 100y_{intk,b}$$

The equations use these variables.

| | |
|---|---|
| $AFR$ | Air-fuel ratio |
| $AFR_s$ | Stoichiometric air-fuel ratio |
| $\dot{m}_{intk}$ | Engine air mass flow |
| $\dot{m}_{fuel}$ | Fuel mass flow |
| $\lambda$ | Relative AFR |
| $y_{intk,b}$ | Intake burned mass fraction |
| $EGR_{pct}$ | EGR percent |
| $\dot{m}_{intk,b}$ | Recirculated burned gas mass flow rate |

### Exhaust

The block calculates the:

- Exhaust gas temperature
- Exhaust gas-specific enthalpy
- Exhaust gas mass flow rate
- Engine-out (EO) exhaust emissions:
  - Hydrocarbon (HC)
  - Carbon monoxide (CO)

- Nitric oxide and nitrogen dioxide (NOx)
- Carbon dioxide ($CO_2$)
- Particulate matter (PM)

The exhaust temperature determines the specific enthalpy.

$$h_{exh} = Cp_{exh}T_{exh}$$

The exhaust mass flow rate is the sum of the intake port air mass flow and the fuel mass flow.

$$\dot{m}_{exh} = \dot{m}_{intake} + \dot{m}_{fuel}$$

To calculate the exhaust emissions, the block multiplies the emission mass fraction by the exhaust mass flow rate. To determine the emission mass fractions, the block uses lookup tables that are functions of the engine torque and speed.

$$y_{exh,i} = f_{i_frac}(T_{brake}, N)$$
$$\dot{m}_{exh,i} = \dot{m}_{exh}y_{exh,i}$$

The fraction of air and fuel entering the intake port, injected fuel, and stoichiometric AFR determine the air mass fraction that exits the exhaust.

$$y_{exh,air} = \max\left[y_{in,air} - \frac{\dot{m}_{fuel} + y_{in,fuel}\dot{m}_{intake}}{\dot{m}_{fuel} + \dot{m}_{intake}}AFR_s\right]$$

If the engine is operating at the stoichiometric or fuel rich AFR, no air exits the exhaust. Unburned hydrocarbons and burned gas comprise the remainder of the exhaust gas. This equation determines the exhaust burned gas mass fraction.

$$y_{exh,b} = \max[(1 - y_{exh,air} - y_{exh,HC}), 0]$$

The equations use these variables.

| | |
|---|---|
| $T_{exh}$ | Engine exhaust temperature |
| $h_{exh}$ | Exhaust manifold inlet-specific enthalpy |
| $Cp_{exh}$ | Exhaust gas specific heat |
| $\dot{m}_{intk}$ | Intake port air mass flow rate |
| $\dot{m}_{fuel}$ | Fuel mass flow rate |
| $\dot{m}_{exh}$ | Exhaust mass flow rate |
| $y_{in,fuel}$ | Intake fuel mass fraction |
| $y_{exh,i}$ | Exhaust mass fraction for i = $CO_2$, CO, HC, NOx, air, burned gas, and PM |
| $\dot{m}_{exh,i}$ | Exhaust mass flow rate for i = $CO_2$, CO, HC, NOx, air, burned gas, and PM |
| $T_{brake}$ | Engine brake torque |
| $N$ | Engine speed |
| $y_{exh,air}$ | Exhaust air mass fraction |
| $y_{exh,b}$ | Exhaust air burned mass fraction |

### Power Accounting

For the power accounting, the block implements equations that depend on **Torque model**.

When you set **Torque model** to `Simple Torque Lookup`, the block implements these equations.

| Bus Signal | | | Description | Equations |
|---|---|---|---|---|
| `PwrInfo` | `PwrTrnsfrd` — Power transferred between blocks<br>• Positive signals indicate flow into block<br>• Negative signals indicate flow out of block | `PwrIntkHeatFlw` | Intake heat flow | $\dot{m}_{intk}h_{intk}$ |
| | | `PwrExhHeatFlw` | Exhaust heat flow | $-\dot{m}_{exh}h_{exh}$ |
| | | `PwrCrkshft` | Crankshaft power | $-T_{brake}\omega$ |
| | `PwrNotTrnsfrd` — Power crossing the block boundary, but not transferred<br>• Positive signals indicate an input<br>• Negative signals indicate a loss | `PwrFuel` | Fuel input power | $\dot{m}_{fuel}LHV$ |
| | | `PwrLoss` | All losses | $T_{brake}\omega - \dot{m}_{fuel}LHV - \dot{m}_{intk}h_{intk} + \dot{m}_{exh}h_{exh}$ |

| Bus Signal | | | Description | Equations |
|---|---|---|---|---|
| | `PwrStored` — Stored energy rate of change<br>• Positive signals indicate an increase<br>• Negative signals indicate a decrease | *Not used* | | |

When you set **Torque model** to `Torque Structure`, the block implements these equations.

| Bus Signal | | | Description | Equations |
|---|---|---|---|---|
| `PwrInfo` | `PwrTrnsfrd` — Power transferred between blocks<br>• Positive signals indicate flow into block<br>• Negative signals indicate flow out of block | `PwrIntkHeatFlw` | Intake heat flow | $\dot{m}_{intk}h_{intk}$ |
| | | `PwrExhHeatFlw` | Exhaust heat flow | $-\dot{m}_{exh}h_{exh}$ |
| | | `PwrCrkshft` | Crankshaft power | $-T_{brake}\omega$ |
| | `PwrNotTrnsfrd` — Power crossing the block boundary, but not transferred<br>• Positive signals indicate an input<br>• Negative signals indicate a loss | `PwrFuel` | Fuel input power | $\dot{m}_{fuel}LHV$ |
| | | `PwrFricLoss` | Friction loss | $-T_{fric}\omega$ |
| | | `PwrPumpLoss` | Pumping loss | $-T_{pump}\omega$ |
| | | `PwrHeatTrnsfrLoss` | Heat transfer loss | $T_{brake}\omega - \dot{m}_{fuel}LHV - \dot{m}_{intk}h_{intk} + \dot{m}_{exh}h_{exh} + T_{fric}\omega + T_{pump}\omega$ |

| Bus Signal | | | Description | Equations |
|---|---|---|---|---|
| | `PwrStored` — Stored energy rate of change<br>• Positive signals indicate an increase<br>• Negative signals indicate a decrease | *Not used* | | |

| | |
|---|---|
| $h_{exh}$ | Exhaust manifold inlet-specific enthalpy |
| $h_{intk}$ | Intake port specific enthalpy |
| $\dot{m}_{intk}$ | Intake port air mass flow rate |
| $\dot{m}_{fuel}$ | Fuel mass flow rate |
| $\dot{m}_{exh}$ | Exhaust mass flow rate |
| $\omega$ | Engine speed |
| $T_{brake}$ | Brake torque |
| $T_{pump}$ | Engine pumping work offset to inner torque |
| $T_{fric}$ | Engine friction torque |
| $LHV$ | Fuel lower heating value |

## Ports

### Input

#### `InjPw` — Fuel injector pulse-width

`scalar`

Fuel injector pulse-width, $Pw_{inj}$, in ms.

#### `SpkAdv` — Spark advance

`scalar`

Spark advance, $SA$, in degrees crank angle before top dead center (degBTDC).

##### Dependencies

To create this port, for the **Torque model** parameter, select `Torque Structure`.

#### `ICP` — Intake cam phase angle command

`scalar`

Intake cam phase angle command, $\varphi_{ICPCMD}$, in degCrkAdv, or degrees crank advance.

#### Dependencies

To create this port, for the **Air mass flow model** parameter, select `Dual-Independent Variable Cam Phasing`.

### `ECP` — Exhaust cam phase angle command

`scalar`

Exhaust cam phase angle command, $\varphi_{ECPCMD}$, in degCrkRet, or degrees crank retard.

#### Dependencies

To create this port, for the **Air mass flow model** parameter, select `Dual-Independent Variable Cam Phasing`.

### `AmbPrs` — Ambient pressure

`scalar`

Ambient pressure, $P_{Amb}$, in Pa.

#### Dependencies

To create this port, for the **Air mass flow model** parameter, select `Dual-Independent Variable Cam Phasing`.

### `EngSpd` — Engine speed

`scalar`

Engine speed, $N$, in rpm.

### `Ect` — Engine cooling temperature

`scalar`

Engine cooling temperature, $T_{coolant}$, in K.

#### Dependencies

To enable this parameter, for **Torque model**, select `Torque Structure`.

### `Intk` — Intake port pressure, temperature, enthalpy, mass fractions

two-way connector port

Bus containing the upstream:

- `Prs` — Pressure, in Pa
- `Temp` — Temperature, in K
- `Enth` — Specific enthalpy, in J/kg
- `MassFrac` — Intake port mass fractions, dimensionless. EGR mass flow at the intake port is burned gas.

  Specifically, a bus with these mass fractions:

  - `O2MassFrac` — Oxygen
  - `N2MassFrac` — Nitrogen

- `UnbrndFuelMassFrac` — Unburned fuel
- `CO2MassFrac` — Carbon dioxide
- `H2OMassFrac` — Water
- `COMassFrac` — Carbon monoxide
- `NOMassFrac` — Nitric oxide
- `NO2MassFrac` — Nitrogen dioxide
- `NOxMassFrac` — Nitric oxide and nitrogen dioxide
- `PmMassFrac` — Particulate matter
- `AirMassFrac` — Air
- `BrndGasMassFrac` — Burned gas

**Exh — Exhaust port pressure, temperature, enthalpy, mass fractions**
two-way connector port

Bus containing the exhaust:

- `Prs` — Pressure, in Pa
- `Temp` — Temperature, in K
- `Enth` — Specific enthalpy, in J/kg
- `MassFrac` — Exhaust port mass fractions, dimensionless.

  Specifically, a bus with these mass fractions:

  - `O2MassFrac` — Oxygen
  - `N2MassFrac` — Nitrogen
  - `UnbrndFuelMassFrac` — Unburned fuel
  - `CO2MassFrac` — Carbon dioxide
  - `H2OMassFrac` — Water
  - `COMassFrac` — Carbon monoxide
  - `NOMassFrac` — Nitric oxide
  - `NO2MassFrac` — Nitrogen dioxide
  - `NOxMassFrac` — Nitric oxide and nitrogen dioxide
  - `PmMassFrac` — Particulate matter
  - `AirMassFrac` — Air
  - `BrndGasMassFrac` — Burned gas

**Output**

**`Info` — Bus signal**
bus

Bus signal that contains these block calculations.

| Signal | Description | Variable | Units |
|---|---|---|---|
| IntkGasMassFlw | Engine intake air mass flow | $\dot{m}_{air}$ | kg/s |
| IntkAirMassFlw | Engine intake port mass flow | $\dot{m}_{intk}$ | kg/s |
| NrmlzdAirChrg | Engine load (that is, normalized cylinder air mass) corrected for final steady-state cam phase angles | $L$ | N/A |
| Afr | Air-fuel ratio at engine exhaust port | $AFR$ | N/A |
| FuelMassFlw | Fuel flow into engine | $\dot{m}_{fuel}$ | kg/s |
| FuelVolFlw | Volumetric fuel flow | $Q_{fuel}$ | $m^3/s$ |
| ExhManGasTemp | Exhaust gas temperature at exhaust manifold inlet | $T_{exh}$ | K |
| EngTrq | Engine brake torque | $T_{brake}$ | N·m |
| EngSpd | Engine speed | $N$ | rpm |
| IntkCamPhase | Intake cam phaser angle | $\varphi_{ICP}$ i | degrees crank advance |
| ExhCamPhase | Exhaust cam phaser angle | $\varphi_{ECP}$ | degrees crank retard |
| CrkAng | Engine crankshaft absolute angle | $\int_0^{(360)Cps} EngSpd\frac{180}{30}d\theta$<br>where $Cps$ is crankshaft revolutions per power stroke | degrees crank angle |
| EgrPct | EGR percent | $EGR_{pct}$ | N/A |
| EoAir | EO air mass flow rate | $\dot{m}_{exh}$ | kg/s |
| EoBrndGas | EO burned gas mass flow rate | $y_{exh,b}$ | kg/s |
| EoHC | EO hydrocarbon emission mass flow rate | $y_{exh,HC}$ | kg/s |
| EoCO | EO carbon monoxide emission mass flow rate | $y_{exh,CO}$ | kg/s |
| EoNOx | EO nitric oxide and nitrogen dioxide emissions mass flow rate | $y_{exh,NOx}$ | kg/s |
| EoCO2 | EO carbon dioxide emission mass flow rate | $y_{exh,CO2}$ | kg/s |

| Signal | | | Description | Variable | Units |
|---|---|---|---|---|---|
| `EoPm` | | | EO particulate matter emission mass flow rate | $\dot{y}_{exh,PM}$ | kg/s |
| `CylPrs` | | | Cylinder pressure | N/A | Pa |
| `EngTrqCrk` | | | Crank-angle based engine torque | N/A | N·m |
| `PwrInfo` | `PwrTrnsfrd` | `PwrIntkHeatFlw` | Intake heat flow | $\dot{m}_{intk}h_{intk}$ | W |
| | | `PwrExhHeatFlw` | Exhaust heat flow | $-\dot{m}_{exh}h_{exh}$ | W |
| | | `PwrCrkshft` | Crankshaft power | $-T_{brake}\omega$ | W |
| | `PwrNotTrnsfrd` | `PwrFuel` | Fuel input power | $\dot{m}_{fuel}LHV$ | W |
| | | `PwrLoss` | For **Torque model** set to `Simple Torque Lookup`:<br><br>All losses | $T_{brake}\omega - \dot{m}_{fuel}LHV$<br>$- \dot{m}_{intk}h_{intk} + \dot{m}_{exh}h_{exh}$ | W |
| | | `PwrFricLoss` | For **Torque model** set to `Torque Structure`:<br><br>Friction loss | $-T_{fric}\omega$ | W |
| | | `PwrPumpLoss` | For **Torque model** set to `Torque Structure`:<br><br>Pumping loss | $-T_{pump}\omega$ | W |
| | | `PwrHeatTrnsfrLoss` | For **Torque model** set to `Torque Structure`:<br><br>Heat transfer loss | $T_{brake}\omega - \dot{m}_{fuel}LHV$<br>$- \dot{m}_{intk}h_{intk} + \dot{m}_{exh}h_{exh}$<br>$+ T_{fric}\omega + T_{pump}\omega$ | W |
| | `PwrStored` | *Not used* | | | |

**`EngTrq` — Engine brake torque**
`scalar`

Engine brake torque, $T_{brake}$, in N·m.

**`Intk` — Intake port mass flow rate, heat flow rate, temperature, mass fraction**
two-way connector port

Bus containing:

- `MassFlwRate` — Intake port mass flow rate, in kg/s
- `HeatFlwRate` — Intake port heat flow rate, in J/s
- `Temp` — Intake port temperature, in K
- `MassFrac` — Intake port mass fractions, dimensionless.

  Specifically, a bus with these mass fractions:

- `O2MassFrac` — Oxygen
- `N2MassFrac` — Nitrogen
- `UnbrndFuelMassFrac` — Unburned fuel
- `CO2MassFrac` — Carbon dioxide
- `H2OMassFrac` — Water
- `COMassFrac` — Carbon monoxide
- `NOMassFrac` — Nitric oxide
- `NO2MassFrac` — Nitrogen dioxide
- `NOxMassFrac` — Nitric oxide and nitrogen dioxide
- `PmMassFrac` — Particulate matter
- `AirMassFrac` — Air
- `BrndGasMassFrac` — Burned gas

**`Exh` — Exhaust port mass flow rate, heat flow rate, temperature, mass fraction**
two-way connector port

Bus containing:

- `MassFlwRate` — Exhaust port mass flow rate, in kg/s
- `HeatFlwRate` — Exhaust heat flow rate, in J/s
- `Temp` — Exhaust temperature, in K
- `MassFrac` — Exhaust port mass fractions, dimensionless.

  Specifically, a bus with these mass fractions:

  - `O2MassFrac` — Oxygen
  - `N2MassFrac` — Nitrogen
  - `UnbrndFuelMassFrac` — Unburned fuel
  - `CO2MassFrac` — Carbon dioxide
  - `H2OMassFrac` — Water
  - `COMassFrac` — Carbon monoxide
  - `NOMassFrac` — Nitric oxide
  - `NO2MassFrac` — Nitrogen dioxide
  - `NOxMassFrac` — Nitric oxide and nitrogen dioxide
  - `PmMassFrac` — Particulate matter
  - `AirMassFrac` — Air
  - `BrndGasMassFrac` — Burned gas

## Parameters

### Block Options

**`Air mass flow model` — Select air mass flow model**
`Dual-Independent Variable Cam Phasing` (default) | `Simple Speed-Density`

To calculate engine air mass flow, configure the SI engine to use either of these air mass flow models.

| Air Mass Flow Model | Description |
|---|---|
| “SI Engine Speed-Density Air Mass Flow Model” | Uses the speed-density equation to calculate the engine air mass flow, relating the engine air mass flow to the intake manifold pressure and engine speed. Consider using this air mass flow model in engines with fixed valvetrain designs. |
| “SI Engine Dual-Independent Cam Phaser Air Mass Flow Model” | To calculate the engine air mass flow, the dual-independent cam phaser model uses:<br>• Empirical calibration parameters developed from engine mapping measurements<br>• Desktop calibration parameters derived from engine computer-aided design (CAD) data<br>In contrast to typical embedded air mass flow calculations based on direct air mass flow measurement with an air mass flow (MAF) sensor, this air mass flow model offers:<br>• Elimination of MAF sensors in dual cam-phased valvetrain applications<br>• Reasonable accuracy with changes in altitude<br>• Semiphysical modeling approach<br>• Bounded behavior<br>• Suitable execution time for electronic control unit (ECU) implementation<br>• Systematic development of a relatively small number of calibration parameters |

#### Dependencies

The table summarizes the parameter dependencies.

| Air Mass Flow Model | Enables Parameters |
|---|---|
| `Dual-Independent Variable Cam Phasing` | **Cylinder volume at intake valve close table, f_vivc**<br>**Cylinder volume intake cam phase breakpoints, f_vivc_icp_bpt**<br>**Cylinder trapped mass correction factor, f_tm_corr**<br>**Normalized density breakpoints, f_tm_corr_nd_bpt**<br>**Engine speed breakpoints, f_tm_corr_n_bpt**<br>**Air mass flow, f_mdot_air**<br>**Exhaust cam phase breakpoints, f_mdot_air_ecp_bpt**<br>**Trapped mass flow breakpoints, f_mdot_trpd_bpt**<br>**Air mass flow correction factor, f_mdot_air_corr**<br>**Engine load breakpoints for air mass flow correction, f_mdot_air_corr_ld_bpt**<br>**Engine speed breakpoints for air mass flow correction, f_mdot_air_n_bpt** |
| `Simple Speed Density` | **Speed-density volumetric efficiency, f_nv**<br>**Speed-density intake manifold pressure breakpoints, f_nv_prs_bpt**<br>**Speed-density engine speed breakpoints, f_nv_n_bpt** |

**`Torque model` — Select torque model**
`Torque Structure` (default) | `Simple Torque Lookup`

To calculate the brake torque, configure the SI engine to use either of these torque models.

| Brake Torque Model | Description |
|---|---|
| “SI Engine Torque Structure Model” | For the structured brake torque calculation, the SI engine uses tables for the inner torque, friction torque, optimal spark, spark efficiency, and lambda efficiency.<br><br>If you select **Crank angle pressure and torque** on the block **Torque** tab, you can:<br><br>• Simulate advanced closed-loop engine controls in desktop simulations and on HIL bench, based on cylinder pressure recorded from a model or laboratory test as a function of crank angle.<br>• Simulate driveline vibrations downstream of the engine due to high-frequency crankshaft torsionals.<br>• Simulate engine misfires due to lean operation or spark plug fouling by using the injector pulse width input.<br>• Simulate cylinder deactivation effect (closed intake and exhaust valves, no injected fuel) on individual cylinder pressures, mean-value airflow, mean-value torque, and crank-angle-based torque.<br>• Simulate the fuel-cut effect on individual cylinder pressure, mean-value torque, and crank-angle-based torque. |
| “SI Engine Simple Torque Model” | For the simple brake torque calculation, the SI engine block uses a torque lookup table map that is a function of engine speed and load. |

#### Dependencies

The table summarizes the parameter dependencies.

| Torque Model | Enables Parameters |
|---|---|
| `Torque Structure` | **Inner torque table, f_tq_inr**<br>**Friction torque table, f_tq_fric**<br>**Engine temperature modifier on friction torque, f_fric_temp_mod**<br>**Engine temperature modifier breakpoints, f_fric_temp_bpt**<br>**Pumping work table, f_tq_pump**<br>**Optimal spark table, f_sa_opt**<br>**Inner torque load breakpoints, f_tq_inr_l_bpt**<br>**Inner torque speed breakpoints, f_tq_inr_n_bpt**<br>**Spark efficiency table, f_m_sa**<br>**Spark retard from optimal, f_del_sa_bpt**<br>**Lambda efficiency, f_m_lam**<br>**Lambda breakpoints, f_m_lam_bpt** |
| `Simple Torque Lookup` | **Torque table, f_tq_nl**<br>**Torque table load breakpoints, f_tq_nl_l_bpt**<br>**Torque table speed breakpoints, f_tq_nl_n_bpt** |

**Air**

**`Number of cylinders, NCyl` — Engine cylinders**
4 (default) | `scalar`

Number of engine cylinders, $N_{cyl}$.

**`Crank revolutions per power stroke, Cps` — Revolutions per stroke**
2 (default) | `scalar`

Crankshaft revolutions per power stroke, *Cps*, in rev/stroke.

**`Total displaced volume, Vd` — Volume**
`0.0015` (default) | `scalar`

Displaced volume, $V_d$, in m^3.

**`Ideal gas constant air, Rair` — Constant**
287 (default) | `scalar`

Ideal gas constant, $R_{air}$, in J/(kg·K).

**`Air standard pressure, Pstd` — Pressure**
`101325` (default) | `scalar`

Standard air pressure, $P_{std}$, in Pa.

**`Air standard temperature, Tstd` — Temperature**
`293.15` (default) | `scalar`

Standard air temperature, $T_{std}$, in K.

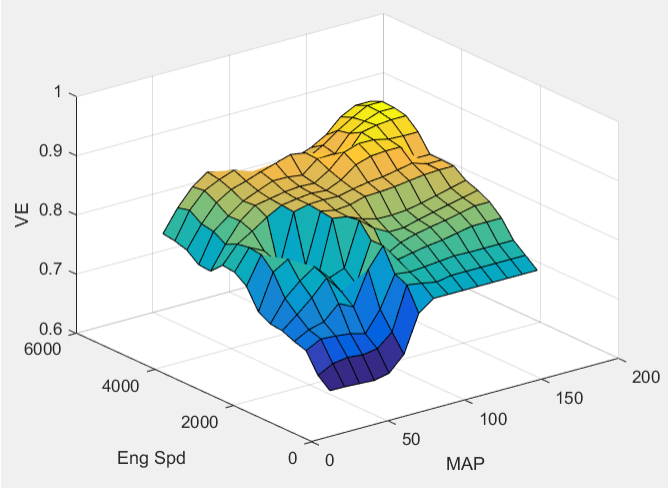
**`Speed-density volumetric efficiency, f_nv` — Lookup table**
`array`

The engine volumetric efficiency lookup table, $f_{\eta_v}$, is a function of intake manifold absolute pressure and engine speed

$$\eta_v = f_{\eta_v}(MAP, N)$$

where:

- $\eta_v$ is engine volumetric efficiency, dimensionless.
- $MAP$ is intake manifold absolute pressure, in KPa.
- $N$ is engine speed, in rpm.




**Dependencies**

To enable this parameter, for the **Air mass flow model** parameter, select `Simple Speed-Density`.

**`Speed-density intake manifold pressure breakpoints, f_nv_prs_bpt` — Breakpoints**
`[31 40.6428571428571 50.2857142857143 59.9285714285714 69.5714285714286 79.2142857142857 88.8571428571429 98.5 108.142857142857 117.785714285714 127.428571428571 137.071428571429 146.714285714286 156.357142857143 166]` (default) | `array`

Intake manifold pressure breakpoints for speed-density volumetric efficiency lookup table, in KPa.

**Dependencies**

To enable this parameter, for the **Air mass flow model** parameter, select `Simple Speed-Density`.

**Speed-density engine speed breakpoints, f_nv_n_bpt — Breakpoints**
`[750 1053.57142857143 1357.14285714286 1660.71428571429 1964.28571428571 2267.85714285714 2571.42857142857 2875 3178.57142857143 3482.14285714286 3785.71428571429 4089.28571428571 4392.85714285714 4696.42857142857 5000]` (default) | `array`

Engine speed breakpoints for speed-density volumetric efficiency lookup table, in rpm.

**Dependencies**

To enable this parameter, for the **Air mass flow model** parameter, select `Simple Speed-Density`.

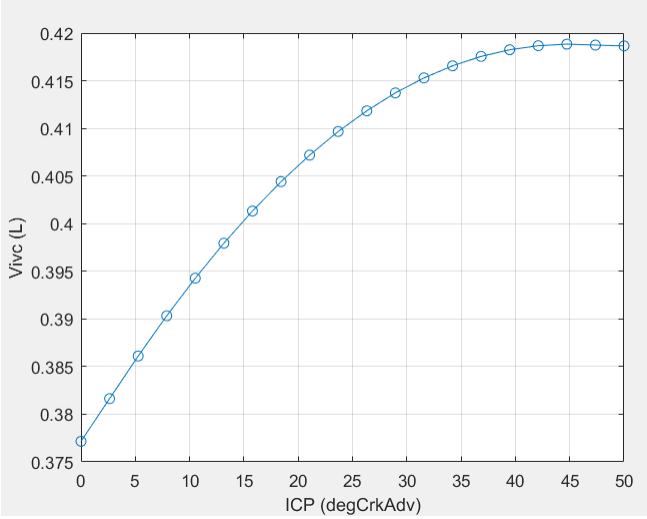
**Cylinder volume at intake valve close table, f_vivc — 2-D lookup table**
`array`

The cylinder volume at intake valve close table (IVC), $f_{Vivc}$ is a function of the intake cam phaser angle

$$V_{IVC} = f_{Vivc}(\varphi_{ICP})$$

where:

- $V_{IVC}$ is cylinder volume at IVC, in L.
- $\varphi_{ICP}$ is intake cam phaser angle, in crank advance degrees.




**Dependencies**

To enable this parameter, for the **Air mass flow model** parameter, select `Dual-Independent Variable Cam Phasing`.

**Cylinder volume intake cam phase breakpoints, f_vivc_icp_bpt — Breakpoints**
`[0 2.6316 5.2632 7.8947 10.5263 13.1579 15.7895 18.4211 21.0526 23.6842 26.3158 28.9474 31.5789 34.2105 36.8421 39.4737 42.1053 44.7368 47.3684 50]` (default) | `vector`

Cylinder volume intake cam phase breakpoints, in L.

**Dependencies**

To enable this parameter, for the **Air mass flow model** parameter, select `Dual-Independent Variable Cam Phasing`.

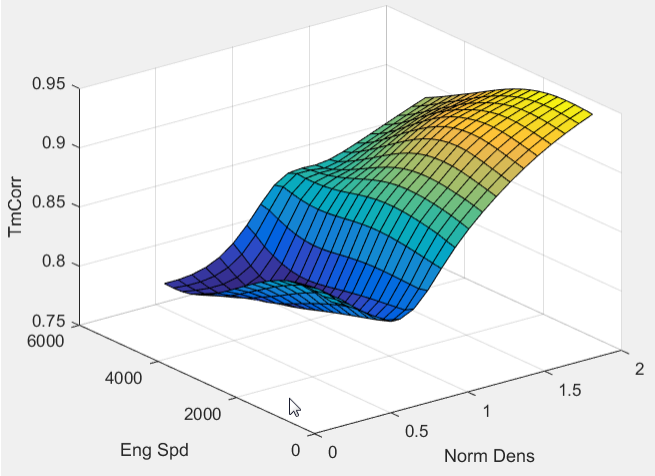
**`Cylinder trapped mass correction factor, f_tm_corr` — Lookup table**
`array`

The trapped mass correction factor table, $f_{TMcorr}$, is a function of the normalized density and engine speed

$$TM_{corr} = f_{TMcorr}(\rho_{norm}, \quad N)$$

where:

- $TM_{corr}$, is trapped mass correction multiplier, dimensionless.
- $\rho_{norm}$ is normalized density, dimensionless.
- $N$ is engine speed, in rpm.




**Dependencies**

To enable this parameter, for the **Air mass flow model** parameter, select `Dual-Independent Variable Cam Phasing`.

**`Normalized density breakpoints, f_tm_corr_nd_bpt` — Breakpoints**
`[0.3 0.38947 0.47895 0.56842 0.65789 0.74737 0.83684 0.92632 1.0158 1.1053 1.1947 1.2842 1.3737 1.4632 1.5526 1.6421 1.7316 1.8211 1.9105 2]` (default) | `vector`

Normalized density breakpoints, dimensionless.

**Dependencies**

To enable this parameter, for the **Air mass flow model** parameter, select `Dual-Independent Variable Cam Phasing`.

**`Engine speed breakpoints, f_tm_corr_n_bpt` — Breakpoints**
`[750 973.6842 1197.3684 1421.0526 1644.7368 1868.4211 2092.1053 2315.7895 2539.4737 2763.1579 2986.8421 3210.5263 3434.2105 3657.8947 3881.5789 4105.2632 4328.9474 4552.6316 4776.3158 5000]` (default) | `vector`

Engine speed breakpoints, in rpm.

**Dependencies**

To enable this parameter, for the **Air mass flow model** parameter, select `Dual-Independent Variable Cam Phasing`.

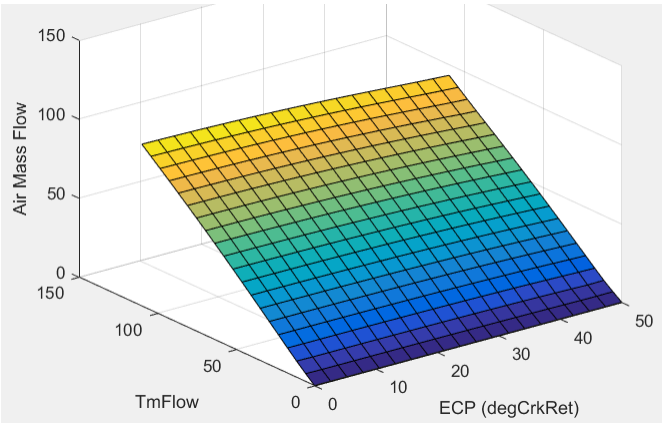
**`Intake mass flow, f_mdot_intk` — Lookup table**
`array`

The phaser intake mass flow model lookup table is a function of exhaust cam phaser angles and trapped air mass flow

$$\dot{m}_{intkideal} = f_{intkideal}(\varphi_{ECP}, TM_{flow})$$

where:

- $\dot{m}_{intkideal}$ is engine intake port mass flow at arbitrary cam phaser angles, in g/s.
- $\varphi_{ECP}$ is exhaust cam phaser angle, in degrees crank retard.
- $TM_{flow}$ is flow rate equivalent to corrected trapped mass at the current engine speed, in g/s.




**Dependencies**

To enable this parameter, for the **Air mass flow model** parameter, select `Dual-Independent Variable Cam Phasing`.

**`Exhaust cam phase breakpoints, f_mdot_air_ecp_bpt` — Breakpoints**
`[0 2.6316 5.2632 7.8947 10.5263 13.1579 15.7895 18.4211 21.0526 23.6842 26.3158 28.9474 31.5789 34.2105 36.8421 39.4737 42.1053 44.7368 47.3684 50]` (default) | `vector`

Exhaust cam phaser breakpoints for air mass flow lookup table, in degrees crank retard.

**Dependencies**

To enable this parameter, for the **Air mass flow model** parameter, select `Dual-Independent Variable Cam Phasing`.

**`Trapped mass flow breakpoints, f_mdot_trpd_bpt` — Breakpoints**
`[0 5.7895 11.5789 17.3684 23.1579 28.9474 34.7368 40.5263 46.3158 52.1053 57.8947 63.6842 69.4737 75.2632 81.0526 86.8421 92.6316 98.4211 104.2105 110]` (default) | `vector`

Trapped mass flow breakpoints for air mass flow lookup table, in g/s.

**Dependencies**

To enable this parameter, for the **Air mass flow model** parameter, select `Dual-Independent Variable Cam Phasing`.

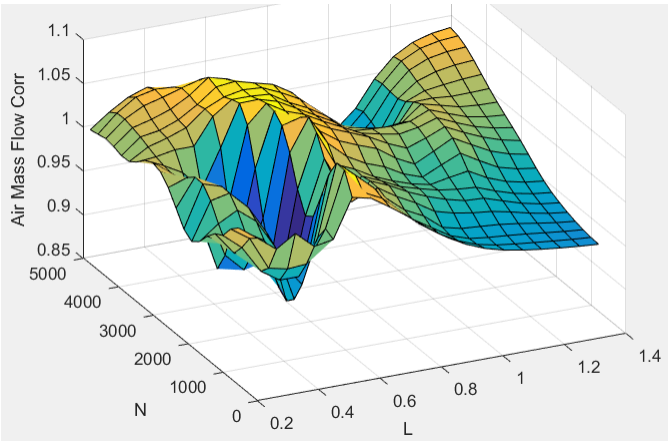
**`Air mass flow correction factor, f_mdot_air_corr` — Lookup table**
`array`

The intake air mass flow correction lookup table, $f_{aircorr}$, is a function of ideal load and engine speed

$$\dot{m}_{air} = \dot{m}_{intkideal} f_{aircorr}(L_{ideal}, N)$$

where:

- $L_{ideal}$ is engine load (normalized cylinder air mass) at arbitrary cam phaser angles, uncorrected for final steady-state cam phaser angles, dimensionless.
- $N$ is engine speed, in rpm.
- $\dot{m}_{air}$ is engine intake air mass flow final correction at steady-state cam phaser angles, in g/s.
- $\dot{m}_{intkideal}$ is engine intake port mass flow at arbitrary cam phaser angles, in g/s.




**Dependencies**

To enable this parameter, for the **Air mass flow model** parameter, select `Dual-Independent Variable Cam Phasing`.

### Engine load breakpoints for air mass flow correction, f_mdot_air_corr_ld_bpt — Breakpoints

vector

Engine load breakpoints for air mass flow final correction, dimensionless.

#### Dependencies

To enable this parameter, for the **Air mass flow model** parameter, select `Dual-Independent Variable Cam Phasing`.

### Engine speed breakpoints for air mass flow correction, f_mdot_air_n_bpt — Breakpoints

vector

Engine speed breakpoints for air mass flow final correction, in rpm.

#### Dependencies

To enable this parameter, for the **Air mass flow model** parameter, select `Dual-Independent Variable Cam Phasing`.

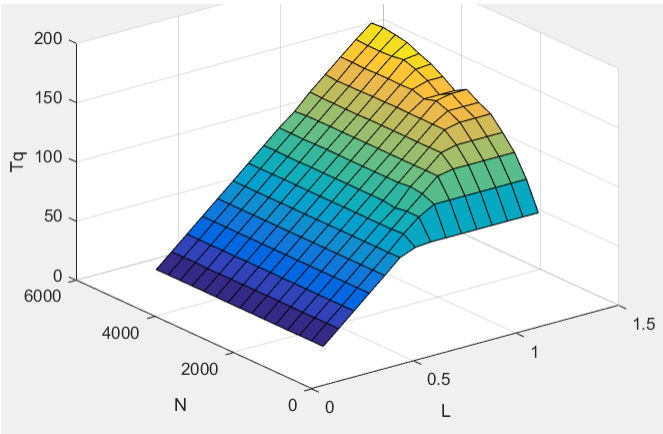
#### Torque

### Torque table, f_tq_nl — Lookup table

[L x N] array

For the simple torque lookup table model, the SI engine uses a lookup table map that is a function of engine speed and load, $T_{brake} = f_{TnL}(L, N)$, where:

- $T_{brake}$ is engine brake torque after accounting for spark advance, AFR, and friction effects, in N·m.
- $L$ is engine load, as a normalized cylinder air mass, dimensionless.
- $N$ is engine speed, in rpm.




The simple torque lookup model assumes that the calibration has negative torque values to indicate the non-firing engine load (L) versus speed (N) condition. The calibrated table (L-by-N) contains the non-firing data in the first table row (1-by-N). When the fuel delivered to the engine is zero, the model uses the data in the first table row (1-by-N) at or above 100 AFR. 100 AFR results from fuel cutoff or very lean operation where combustion cannot occur.

**Dependencies**

To enable this parameter, for the **Torque model** parameter, select `Simple Torque Lookup`.

**`Torque table load breakpoints, f_tq_nl_l_bpt` — Breakpoints**
`[0.2 0.275 0.35 0.425 0.5 0.575 0.65 0.725 0.8 0.875 0.95 1.025 1.1 1.175 1.25]` (default) | `vector` | `[1 x L] vector`

Engine load breakpoints, *L*, dimensionless.

**Dependencies**

To enable this parameter, for the **Torque model** parameter, select `Simple Torque Lookup`.

**`Torque table speed breakpoints, f_tq_nl_n_bpt` — Breakpoints**
`[750 1053.57142857143 1357.14285714286 1660.71428571429 1964.28571428571 2267.85714285714 2571.42857142857 2875 3178.57142857143 3482.14285714286 3785.71428571429 4089.28571428571 4392.85714285714 4696.42857142857 5000]` (default) | `vector` | `[1 x N] vector`

Engine speed breakpoints, *N*, in rpm.

**Dependencies**

To enable this parameter, for the **Torque model** parameter, select `Simple Torque Lookup`.

**`Crank angle pressure and torque` — Enable Crank angle signals**
`off` (default) | `on`

If you select **Crank angle pressure and torque** on the block **Torque** tab, you can:

- Simulate advanced closed-loop engine controls in desktop simulations and on HIL bench, based on cylinder pressure recorded from a model or laboratory test as a function of crank angle.
- Simulate driveline vibrations downstream of the engine due to high-frequency crankshaft torsionals.
- Simulate engine misfires due to lean operation or spark plug fouling by using the injector pulse width input.
- Simulate cylinder deactivation effect (closed intake and exhaust valves, no injected fuel) on individual cylinder pressures, mean-value airflow, mean-value torque, and crank-angle-based torque.
- Simulate the fuel-cut effect on individual cylinder pressure, mean-value torque, and crank-angle-based torque.

**Dependencies**

To enable this parameter, set **Torque model** to `Torque Structure`.

**`Cylinder pressure, f_crk_prs` — Cylinder pressure table**
`L x M x N` array

Cylinder pressure table *Prs*, as a function of speed *N*, load *L*, and crank angle *M*, in Pa.

**Dependencies**

To enable this parameter, for the **Torque model** parameter, select `Torque Structure`. Select **Crank angle pressure and torque**.

**`Brake torque, f_crk_btq` — Brake torque table**
`L x M x N` array

Brake torque table $T_{brake}$, as a function of speed $N$, load $L$, and crank angle $M$, in N·m.

**Dependencies**

To enable this parameter, for the **Torque model** parameter, select `Torque Structure`. Select **Crank angle pressure and torque**.

**`Speed breakpoints, f_crk_n_bpt` — Speed breakpoints**
`[750 5000]` (default) | `1 x N` vector

Speed breakpoints, $N$, in rpm.

**Dependencies**

To enable this parameter, for the **Torque model** parameter, select `Torque Structure`. Select **Crank angle pressure and torque**.

**`Load breakpoints, f_crk_l_bpt` — Load breakpoints**
`[0.2 1.4]` (default) | `1 x L` vector

Load breakpoints, $L$. No dimension.

**Dependencies**

To enable this parameter, for the **Torque model** parameter, select `Torque Structure`. Select **Crank angle pressure and torque**.

**`Crank angle breakpoints, f_crk_ang_bpt` — Crank angle breakpoints**
`[60 660]` (default) | `1 x M` vector

Crank angle breakpoints, $M$, in deg.

**Dependencies**

To enable this parameter, for the **Torque model** parameter, select `Torque Structure`. Select **Crank angle pressure and torque**.

**`TDC compression angles by cylinder, f_crk_tdc_ang` — TDC compression angles by cylinder**
`[0 540 180 360]` (default) | vector

Top dead center (TDC) compression angles by cylinder, in deg.

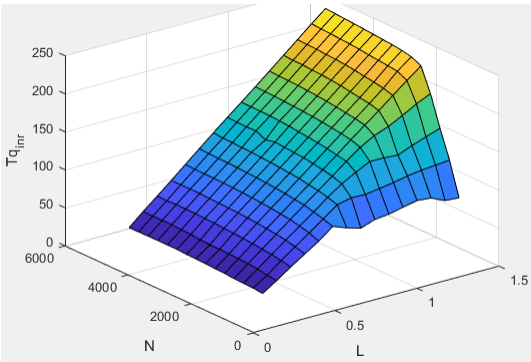
**Dependencies**

To enable this parameter, for the **Torque model** parameter, select `Torque Structure`. Select **Crank angle pressure and torque**.

**`Inner torque table, f_tq_inr` — Lookup table**
`array`

The inner torque lookup table, $f_{Tqinr}$, is a function of engine speed and engine load, $Tq_{inr} = f_{Tqinr}(L, N)$, where:

- $Tq_{inr}$ is inner torque based on gross indicated mean effective pressure, in N·m.
- $L$ is engine load at arbitrary cam phaser angles, corrected for final steady-state cam phaser angles, dimensionless.
- $N$ is engine speed, in rpm.



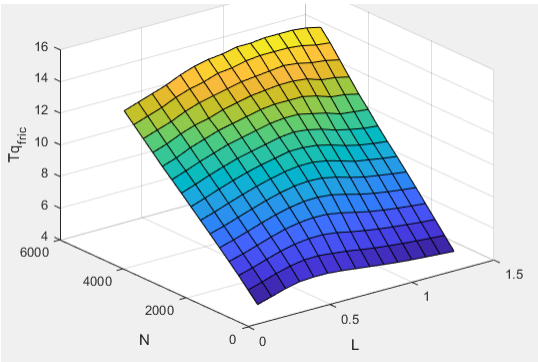

**Dependencies**

To enable this parameter, for the **Torque model** parameter, select `Torque Structure`.

**`Friction torque table, f_tq_fric` — Lookup table**
`array`

The friction torque lookup table, $f_{Tfric}$, is a function of engine speed and engine load, $T_{fric} = f_{Tfric}(L, N)$, where:

- $T_{fric}$ is friction torque offset to inner torque, in N·m.
- $L$ is engine load at arbitrary cam phaser angles, corrected for final steady-state cam phaser angles, dimensionless.
- $N$ is engine speed, in rpm.




**Dependencies**

To enable this parameter, for the **Torque model** parameter, select `Torque Structure`.

**`Engine temperature modifier on friction torque, f_fric_temp_mod` — Lookup table**
`[3.96 3.22 2.56 2.26 2.11 2 1.9 1.83 1.76 1.7 1.65 1.6 1.55 1.49 1.44 1.41 1.38 1.35 1.32 1.3 1.27 1.25 1.24 1.21 1.2 1.18 1.16 1.15 1.13 1.12 1.11 1.1`

`1.09 1.08 1.07 1.06 1.05 1.05 1.04 1.03 1.02 1.02 1.01 1.01 1 1 1 0.999 0.997 0.995 0.993 0.991 0.989 0.987]` (default) | `vector` | `vector`

Engine temperature modifier on friction torque, $f_{fric,temp}$, dimensionless.

**Dependencies**

To enable this parameter, for the **Torque model** parameter, select `Torque Structure`.

**`Engine temperature modifier breakpoints, f_fric_temp_bpt` — Breakpoints**
`[274 276 278 280 282 284 286 288 290 292 294 296 298 300 302 304 306 308 310 312 314 316 318 320 322 324 326 328 330 332 334 336 338 340 342 344 346 348 350 352 354 356 358 360 362 364 366 368 370 372 374 376 378 380]` (default) | `vector` | `vector`

Engine temperature modifier breakpoints, in K.

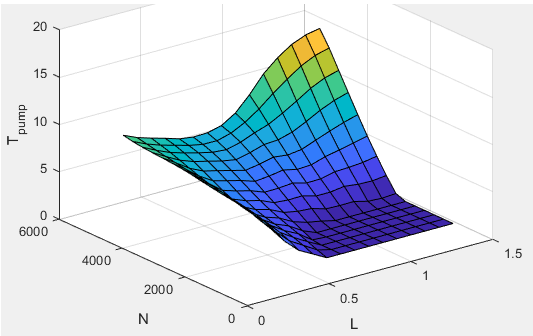
**Dependencies**

To enable this parameter, for the **Torque model** parameter, select `Torque Structure`.

**`Pumping work table, f_tq_pump` — Lookup table**
`array`

The pumping work lookup table, $f_{Tpump}$, is a function of engine load and engine speed, $T_{pump}$=`f`$_{\texttt{Tpump}}$`(L,N)`, where:

- $T_{pump}$ is pumping work, in N·m.
- $L$ is engine load, as a normalized cylinder air mass, dimensionless.
- $N$ is engine speed, in rpm.




**Dependencies**

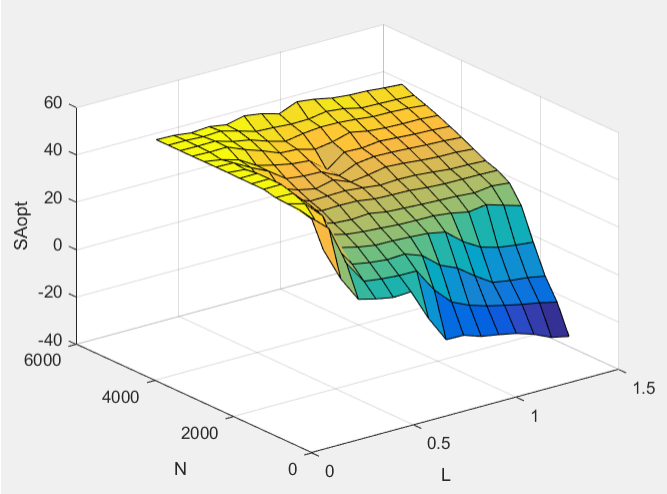
To enable this parameter, for the **Torque model** parameter, select `Torque Structure`.

**`Optimal spark table, f_sa_opt` — Lookup table**
`array`

The optimal spark lookup table, $f_{SAopt}$, is a function of engine speed and engine load, $SA_{opt} = f_{SAopt}(L, N)$, where:

- $SA_{opt}$ is optimal spark advance timing for maximum inner torque at stoichiometric air-fuel ratio (AFR), in deg.

- $L$ is engine load at arbitrary cam phaser angles, corrected for final steady-state cam phaser angles, dimensionless.
- $N$ is engine speed, in rpm.




**Dependencies**

To enable this parameter, for the **Torque model** parameter, select `Torque Structure`.

**`Inner torque load breakpoints, f_tq_inr_l_bpt` — Breakpoints**
`[0.2 0.28571 0.37143 0.45714 0.54286 0.62857 0.71429 0.8 0.88571 0.97143 1.0571 1.1429 1.2286 1.3143 1.4]` (default) | `vector`

Inner torque load breakpoints, dimensionless.

**Dependencies**

To enable this parameter, for the **Torque model** parameter, select `Torque Structure`.

**`Inner torque speed breakpoints, f_tq_inr_n_bpt` — Breakpoints**
`[750 1053.5714 1357.1429 1660.7143 1964.2857 2267.8571 2571.4286 2875 3178.5714 3482.1429 3785.7143 4089.2857 4392.8571 4696.4286 5000]` (default) | `vector`

Inner torque speed breakpoints, in rpm.

**Dependencies**

To enable this parameter, for the **Torque model** parameter, select `Torque Structure`.

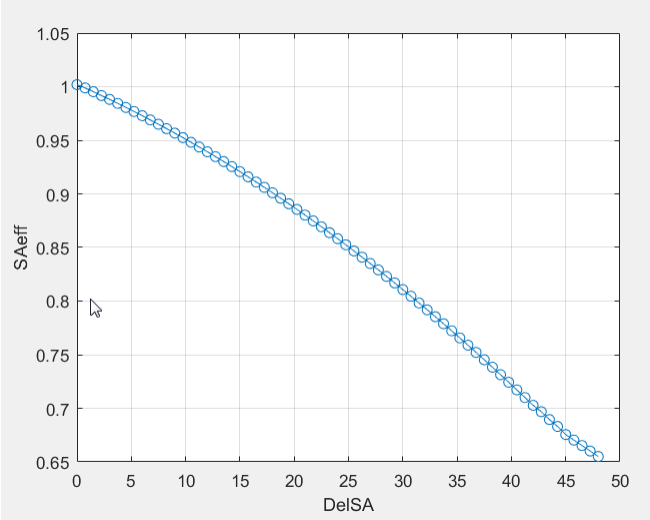
**`Spark efficiency table, f_m_sa` — Lookup table**
`array`

The spark efficiency lookup table, $f_{Msa}$, is a function of the spark retard from optimal

$$M_{sa} = f_{Msa}(\Delta SA)$$
$$\Delta SA = SA_{opt} - SA$$

where:

- $M_{sa}$ is the spark retard efficiency multiplier, dimensionless.
- $\Delta SA$is the spark retard timing distance from optimal spark advance, in deg.




**Dependencies**

To enable this parameter, for the **Torque model** parameter, select `Torque Structure`.

**`Spark retard from optimal, f_del_sa_bpt` — Breakpoints**
`[0 0.75 1.5 2.25 3 3.75 4.5 5.25 6 6.75 7.5 8.25 9 9.75 10.5 11.25 12 12.75 13.5 14.25 15 15.75 16.5 17.25 18 18.75 19.5 20.25 21 21.75 22.5 23.25 24 24.75 25.5 26.25 27 27.75 28.5 29.25 30 30.75 31.5 32.25 33 33.75 34.5 35.25 36 36.75 37.5 38.25 39 39.75 40.5 41.25 42 42.75 43.5 44.25 45 45.75 46.5 47.25 48]` (default) | `vector`

Spark retard from optimal inner torque timing breakpoints, in deg.

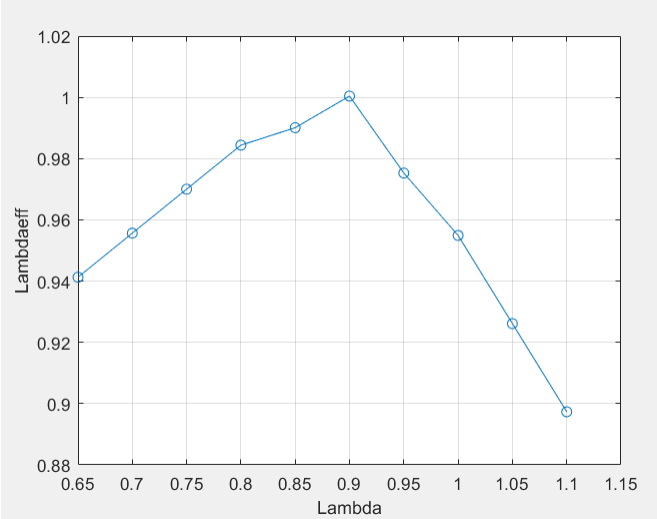
**Dependencies**

To enable this parameter, for the **Torque model** parameter, select `Torque Structure`.

**`Lambda efficiency, f_m_lam` — Lookup table**
`array`

The lambda efficiency lookup table, $f_{M\lambda}$, is a function of lambda, $M_\lambda = f_{M\lambda}(\lambda)$, where:

- $M_\lambda$ is the lambda multiplier on inner torque to account for the air-fuel ratio (AFR) effect, dimensionless.
- $\lambda$ is lambda, AFR normalized to stoichiometric fuel AFR, dimensionless.




**Dependencies**

To enable this parameter, for the **Torque model** parameter, select `Torque Structure`.

**`Lambda breakpoints, f_m_lam_bpt` — Breakpoints**
`[0.65 0.7 0.75 0.8 0.85 0.9 0.95 1 1.05 1.1]` (default) | `vector`

Lambda effect on inner torque lambda breakpoints, dimensionless.

**Dependencies**

To enable this parameter, for the **Torque model** parameter, select `Torque Structure`.

**Exhaust**

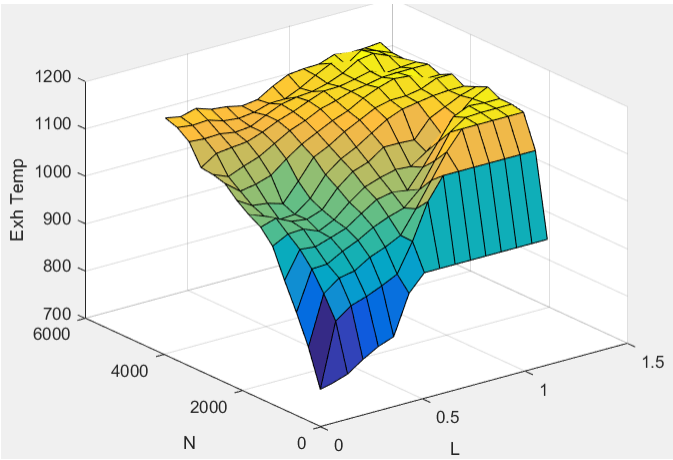
**`Exhaust temperature table, f_t_exh` — Lookup table**
`array`

The exhaust temperature lookup table, $f_{Texh}$, is a function of engine load and engine speed

$$T_{exh} = f_{Texh}(L, N)$$

where:

- $T_{exh}$ is engine exhaust temperature, in K.
- $L$ is normalized cylinder air mass or engine load, dimensionless.
- $N$ is engine speed, in rpm.




**`Load breakpoints, f_t_exh_l_bpt` — Breakpoints**
`[0.2 0.275 0.35 0.425 0.5 0.575 0.65 0.725 0.8 0.875 0.95 1.025 1.1 1.175 1.25]` (default) | `vector`

Engine load breakpoints used for exhaust temperature lookup table, dimensionless.

**`Speed breakpoints, f_t_exh_n_bpt` — Breakpoints**
`[750 1053.57142857143 1357.14285714286 1660.71428571429 1964.28571428571 2267.85714285714 2571.42857142857 2875 3178.57142857143 3482.14285714286 3785.71428571429 4089.28571428571 4392.85714285714 4696.42857142857 5000]` (default) | `vector`

Engine speed breakpoints used for exhaust temperature lookup table, in rpm.

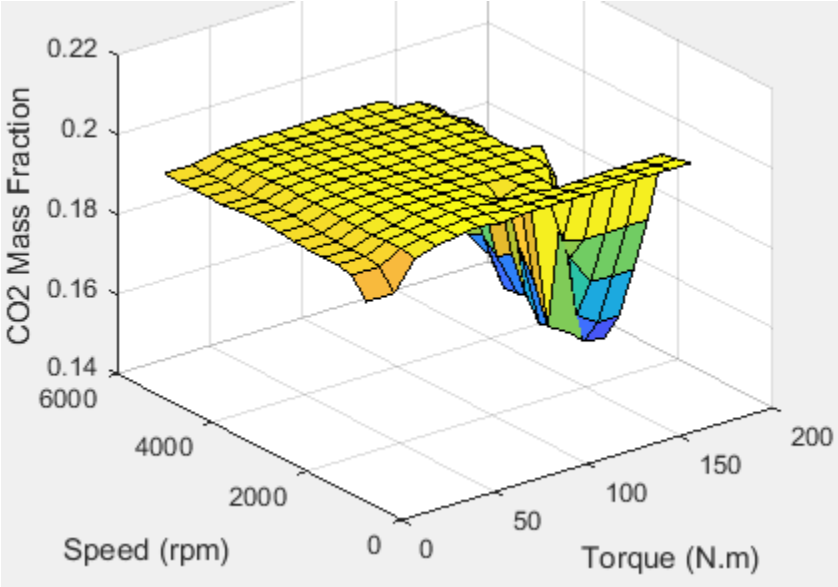
**`Exhaust gas specific heat at constant pressure, cp_exh` — Specific heat**
`1005` (default) | `scalar`

Exhaust gas-specific heat, $Cp_{exh}$, in J/(kg·K).

**`CO2 mass fraction table, f_CO2_frac` — Carbon dioxide ($CO_2$) emission lookup table**
`array`

The SI Core Engine $CO_2$ emission mass fraction lookup table is a function of engine torque and engine speed, *CO2 Mass Fraction* = *f*(*Speed*, *Torque*), where:

- *CO2 Mass Fraction* is the $CO_2$ emission mass fraction, dimensionless.
- *Speed* is engine speed, in rpm.
- *Torque* is engine torque, in N·m.




**Dependencies**

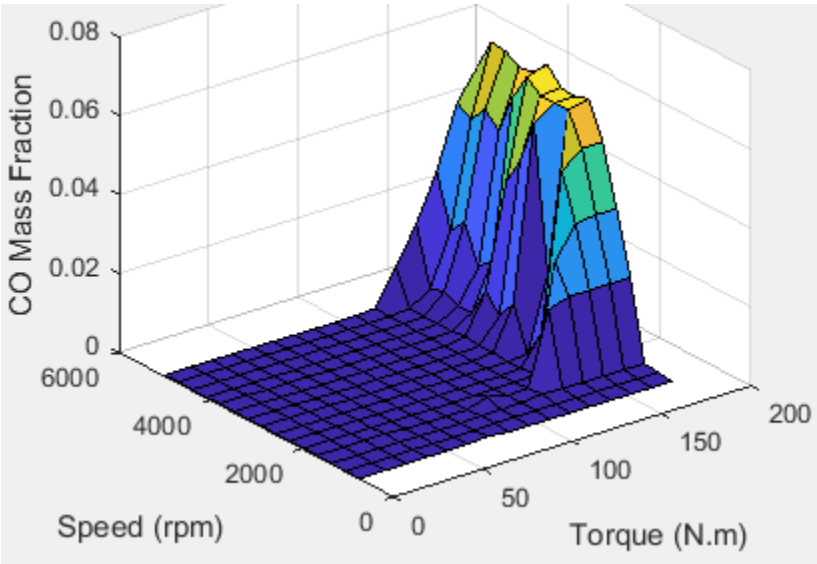
To enable this parameter, on the **Exhaust** tab, select **CO2**.

### `CO mass fraction table, f_CO_frac` — Carbon monoxide (CO) emission lookup table

`array`

The SI Core Engine CO emission mass fraction lookup table is a function of engine torque and engine speed, *CO Mass Fraction* = *f*(*Speed*, *Torque*), where:

- *CO Mass Fraction* is the CO emission mass fraction, dimensionless.
- *Speed* is engine speed, in rpm.
- *Torque* is engine torque, in N·m.



**Dependencies**

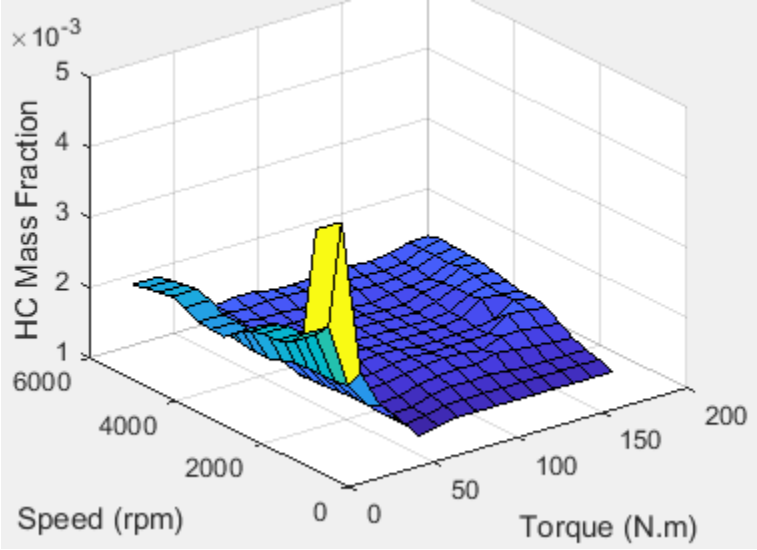
To enable this parameter, on the **Exhaust** tab, select **CO**.

### `HC mass fraction table, f_HC_frac` — Hydrocarbon (HC) emission lookup table

`array`

The SI Core Engine HC emission mass fraction lookup table is a function of engine torque and engine speed, *HC Mass Fraction* = *f*(*Speed*, *Torque*), where:

- *HC Mass Fraction* is the HC emission mass fraction, dimensionless.
- *Speed* is engine speed, in rpm.
- *Torque* is engine torque, in N·m.




**Dependencies**

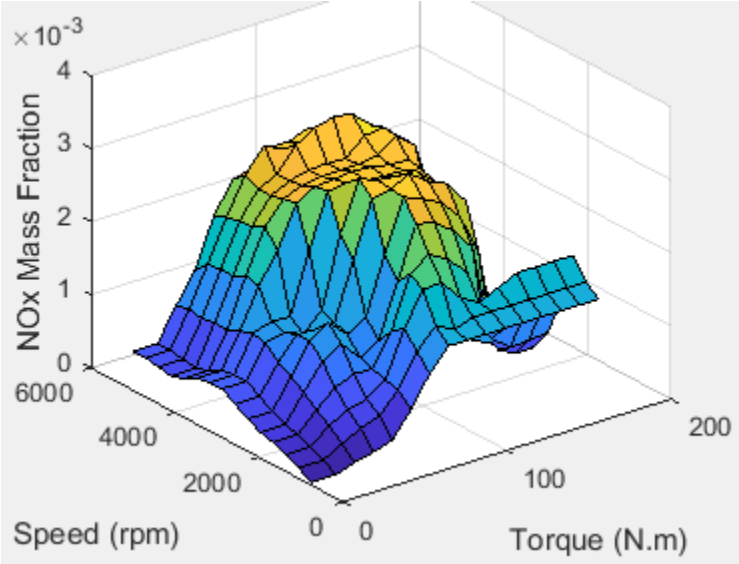
To enable this parameter, on the **Exhaust** tab, select **HC**.

### `NOx mass fraction table, f_NOx_frac` — Nitric oxide and nitrogen dioxide (NOx) emission lookup table

`array`

The SI Core Engine NOx emission mass fraction lookup table is a function of engine torque and engine speed, *NOx Mass Fraction* = *f*(*Speed*, *Torque*), where:

- *NOx Mass Fraction* is the NOx emission mass fraction, dimensionless.
- *Speed* is engine speed, in rpm.
- *Torque* is engine torque, in N·m.




**Dependencies**

To enable this parameter, on the **Exhaust** tab, select **NOx**.

**`PM mass fraction table, f_PM_frac` — Particulate matter (PM) emission lookup table**
`array`

The SI Core Engine PM emission mass fraction lookup table is a function of engine torque and engine speed where:

- *PM* is the PM emission mass fraction, dimensionless.
- *Speed* is engine speed, in rpm.
- *Torque* is engine torque, in N·m.

**Dependencies**

To enable this parameter, on the **Exhaust** tab, select **PM**.

**`Engine speed breakpoints, f_exhfrac_n_bpt` — Breakpoints**
`[750 1053.57142857143 1357.14285714286 1660.71428571429 1964.28571428571 2267.85714285714 2571.42857142857 2875 3178.57142857143 3482.14285714286 3785.71428571429 4089.28571428571 4392.85714285714 4696.42857142857 5000]` (default) | `vector`

Engine speed breakpoints used for the emission mass fractions lookup tables, in rpm.

**Dependencies**

To enable this parameter, on the **Exhaust** tab, select **CO2**, **CO**, **NOx**, **HC**, or **PM**.

**`Engine torque breakpoints, f_exhfrac_trq_bpt` — Breakpoints**
`[0 15 26.4285714285714 37.8571428571429 49.2857142857143 60.7142857142857 72.1428571428571 83.5714285714286 95 106.428571428571 117.857142857143 129.285714285714 140.714285714286 152.142857142857 163.571428571429 175]` (default) | `vector`

Engine torque breakpoints used for the emission mass fractions lookup tables, in N·m.

#### Dependencies

To enable this parameter, on the **Exhaust** tab, select **CO2**, **CO**, **NOx**, **HC**, or **PM**.

#### Fuel

**`Injector slope, Sinj` — Slope**
`6.45161290322581` (default) | `scalar`

Fuel injector slope, $S_{inj}$, mg/ms.

**`Stoichiometric air-fuel ratio, afr_stoich` — Air-fuel ratio**
`14.6` (default) | `scalar`

Air-fuel ratio, *AFR*.

**`Fuel lower heating value, fuel_lhv` — Heating value**
`46e6` (default) | `scalar`

Fuel lower heating value, *LHV*, in J/kg.

**`Fuel specific gravity, fuel_sg` — Specific gravity**
`0.745` (default) | `scalar`

Specific gravity of fuel, $Sg_{fuel}$, dimensionless.

## References

[1] Gerhardt, J., Hönninger, H., and Bischof, H., *A New Approach to Functional and Software Structure for Engine Management Systems — BOSCH ME7*. SAE Technical Paper 980801, 1998.

[2] Heywood, John B. *Internal Combustion Engine Fundamentals*. New York: McGraw-Hill, 1988.

## Extended Capabilities

#### C/C++ Code Generation
Generate C and C++ code using Simulink® Coder™.

## See Also
Mapped SI Engine | SI Controller

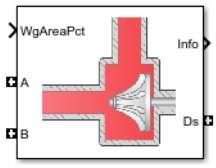
#### Topics
“SI Core Engine Air Mass Flow and Torque Production”
“Engine Calibration Maps”

#### Introduced in R2017a

# Turbine

Turbine for boosted engines

**Library:** Powertrain Blockset / Propulsion / Combustion Engine Components / Boost

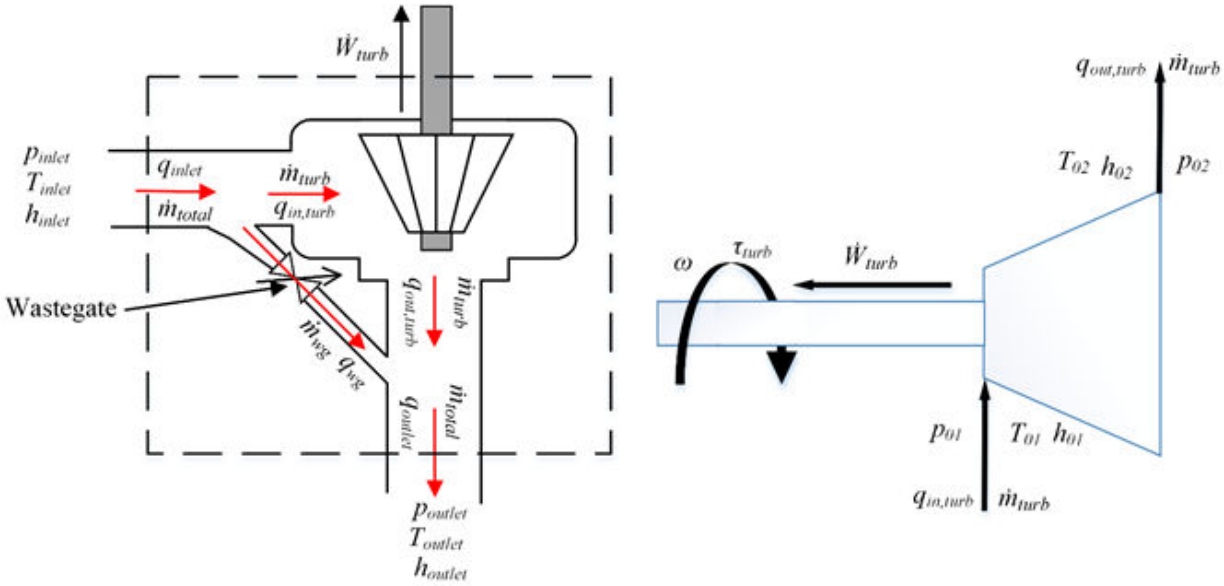



## Description

The Turbine block uses the conservation of mass and energy to calculate mass and heat flow rates for turbines with either fixed or variable geometry. You can configure the block with a wastegate valve to bypass the turbine. The block uses two-way ports to connect to the inlet and outlet control volumes and the drive shaft. You can specify the lookup tables to calculate the mass flow rate and turbine efficiency. Typically, turbine manufacturers provide the mass flow rate and efficiency tables as a function of corrected speed and pressure ratio. The block does not support reverse mass flow.

If you have Model-Based Calibration Toolbox, click **Calibrate Performance Maps** to virtually calibrate the mass flow rate and turbine efficiency lookup tables using measured data.

The mass flows from the inlet control volume to outlet control volume.




The Turbine block implements equations to model the performance, wastegate flow, and combined flow.

### Virtual Calibration

If you have Model-Based Calibration Toolbox, click **Calibrate Performance Maps** to virtually calibrate the corrected mass flow rate and turbine efficiency lookup tables using measured data. The dialog box steps through these tasks.

<table>
<tr><th>Task</th><th>Description</th></tr>
<tr><td>Import turbine data</td><td>Import this turbine data from a file. For more information, see “Using Data” (Model-Based Calibration Toolbox).
<table>
<tr><th>Turbine type</th><th>Data</th></tr>
<tr><td><code>Fixed geometry</code></td><td>
<ul>
<li>Speed, Spd, in rad/s</li>
<li>Corrected mass flow rate, MassFlwRate, in kg/s</li>
<li>Pressure ratio, PrsRatio, dimensionless</li>
<li>Efficiency, Eff, dimensionless</li>
</ul>
The speed, corrected mass flow rate, pressure ratio, and efficiency are in the 2nd-5th columns of the data file, respectively. The first and second rows of the data file provide the variable names and units. For example, use this format.
<table>
<tr><td>Name:</td><td>Spd</td><td>MassFlwRate</td><td>PrsRatio</td><td>Eff</td></tr>
<tr><td>Unit:</td><td>rad/s</td><td>kg/s</td><td></td><td></td></tr>
<tr><td>Data:</td><td><code>8373.3</code></td><td><code>0.02</code></td><td><code>1.21</code></td><td><code>0.44</code></td></tr>
<tr><td></td><td>...</td><td>...</td><td>...</td><td>...</td></tr>
</table>
</td></tr>
<tr><td><code>Variable geometry</code></td><td>
<ul>
<li>Speed, Spd, in rad/s</li>
<li>Corrected mass flow rate, MassFlwRate, kg/s</li>
<li>Rack position, RackPos, dimensionless</li>
<li>Pressure ratio, PrsRatio, dimensionless</li>
<li>Efficiency, Eff, dimensionless</li>
</ul>
Include data for several test points at each rack position operating point.<br>
The speed, corrected mass flow rate, rack position, pressure ratio, and efficiency are in the 2nd-6th columns of the data file, respectively. The first and second rows of the data file provide the variable names and units. For example, use this format.
<table>
<tr><td>Name:</td><td>Spd</td><td>MassFlwRate</td><td>RackPos</td><td>PrsRatio</td><td>Eff</td></tr>
<tr><td>Unit:</td><td>rad/s</td><td>kg/s</td><td></td><td></td><td></td></tr>
<tr><td>Data:</td><td><code>8373.3</code></td><td><code>0.02</code></td><td><code>1</code></td><td><code>1.21</code></td><td><code>0.44</code></td></tr>
<tr><td></td><td>...</td><td>...</td><td></td><td>...</td><td>...</td></tr>
</table>
</td></tr>
</table>
Model-Based Calibration Toolbox limits the speed and pressure ratio breakpoint values to the maximum values in the file.</td></tr>
</table>

<table>
<tr><th>Task</th><th>Description</th></tr>
<tr><td></td><td>To filter or edit the data, select Edit in Application. The Model-Based Calibration Toolbox Data Editor opens.</td></tr>
<tr><td>Generate response models</td><td>Model-Based Calibration Toolbox fits the imported data and generates response models.
<table>
<tr><th>Turbine type</th><th>Description</th></tr>
<tr><td><code>Fixed geometry</code></td><td>
<table>
<tr><th>Data</th><th>Response Model</th></tr>
<tr><td>Corrected mass flow rate</td><td>Square root turbine flow model described in Modeling and Control of Engines and Drivelines[2]</td></tr>
<tr><td>Efficiency</td><td>Blade speed ratio (BSR) model described in Modeling and Control of Engines and Drivelines[2]</td></tr>
</table>
</td></tr>
<tr><td><code>Variable geometry</code></td><td>Model-Based Calibration Toolbox uses a point-by-point test plan to fit the data. For each rack position, the block uses these response models to fit the corrected mass flow rate and efficiency data.
<table>
<tr><th>Data</th><th>Response Model</th></tr>
<tr><td>Corrected mass flow rate</td><td>Square root turbine flow model described in Modeling and Control of Engines and Drivelines[2]</td></tr>
<tr><td>Efficiency</td><td>Blade speed ratio (BSR) model described in Modeling and Control of Engines and Drivelines[2]</td></tr>
</table>
</td></tr>
</table>
To assess or adjust the response model fit, select Edit in Application. The Model-Based Calibration Toolbox Model Browser opens. For more information, see “Model Assessment” (Model-Based Calibration Toolbox).</td></tr>
<tr><td>Generate calibration</td><td>Model-Based Calibration Toolbox calibrates the response model and generates calibrated tables.
<table>
<tr><th>Turbine type</th><th>Description</th></tr>
<tr><td><code>Fixed geometry</code></td><td>Model-Based Calibration Toolbox uses the response models for the corrected mass flow rate and efficiency tables.</td></tr>
<tr><td><code>Variable geometry</code></td><td>Model-Based Calibration Toolbox fills the corrected mass flow rate and efficiency tables for each rack position. Model-Based Calibration Toolbox then combines the rack position-dependent tables into 3D lookup tables for corrected mass flow rate and efficiency.</td></tr>
</table>
To assess or adjust the calibration, select Edit in Application. The Model-Based Calibration Toolbox CAGE Browser opens. For more information, see “Calibration Lookup Tables” (Model-Based Calibration Toolbox).</td></tr>
</table>

<table>
<tr><th>Task</th><th>Description</th></tr>
<tr><td>Update block parameters</td><td>Update these corrected mass flow rate and efficiency parameters with the calibration.
<table>
<tr><th>Turbine type</th><th>Parameters</th></tr>
<tr><td><code>Fixed geometry</code></td><td>• Corrected mass flow rate table, mdot_corrfx_tbl<br>• Efficiency table, eta_turbfx_tbl<br>• Corrected speed breakpoints, w_corrfx_bpts1<br>• Pressure ratio breakpoints, Pr_fx_bpts2</td></tr>
<tr><td><code>Variable geometry</code></td><td>• Corrected mass flow rate table, mdot_corrvr_tbl<br>• Efficiency table, eta_turbvr_tbl<br>• Corrected speed breakpoints, w_corrvr_bpts2<br>• Pressure ratio breakpoints, Pr_vr_bpts2<br>• Rack breakpoints, L_rack_bpts3</td></tr>
</table></td></tr>
</table>

### Thermodynamics

The block uses these equations to model the thermodynamics.

| Calculation | Equations |
|---|---|
| Forward mass flow | $\dot{m}_{turb} > 0$<br>$p_{01} = p_{inlet}$<br>$p_{02} = p_{outlet}$<br>$T_{01} = T_{inlet}$<br>$h_{01} = h_{inlet}$ |
| First law of thermodynamics | $\dot{W}_{turb} = \dot{m}_{turb} c_p (T_{01} - T_{02})$ |
| Isentropic efficiency | $\eta_{turb} = \frac{h_{01} - h_{02}}{h_{01} - h_{02s}} = \frac{T_{01} - T_{02}}{T_{01} - T_{02s}}$ |
| Isentropic outlet temperature, assuming ideal gas, and constant specific heats | $T_{02s} = T_{01}\left(\frac{p_{02}}{p_{01}}\right)^{\frac{\gamma - 1}{\gamma}}$ |
| Specific heat ratio | $\gamma = \frac{c_p}{c_p - R}$ |
| Outlet temperature | $T_{02} = T_{01} + \eta_{turb} T_{01}\left\{1 - \left(\frac{p_{02}}{p_{01}}\right)^{\frac{\gamma - 1}{\gamma}}\right\}$ |
| Heat flows | $q_{in,\,turb} = \dot{m}_{turb} c_p T_{01}$<br>$q_{out,\,turb} = \dot{m}_{turb} c_p T_{02}$ |

| Calculation | Equations |
|---|---|
| Drive shaft torque | $\tau_{turb} = \frac{\dot{W}_{turb}}{\omega}$ |

The equations use these variables.

| | |
|---|---|
| $p_{\text{inlet}}$,$p_{01}$ | Inlet control volume total pressure |
| $T_{inlet}$, $T_{01}$ | Inlet control volume total temperature |
| $h_{inlet}$, $h_{01}$ | Inlet control volume total specific enthalpy |
| $p_{outlet}$, $p_{02}$ | Outlet control volume total pressure |
| $T_{outlet}$ | Outlet control volume total temperature |
| $h_{outlet}$ | Outlet control volume total specific enthalpy |
| $\dot{W}_{turb}$ | Drive shaft power |
| $T_{02}$ | Temperature exiting the turbine |
| $h_{02}$ | Outlet total specific enthalpy |
| $\dot{m}_{turb}$ | Turbine mass flow rate |
| $q_{in,\,turb}$ | Turbine inlet heat flow rate |
| $q_{out,\,turb}$ | Turbine outlet heat flow rate |
| $\eta_{turb}$ | Turbine isentropic efficiency |
| $T_{02s}$ | Isentropic outlet total temperature |
| $h_{02s}$ | Isentropic outlet total specific enthalpy |
| $R$ | Ideal gas constant |
| $c_p$ | Specific heat at constant pressure |
| $\gamma$ | Specific heat ratio |
| $\tau_{turb}$ | Drive shaft torque |

### Performance Lookup Tables

The block implements lookup tables based on these equations.

| Calculation | Equation | |
|---|---|---|
| Corrected mass flow rate | $\dot{m}_{corr} = \dot{m}_{turb}\frac{\sqrt{T_{01}/T_{ref}}}{p_{01}/p_{ref}}$ | |
| Corrected speed | $\omega_{corr} = \frac{\omega}{\sqrt{T_{01}/T_{ref}}}$ | |
| Pressure expansion ratio | $p_r = \frac{p_{01}}{p_{02}}$ | |
| Efficiency lookup table | Fixed geometry (3-D table) | $\eta_{turbfx,\,tbl} = f(\omega_{corr}, p_r)$ |
| | Variable geometry (3-D table) | $\eta_{turbvr,\,tbl} = f(\omega_{corr}, p_r, L_{rack})$ |

| Calculation | Equation | |
|---|---|---|
| Corrected mass flow lookup table | Fixed geometry (3-D table) | $\dot{m}_{corrfx,tbl} = f(\omega_{corr}, p_r)$ |
| | Variable geometry (3-D table) | $\dot{m}_{corrvr,tbl} = f(\omega_{corr}, p_r, L_{rack})$ |

The equations use these variables.

| | |
|---|---|
| $p_{01}$ | Inlet control volume total pressure |
| $p_r$ | Pressure expansion ratio |
| $p_{02}$ | Outlet control volume total pressure |
| $P_{ref}$ | Lookup table reference pressure |
| $T_{01}$ | Inlet control volume total temperature |
| $T_{ref}$ | Lookup table reference temperature |
| $\dot{m}_{turb}$ | Turbine mass flow rate |
| $\omega$ | Drive shaft speed |
| $\omega_{corr}$ | Corrected drive shaft speed |
| $L_{rack}$ | Variable geometry turbine rack position |
| $\eta_{turbfx,tbl}$ | Efficiency 3-D lookup table for fixed geometry |
| $\dot{m}_{corrfx,tbl}$ | Corrected mass flow rate 3-D lookup table for fixed geometry |
| $\eta_{turbvr,tbl}$ | Efficiency 3-D lookup table for variable geometry |
| $\dot{m}_{corrvr,tbl}$ | Corrected mass flow rate 3-D lookup table for variable geometry |

### Wastegate

To calculate the wastegate heat and mass flow rates, the Turbine block uses a Flow Restriction block. The Flow Restriction block uses the wastegate flow area.

$$A_{wg} = A_{wgpctcmd}\frac{A_{wgopen}}{100}$$

The equation uses these variables.

| | |
|---|---|
| $A_{wgpctcmd}$ | Wastegate valve area percent command |
| $A_{wg}$ | Wastegate valve area |
| $A_{wgopen}$ | Wastegate valve area when fully open |

### Combined Flow

To represent flow through the wastegate valve and turbine, the block uses these equations.

| Calculation | Equations |
|---|---|
| Blocks not configured with a wastegate valve | $\dot{m}_{wg} = q_{wg} = 0$ |
| Total mass flow rate | $\dot{m}_{total} = \dot{m}_{turb} + \dot{m}_{wg}$ |

| Calculation | Equations |
|---|---|
| Total heat flow rate | $q_{inlet} = q_{in,turb} + q_{wg}$<br><br>$q_{outlet} = q_{out,turb} + q_{wg}$ |
| Combined temperature exiting the wastegate valve and turbine | $T_{outflw} = \begin{cases} \frac{q_{outlet}}{\dot{m}_{total} c_p} & \dot{m}_{total} > \dot{m}_{thresh} \\ \frac{T_{02} + T_{outflw,wg}}{2} & else \end{cases}$ |

The block uses the internal signal `FlwDir` to track the direction of the flow.

The equations use these variables.

| | |
|---|---|
| $\dot{m}_{total}$ | Total mass flow rate through the wastegate valve and turbine |
| $\dot{m}_{turb}$ | Turbine mass flow rate |
| $\dot{m}_{wg}$ | Mass flow rate through the wastegate valve |
| $q_{inlet}$ | Total inlet heat flow rate |
| $q_{outlet}$ | Total outlet heat flow rate |
| $q_{in,turb}$ | Turbine inlet heat flow rate |
| $q_{out,turb}$ | Turbine outlet heat flow rate |
| $q_{wg}$ | Wastegate valve heat flow rate |
| $T_{02}$ | Temperature exiting the turbine |
| $T_{outflw}$ | Total temperature exiting the block |
| $T_{outflw,wg}$ | Temperature exiting the wastegate valve |
| $\dot{m}_{thresh}$ | Mass flow rate threshold to prevent dividing by zero |
| $c_p$ | Specific heat at constant pressure |

### Power Accounting

For the power accounting, the block implements these equations.

| Bus Signal | | | Description | Equations |
|---|---|---|---|---|
| `PwrInfo` | `PwrTrnsfrd` — Power transferred between blocks<br>• Positive signals indicate flow into block<br>• Negative signals indicate flow out of block | `PwrDriveshft` | Power transmitted from the shaft | $-\dot{W}_{turb}$ |
| | | `PwrHeatFlwIn` | Heat flow rate at port A | $q_{outlet}$ |
| | | `PwrHeatFlwOut` | Heat flow rate at port B | $q_{outlet}$ |

| Bus Signal | | | Description | Equations |
|---|---|---|---|---|
| | `PwrNotTrnsfrd` — Power crossing the block boundary, but not transferred<br>• Positive signals indicate an input<br>• Negative signals indicate a loss | `PwrLoss` | Power loss | $-q_{inlet}$<br>$-q_{outlet}$<br>$+\dot{W}_{turb}$ |
| | `PwrStored` — Stored energy rate of change<br>• Positive signals indicate an increase<br>• Negative signals indicate a decrease | *Not used* | | |

The equations use these variables.

| | |
|---|---|
| $\dot{W}_{turb}$ | Drive shaft power |
| $q_{outlet}$ | Total outlet heat flow rate |
| $q_{inlet}$ | Total inlet heat flow rate |

## Ports

### Input

#### `Ds` — Drive shaft speed

two-way connector port

`ShaftSpd` — Signal containing the drive shaft angular speed, $\omega$, in rad/s.

#### `A` — Inlet pressure, temperature, enthalpy, mass fractions

two-way connector port

Bus containing the inlet control volume:

- `InPrs` — Pressure, $p_{\text{inlet}}$, in Pa
- `InTemp` — Temperature, $T_{inlet}$, in K
- `InEnth` — Specific enthalpy, $h_{inlet}$, in J/kg

#### `B` — Outlet pressure, temperature, enthalpy, mass fractions

two-way connector port

Bus containing the outlet control volume:

- `OutPrs` — Pressure, $p_{outlet}$, in Pa
- `OutTemp` — Temperature, $T_{outlet}$, in K
- `OutEnth` — Specific enthalpy, $h_{outlet}$, in J/kg

### RackPos — Rack position

scalar

Variable geometry turbine rack position, $L_{rack}$.

#### Dependencies

To create this port, select `Variable geometry` for the **Turbine type** parameter.

### WgAreaPct — Wastegate area percent

scalar

Wastegate valve area percent, $A_{wgpctcmd}$.

#### Dependencies

To create this port, select **Include wastegate**.

## Output

### Info — Bus signal

bus

Bus signal containing these block calculations.

| Signal | | | Description | Units |
|---|---|---|---|---|
| TurbOutletTemp | | | Temperature exiting the turbine | K |
| DriveshftPwr | | | Drive shaft power | W |
| DriveshftTrq | | | Drive shaft torque | N·m |
| TurbMassFlw | | | Turbine mass flow rate | kg/s |
| PrsRatio | | | Pressure ratio | N/A |
| DriveshftCorrSpd | | | Corrected drive shaft speed | rad/s |
| TurbEff | | | Turbine isentropic efficiency | N/A |
| CorrMassFlw | | | Corrected mass flow rate | kg/s |
| WgArea | | | Wastegate valve area | m^2 |
| WgMassFlw | | | Mass flow rate through the wastegate valve | kg/s |
| WgOutletTemp | | | Temperature exiting the wastegate valve | K |
| PwrInfo | PwrTrnsfrd | PwrDriveshft | Power transmitted from the shaft | W |
| | | PwrHeatFlwIn | Heat flow rate at port A | W |
| | | PwrHeatFlwOut | Heat flow rate at port B | W |
| | PwrNotTrnsfrd | PwrLoss | Power loss | W |
| | PwrStored | | *Not used* | |

### Ds — Drive shaft torque

two-way connector port

`Trq` — Signal containing the drive shaft torque, $\tau_{turb}$, in N·m.

**`A` — Inlet mass flow rate, heat flow rate, temperature, mass fractions**
two-way connector port

Bus containing:

- `MassFlwRate` — Total mass flow rate through wastegate valve and turbine, $-\dot{m}_{total}$, in kg/s
- `HeatFlwRate` — Total inlet heat flow rate, $-q_{inlet}$, in J/s
- `Temp` — Total inlet temperature, $T_{inlet}$, in K
- `MassFrac` — Mass fractions, dimensionless.

  Specifically, a bus with these mass fractions:

  - `O2MassFrac` — Oxygen
  - `N2MassFrac` — Nitrogen
  - `UnbrndFuelMassFrac` — Unburned fuel
  - `CO2MassFrac` — Carbon dioxide
  - `H2OMassFrac` — Water
  - `COMassFrac` — Carbon monoxide
  - `NOMassFrac` — Nitric oxide
  - `NO2MassFrac` — Nitrogen dioxide
  - `NOxMassFrac` — Nitric oxide and nitrogen dioxide
  - `PmMassFrac` — Particulate matter
  - `AirMassFrac` — Air
  - `BrndGasMassFrac` — Burned gas

**`B` — Outlet mass flow rate, heat flow rate, temperature, mass fractions**
two-way connector port

Bus containing:

- `MassFlwRate` — Turbine mass flow rate through wastegate valve and turbine, $\dot{m}_{turb}$, in kg/s
- `HeatFlwRate` — Total outlet heat flow rate, $q_{outlet}$, in J/s
- `Temp` — Total outlet temperature, $T_{outflw}$, in K
- `MassFrac` — Mass fractions, dimensionless.

  Specifically, a bus with these mass fractions:

  - `O2MassFrac` — Oxygen
  - `N2MassFrac` — Nitrogen
  - `UnbrndFuelMassFrac` — Unburned fuel
  - `CO2MassFrac` — Carbon dioxide
  - `H2OMassFrac` — Water
  - `COMassFrac` — Carbon monoxide

- `NOMassFrac` — Nitric oxide
- `NO2MassFrac` — Nitrogen dioxide
- `NOxMassFrac` — Nitric oxide and nitrogen dioxide
- `PmMassFrac` — Particulate matter
- `AirMassFrac` — Air
- `BrndGasMassFrac` — Burned gas

## Parameters

#### Block Options

**`Turbine type` — Select turbine type**
`Fixed geometry` (default) | `Variable geometry`

Turbine type.

#### Dependencies

The table summarizes the parameter and port dependencies.

| Value | Enables Parameters | Creates Ports |
|---|---|---|
| `Fixed geometry` | **Corrected mass flow rate table, mdot_corrfx_tbl**<br>**Efficiency table, eta_turbfx_tbl**<br>**Corrected speed breakpoints, w_corrfx_bpts1**<br>**Pressure ratio breakpoints, Pr_fx_bpts2** | None |
| `Variable geometry` | **Corrected mass flow rate table, mdot_corrvr_tbl**<br>**Efficiency table, eta_turbvr_tbl**<br>**Corrected speed breakpoints, w_corrvr_bpts2**<br>**Pressure ratio breakpoints, Pr_vr_bpts2**<br>**Rack breakpoints, L_rack_bpts3** | RP |

**`Include wastegate` — Select**
on (default) | off

#### Dependencies

Selecting the **Include wastegate** parameter enables:

- **Wastegate flow area, A_wgopen**
- **Pressure ratio linearize limit, Plim_wg**

#### Performance Tables

**`Calibrate Performance Maps` — Calibrate tables with measured data selection**

If you have Model-Based Calibration Toolbox, click **Calibrate Performance Maps** to virtually calibrate the corrected mass flow rate and turbine efficiency lookup tables using measured data. The dialog box steps through these tasks.

<table>
<tr><th>Task</th><th>Description</th></tr>
<tr><td>Import turbine data</td><td>Import this turbine data from a file. For more information, see “Using Data” (Model-Based Calibration Toolbox).
<table>
<tr><th>Turbine type</th><th>Data</th></tr>
<tr><td><code>Fixed geometry</code></td><td>
<ul>
<li>Speed, Spd, in rad/s</li>
<li>Corrected mass flow rate, MassFlwRate, in kg/s</li>
<li>Pressure ratio, PrsRatio, dimensionless</li>
<li>Efficiency, Eff, dimensionless</li>
</ul>
The speed, corrected mass flow rate, pressure ratio, and efficiency are in the 2nd-5th columns of the data file, respectively. The first and second rows of the data file provide the variable names and units. For example, use this format.
<table>
<tr><td>Name:</td><td>Spd</td><td>MassFlwRate</td><td>PrsRatio</td><td>Eff</td></tr>
<tr><td>Unit:</td><td>rad/s</td><td>kg/s</td><td></td><td></td></tr>
<tr><td>Data:</td><td><code>8373.3</code></td><td><code>0.02</code></td><td><code>1.21</code></td><td><code>0.44</code></td></tr>
<tr><td></td><td>...</td><td>...</td><td>...</td><td>...</td></tr>
</table>
</td></tr>
<tr><td><code>Variable geometry</code></td><td>
<ul>
<li>Speed, Spd, in rad/s</li>
<li>Corrected mass flow rate, MassFlwRate, kg/s</li>
<li>Rack position, RackPos, dimensionless</li>
<li>Pressure ratio, PrsRatio, dimensionless</li>
<li>Efficiency, Eff, dimensionless</li>
</ul>
Include data for several test points at each rack position operating point.<br><br>
The speed, corrected mass flow rate, rack position, pressure ratio, and efficiency are in the 2nd-6th columns of the data file, respectively. The first and second rows of the data file provide the variable names and units. For example, use this format.
<table>
<tr><td>Name:</td><td>Spd</td><td>MassFlwRate</td><td>RackPos</td><td>PrsRatio</td><td>Eff</td></tr>
<tr><td>Unit:</td><td>rad/s</td><td>kg/s</td><td></td><td></td><td></td></tr>
<tr><td>Data:</td><td><code>8373.3</code></td><td><code>0.02</code></td><td><code>1</code></td><td><code>1.21</code></td><td><code>0.44</code></td></tr>
<tr><td></td><td>...</td><td>...</td><td></td><td>...</td><td>...</td></tr>
</table>
</td></tr>
</table>
Model-Based Calibration Toolbox limits the speed and pressure ratio breakpoint values to the maximum values in the file.</td></tr>
</table>

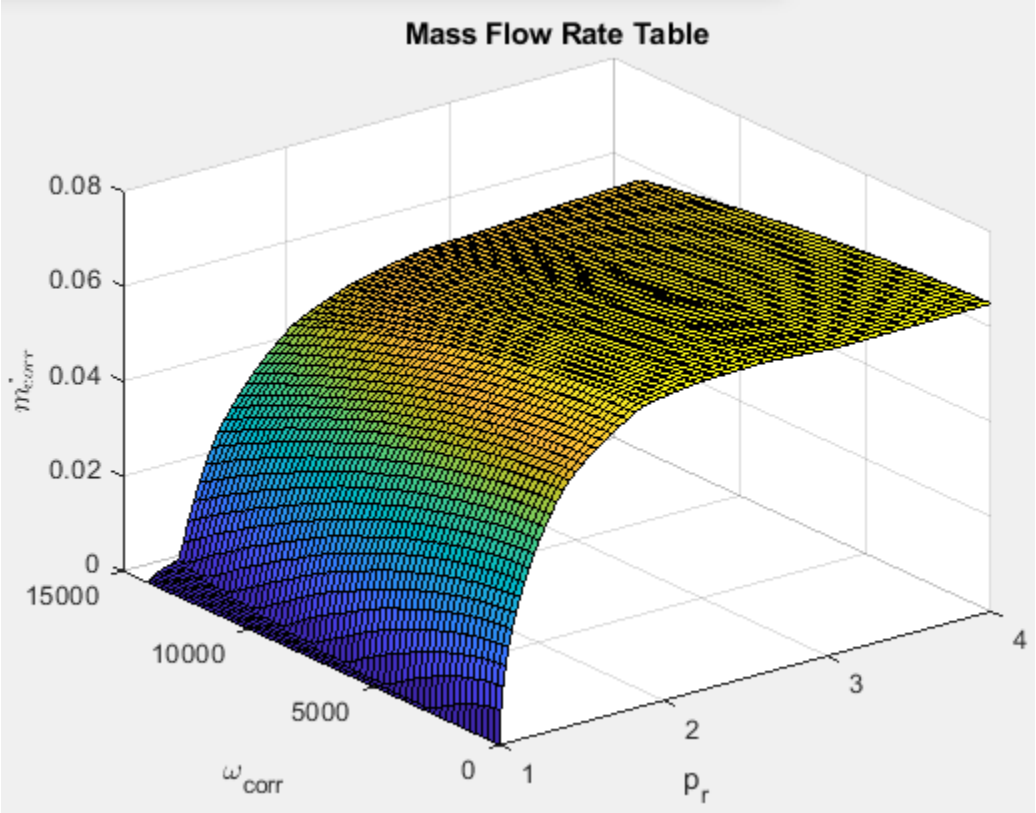
<table>
<tr><th>Task</th><th>Description</th></tr>
<tr><td></td><td>To filter or edit the data, select Edit in Application. The Model-Based Calibration Toolbox Data Editor opens.</td></tr>
<tr><td>Generate response models</td><td>Model-Based Calibration Toolbox fits the imported data and generates response models.
<table>
<tr><th>Turbine type</th><th>Description</th></tr>
<tr><td><code>Fixed geometry</code></td><td>
<table>
<tr><th>Data</th><th>Response Model</th></tr>
<tr><td>Corrected mass flow rate</td><td>Square root turbine flow model described in Modeling and Control of Engines and Drivelines[2]</td></tr>
<tr><td>Efficiency</td><td>Blade speed ratio (BSR) model described in Modeling and Control of Engines and Drivelines[2]</td></tr>
</table>
</td></tr>
<tr><td><code>Variable geometry</code></td><td>Model-Based Calibration Toolbox uses a point-by-point test plan to fit the data. For each rack position, the block uses these response models to fit the corrected mass flow rate and efficiency data.
<table>
<tr><th>Data</th><th>Response Model</th></tr>
<tr><td>Corrected mass flow rate</td><td>Square root turbine flow model described in Modeling and Control of Engines and Drivelines[2]</td></tr>
<tr><td>Efficiency</td><td>Blade speed ratio (BSR) model described in Modeling and Control of Engines and Drivelines[2]</td></tr>
</table>
</td></tr>
</table>
To assess or adjust the response model fit, select Edit in Application. The Model-Based Calibration Toolbox Model Browser opens. For more information, see “Model Assessment” (Model-Based Calibration Toolbox).</td></tr>
<tr><td>Generate calibration</td><td>Model-Based Calibration Toolbox calibrates the response model and generates calibrated tables.
<table>
<tr><th>Turbine type</th><th>Description</th></tr>
<tr><td><code>Fixed geometry</code></td><td>Model-Based Calibration Toolbox uses the response models for the corrected mass flow rate and efficiency tables.</td></tr>
<tr><td><code>Variable geometry</code></td><td>Model-Based Calibration Toolbox fills the corrected mass flow rate and efficiency tables for each rack position. Model-Based Calibration Toolbox then combines the rack position-dependent tables into 3D lookup tables for corrected mass flow rate and efficiency.</td></tr>
</table>
To assess or adjust the calibration, select Edit in Application. The Model-Based Calibration Toolbox CAGE Browser opens. For more information, see “Calibration Lookup Tables” (Model-Based Calibration Toolbox).</td></tr>
</table>

| Task | Description |
|---|---|
| Update block parameters | Update these corrected mass flow rate and efficiency parameters with the calibration. |

| Turbine type | Parameters |
|---|---|
| `Fixed geometry` | • **Corrected mass flow rate table, mdot_corrfx_tbl**<br>• **Efficiency table, eta_turbfx_tbl**<br>• **Corrected speed breakpoints, w_corrfx_bpts1**<br>• **Pressure ratio breakpoints, Pr_fx_bpts2** |
| `Variable geometry` | • **Corrected mass flow rate table, mdot_corrvr_tbl**<br>• **Efficiency table, eta_turbvr_tbl**<br>• **Corrected speed breakpoints, w_corrvr_bpts2**<br>• **Pressure ratio breakpoints, Pr_vr_bpts2**<br>• **Rack breakpoints, L_rack_bpts3** |

**`Corrected mass flow rate table, mdot_corrfx_tbl` — Lookup table**
`array`

Corrected mass flow rate lookup table for fixed geometry, $\dot{m}_{corrfx,tbl}$, as a function of corrected driveshaft speed, $\omega_{corr}$, and pressure ratio, $p_r$, in kg/s.



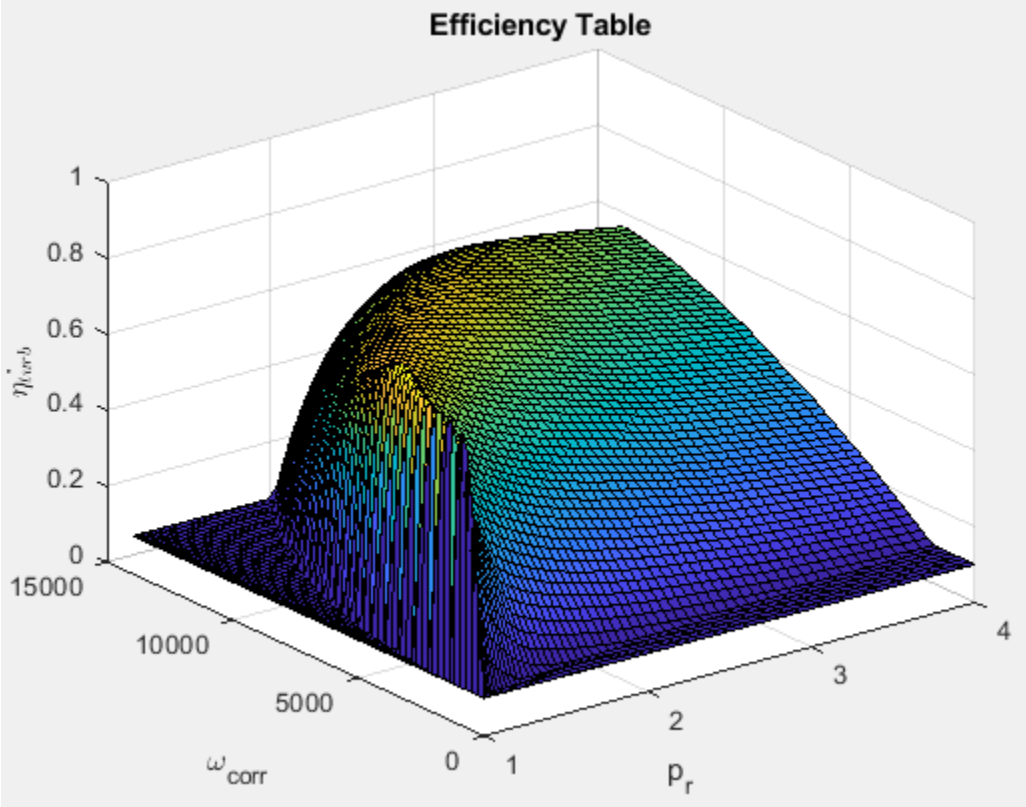
**Dependencies**

To enable this parameter, select `Fixed geometry` for the **Turbine type** parameter.

### `Efficiency table, eta_turbfx_tb` — Lookup table

`array`

Efficiency lookup table for fixed geometry, $\eta_{turbfx,tbl}$, as a function of corrected driveshaft speed, $\omega_{corr}$, and pressure ratio, $p_r$, dimensionless.




**Dependencies**

To enable this parameter, select `Fixed geometry` for the **Turbine type** parameter.

### `Corrected speed breakpoints, w_corrfx_bpts1` — Fixed geometry

`[0 1552 3104 4657 6209 7761 9313 1.087e+04 1.242e+04 1.397e+04]` (default) | `vector`

Corrected drive shaft speed breakpoints for fixed geometry, $\omega_{corrfx,bpts1}$, in rad/s.

**Dependencies**

To enable this parameter, select `Fixed geometry` for the **Turbine type** parameter.

### `Pressure ratio breakpoints, Pr_fx_bpts2` — Fixed geometry

`[1 1.333 1.667 2 2.333 2.667 3 3.333 3.667 4]` (default) | `vector`

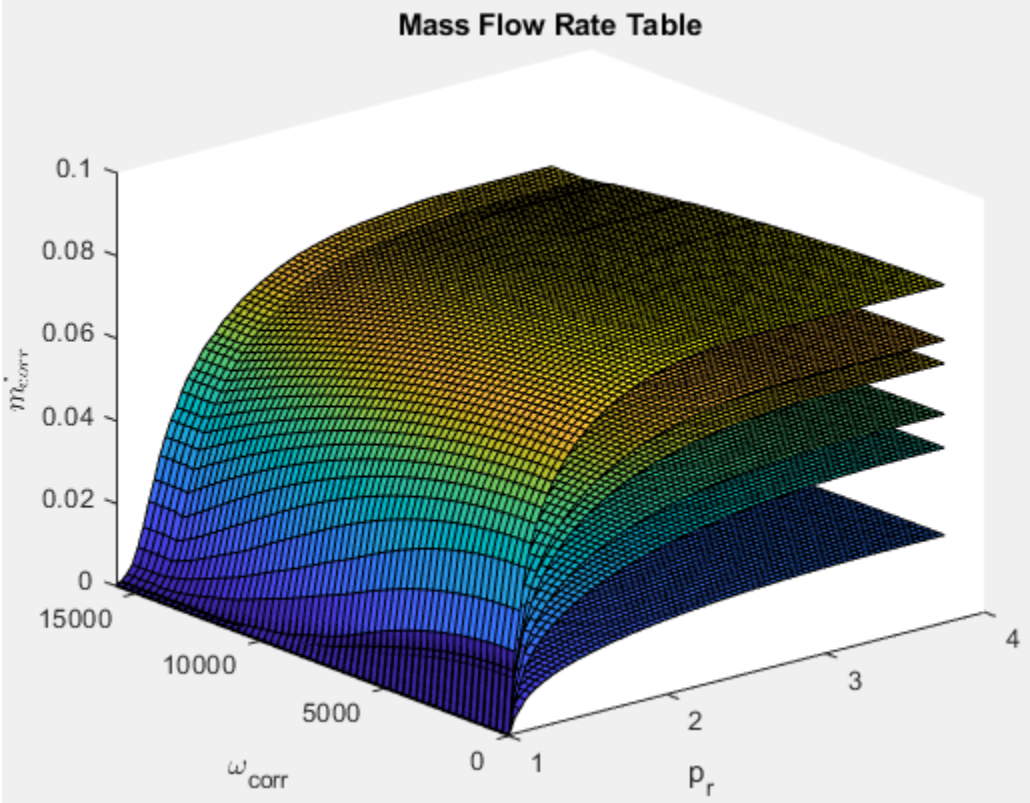
Pressure ratio breakpoints for fixed geometry, $p_{rfx,bpts2}$.

**Dependencies**

To enable this parameter, select `Fixed geometry` for the **Turbine type** parameter.

**`Corrected mass flow rate table, mdot_corrvr_tbl` — Lookup table**
`array`

Corrected mass flow rate lookup table for variable geometry, $\dot{m}_{corrvr,tbl}$, as a function of corrected driveshaft speed, $\omega_{corr}$, and pressure ratio, $p_r$, in kg/s.

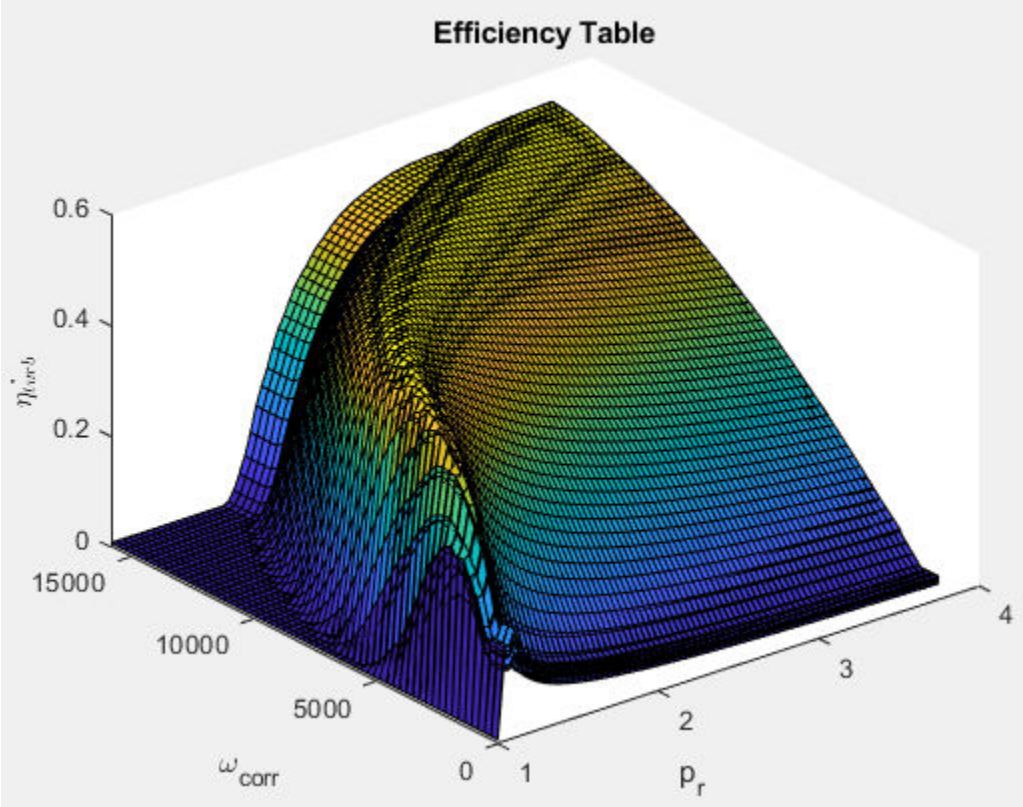



**Dependencies**

To enable this parameter, select `Variable geometry` for the **Turbine type** parameter.

**`Efficiency table, eta_turbvr_tbl` — Lookup table**
`array`

Efficiency lookup table for variable geometry, $\eta_{turbvr,tbl}$, as a function of corrected driveshaft speed, $\omega_{corr}$, and pressure ratio, $p_r$, dimensionless.




**Dependencies**

To enable this parameter, select `Variable geometry` for the **Turbine type** parameter.

**`Corrected speed breakpoints, w_corrvr_bpts2` — Variable geometry**
`[0 1752 3504 5257 7009 8761 1.051e+04 1.227e+04 1.402e+04 1.577e+04]` (default) | `vector`

Corrected drive shaft speed breakpoints for variable geometry, $\omega_{corrvr,bpts1}$, in rad/s.

**Dependencies**

To enable this parameter, select `Variable geometry` for the **Turbine type** parameter.

**`Pressure ratio breakpoints, Pr_vr_bpts2` — Variable geometry**
`[1 1.306 1.611 1.917 2.222 2.528 2.833 3.139 3.444 3.75]` (default) | `vector`

Pressure ratio breakpoints for variable geometry.

**Dependencies**

To enable this parameter, select `Variable geometry` for the **Turbine type** parameter.

**`Rack breakpoints, L_rack_bpts3` — Variable geometry**
`[0 0.2 0.3 0.5 0.7 1]` (default) | `vector`

Rack position breakpoints for variable geometry, $L_{rack,bpts3}$.

#### Dependencies

To enable this parameter, select `Variable geometry` for the **Turbine type** parameter.

**`Reference temperature, T_ref` — Temperature**
`293.15` (default) | `scalar`

Performance map reference temperature, $T_{ref}$, in K.

**`Reference pressure, P_ref` — Pressure**
`101325` (default) | `scalar`

Performance map reference pressure, $P_{ref}$, in Pa.

#### Wastegate

**`Wastegate flow area, A_wgopen` — Area**
`0.0003` (default) | `scalar`

Area of fully opened wastegate valve, $A_{wgopen}$, in m^2.

#### Dependencies

To enable **Wastegate flow area, A_wgopen**, select the **Include wastegate** parameter.

**`Pressure ratio linearize limit, Plim_wg` — Area, m^2**
`0.95` (default) | `scalar`

#### Dependencies

Flow restriction linearization limit, $p_{lim,\,wg}$.

To enable **Pressure ratio linearize limit, Plim_wg**, select the **Include wastegate** parameter.

#### Properties

**`Ideal gas constant, R` — Constant**
`287` (default) | `scalar`

Ideal gas constant $R$, in J/(kg·K).

**`Specific heat at constant pressure, cp` — Specific heat**
`1005` (default) | `scalar`

Specific heat at constant pressure, $c_p$, in J/(kg·K).

## References

[1] Heywood, John B. *Internal Combustion Engine Fundamentals*. New York: McGraw-Hill, 1988.

[2] Eriksson, Lars and Lars Nielsen. *Modeling and Control of Engines and Drivelines*. Chichester, West Sussex, United Kingdom: John Wiley & Sons Ltd, 2014.

## Extended Capabilities

### C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

## See Also

Boost Drive Shaft | Compressor

### Topics

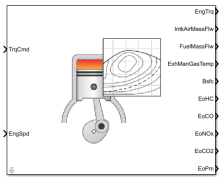
“Model-Based Calibration Toolbox”

### Introduced in R2017a

# Mapped Core Engine

Steady-state core engine model using lookup tables

**Library:** Powertrain Blockset / Propulsion / Combustion Engine Components / Core Engine




## Description

The Mapped Core Engine block implements a steady-state core engine model using power, air mass flow, fuel flow, exhaust temperature, efficiency, and emission performance lookup tables. You can use the block for:

- Hardware-in-the-loop (HIL) engine control design.
- Vehicle-level fuel economy and performance simulations.

The block enables you to specify lookup tables for these engine characteristics. The lookup tables are functions of engine load, $L$, and engine speed $N$. If you select **Input engine temperature**, the tables are also a function of engine temperature, $T$.

- Power
- Air
- Fuel
- Temperature
- Efficiency
- Emissions
  - Hydrocarbon (HC)
  - Carbon monoxide (CO)
  - Nitric oxide and nitrogen dioxide (NOx)
  - Carbon dioxide ($CO_2$)
  - Particulate matter (PM) emissions

To bound the Mapped Core Engine block output, the block does not extrapolate the lookup table data.

## Ports

### Input

#### *<TrqCmd>* — Engine load

`TrqCmd` (default)

Engine load, $L$. Examples of engine load include:

- Commanded torque
- Commanded indicated mean effective pressure (IMEP) in the engine cylinder
- Normalized cylinder air mass
- Injected fuel mass

**Dependencies**

To specify an engine load port name, on the **Configuration** tab, enter a name in the **Load input port name** parameter field.

### *<EngSpd>* — Engine speed

`EngSpd` (default)

Engine speed, $N$.

**Dependencies**

To specify an engine load port name, on the **Configuration** tab, enter a name in the **Speed input port name** parameter field.

### *<EngTemp>* — Engine temperature

`EngTemp` (default)

Engine temperature, $T$.

**Dependencies**

To create the engine temperature input port name, select **Input engine temperature** parameter field.

To specify an engine load port name, on the **Configuration** tab, enter a name in the **Temperature input port name** parameter field.

## Output

### *<EngTrq>* — Power

`EngTrq` (default)

Engine power, $T_{brake}$.

**Dependencies**

- To create this port, on the **Configuration** tab, select **Power**.
- To specify the port name, on the **Power** tab, enter a name in the **Power output port name** parameter field.

### *<IntkAirMassFlw>* — Air mass flow

`IntkAirMassFlw` (default)

Engine air mass flow, $\dot{m}_{intk}$.

**Dependencies**

- To create this port, on the **Configuration** tab, select **Air**.

- To specify the port name, on the **Air** tab, enter a name in the **Air output port name** parameter field.

#### *<FuelMassFlw>* — Fuel flow

`FuelMassFlw` (default)

Engine fuel flow, $\dot{m}_{fuel}$.

**Dependencies**

- To create this port, on the **Configuration** tab, select **Fuel**.
- To specify the port name, on the **Fuel** tab, enter a name in the **Fuel output port name** parameter field.

#### *<ExhManGasTemp>* — Exhaust temperature

`ExhManGasTemp` (default)

Engine exhaust temperature, $T_{exh}$.

**Dependencies**

- To create this port, on the **Configuration** tab, select **Temperature**.
- To specify the port name, on the **Temperature** tab, enter a name in the **Temperature output port name** parameter field.

#### *<Bsfc>* — Efficiency

`Bsfc` (default)

Brake-specific fuel consumption (BSFC), *Eff*.

**Dependencies**

- To create this port, on the **Configuration** tab, select **Efficiency**.
- To specify the port name, on the **Efficiency** tab, enter a name in the **Efficiency output port name** parameter field.

#### *<EoHC>* — Hydrocarbon emissions

`EoHC` (default)

Hydrocarbon emissions, *HC*.

**Dependencies**

- To create this port, on the **Configuration** tab, select **HC**.
- To specify the port name, on the **HC** tab, enter a name in the **HC output port name** parameter field.

#### *<EoCO>* — Carbon monoxide emissions

`EoCO` (default)

Carbon monoxide emissions, *CO*.

**Dependencies**

- To create this port, on the **Configuration** tab, select **CO**.

- To specify the port name, on the **CO** tab, enter a name in the **CO output port name** parameter field.

#### *<EoNOx>* — Nitric oxide and nitrogen dioxide emissions
`EoNOx` (default)

Nitric oxide and nitrogen dioxide emissions, *NOx*.

##### Dependencies

- To create this port, on the **Configuration** tab, select **NOx**.
- To specify the port name, on the **NOx** tab, enter a name in the **NOx output port name** parameter field.

#### *<EoCO2>* — Carbon dioxide emissions
`EoCO2` (default)

Carbon dioxide emissions, *CO2*.

##### Dependencies

- To create this port, on the **Configuration** tab, select **CO2**.
- To specify the port name, on the **CO2** tab, enter a name in the **CO2 output port name** parameter field.

#### *<EoPm>* — Particulate matter emissions
`EoPm` (default)

Particulate matter emissions, *PM*.

##### Dependencies

- To create this port, on the **Configuration** tab, select **PM**.
- To specify the port name, on the **PM** tab, enter a name in the **PM output port name** parameter field.

## Parameters

### Configuration

#### `Engine Type` — Type of engine image
`Compression-ignition (CI)` (default) | `Spark-ignition (SI)`

Type of mapped internal combustion engine image to use in the block.

#### `Load input port name` — Name
`TrqCmd` (default)

Engine load input port name.

#### `Breakpoints for load input` — Breakpoints
`vector`

Breakpoints for engine load input.

**`Speed input port name` — Name**
`EngSpd` (default)

Speed input port name.

**`Breakpoints for speed input` — Breakpoints**
`vector`

Breakpoints for engine speed input.

**`Temperature input port name` — Name**
`EngTemp` (default)

Temperature input port name.

**Dependencies**

To enable this parameter, select **Input engine temperature**.

**`Breakpoints for temperature input` — Breakpoints**
`[233.15 273.15 373.15]` (default) | `vector`

Breakpoints for engine temperature input.

**Dependencies**

To enable this parameter, select **Input engine temperature**.

**`Output Configuration` — Create output ports**
`on` (default)

Create the output ports.

**Dependencies**

The table summarizes the output ports that are created for each **Output** parameter selection.

| Output Selection | Creates Port | Creates Tab |
|---|---|---|
| **Power** | *EngTrq* | **Power** |
| **Air** | *IntkAirMassFlw* | **Air** |
| **Fuel** | *FuelMassFlw* | **Fuel** |
| **Temperature** | *ExhManGasTemp* | **Temperature** |
| **Efficiency** | *Bsfc* | **Efficiency** |
| **HC** | *EoHC* | **HC** |
| **CO** | *EoCO* | **CO** |
| **NOx** | *EoNOx* | **NOx** |
| **CO2** | *EoCO2* | **CO2** |
| **PM** | *EoPm* | **PM** |

**Power**

**`Power output port name` — Power**
`EngTrq` (default)

Power output port name.

#### Dependencies

To create this parameter, on the **Configuration** tab, select **Power**.

### `Power table` — Power
`array`

Power table.

#### Dependencies

To create this parameter, on the **Configuration** tab, select **Power**.

#### Air

### `Air output port name` — Air
`IntkAirMassFlw` (default)

Air mass flow output port name.

#### Dependencies

To create this parameter, on the **Configuration** tab, select **Air**.

### `Air table` — Air
`array`

Air mass flow table.

#### Dependencies

To create this parameter, on the **Configuration** tab, select **Air**.

#### Fuel

### `Fuel output port name` — Fuel
`FuelMassFlw` (default)

Fuel output port name.

#### Dependencies

To create this parameter, on the **Configuration** tab, select **Fuel**.

### `Fuel table` — Fuel
`array`

Fuel table.

#### Dependencies

To create this parameter, on the **Configuration** tab, select **Fuel**.

#### Temperature

### `Temperature output port name` — Temperature
`ExhManGasTemp` (default)

Temperature output port name.

**Dependencies**

To create this parameter, on the **Configuration** tab, select **Temperature**.

**`Temperature table` — Temperature**
`array`

Temperature table.

**Dependencies**

To create this parameter, on the **Configuration** tab, select **Temperature**.

**Efficiency**

**`Efficiency output port name` — Efficiency**
`Bsfc` (default)

Efficiency output port name.

**Dependencies**

To create this parameter, on the **Configuration** tab, select **Efficiency**.

**`Efficiency table` — Efficiency**
`array`

Efficiency table.

**Dependencies**

To create this parameter, on the **Configuration** tab, select **Efficiency**.

**HC**

**`HC output port name` — Hydrocarbon**
`EoHC` (default)

Hydrocarbon output port name.

**Dependencies**

To create this parameter, on the **Configuration** tab, select **HC**.

**`HC table` — Hydrocarbon**
`array`

Hydrocarbon table.

**Dependencies**

To create this parameter, on the **Configuration** tab, select **HC**.

**CO**

**`CO output port name` — Carbon dioxide**
`EoCO` (default)

Carbon monoxide output port name.

**Dependencies**

To create this parameter, on the **Configuration** tab, select **CO**.

**`CO table` — Carbon dioxide**
`array`

Carbon dioxide table.

**Dependencies**

To create this parameter, on the **Configuration** tab, select **CO**.

**NOx**

**`NOx output port name` — Nitric oxide *NO* and nitrogen dioxide $NO_2$**
`EoNOx` (default)

NOx output port name. NOx is nitric oxide *NO* and nitrogen dioxide $NO_2$.

**Dependencies**

To create this parameter, on the **Configuration** tab, select **NOx**.

**`NOx table` — Nitric oxide *NO* and nitrogen dioxide $NO_2$**
`array`

NOx emissions table. NOx is nitric oxide *NO* and nitrogen dioxide $NO_2$.

**Dependencies**

To create this parameter, on the **Configuration** tab, select **NOx**.

**CO2**

**`CO2 output port name` — Carbon dioxide**
`EoCO2` (default)

Carbon dioxide output port name.

**Dependencies**

To create this parameter, on the **Configuration** tab, select **CO2**.

**`CO2 table` — Carbon dioxide**
`array`

Carbon dioxide table.

**Dependencies**

To create this parameter, on the **Configuration** tab, select **CO2**.

**PM**

**`PM output port name` — Particulate matter**
`EoPm` (default)

Particulate matter output port name.

**Dependencies**

To create this parameter, on the **Configuration** tab, select **PM**.

**`PM table` — Particulate matter**
`array`

Particulate matter table.

**Dependencies**

To create this parameter, on the **Configuration** tab, select **PM**.

## Extended Capabilities

**C/C++ Code Generation**
Generate C and C++ code using Simulink® Coder™.

## See Also

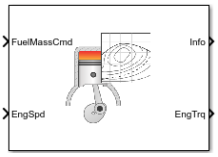
CI Core Engine | SI Core Engine

**Introduced in R2017a**

# Mapped CI Engine

Compression-ignition engine model using lookup tables

**Library:** Powertrain Blockset / Propulsion / Combustion Engines
Vehicle Dynamics Blockset / Powertrain / Propulsion




## Description

The Mapped CI Engine block implements a mapped compression-ignition (CI) engine model using power, air mass flow, fuel flow, exhaust temperature, efficiency, and emission performance lookup tables. You can use the block for:

- Hardware-in-the-loop (HIL) engine control design
- Vehicle-level fuel economy and performance simulations

The lookup tables, developed with the Model-Based Calibration Toolbox, are functions of injected fuel mass, $F$, engine torque, $T$, engine speed, $N$, and engine temperature, $Temp_{Eng}$.

| Input Command Setting | Input Engine Temperature Parameter Setting | Lookup Tables |
|---|---|---|
| `Fuel mass` | `off` | $f(F,N)$ |
| | `on` | $f(F,N,Temp_{Eng})$ |
| `Torque` | `off` | $f(T,N)$ |
| | `on` | $f(T,N,Temp_{Eng})$ |

The block enables you to specify lookup tables for these engine characteristics:

- Power
- Air
- Fuel
- Temperature
- Efficiency
- Hydrocarbon (HC) emissions
- Carbon monoxide (CO) emissions
- Nitric oxide and nitrogen dioxide (NOx) emissions
- Carbon dioxide ($CO_2$) emissions
- Particulate matter (PM) emissions

To bound the Mapped CI Engine block output, the block does not extrapolate the lookup table data.

### Virtual Calibration

If you have Model-Based Calibration Toolbox, click **Calibrate Maps** to virtually calibrate the 2D lookup tables using measured data. The dialog box steps through these tasks.

<table>
<tr><th>Task</th><th>Description</th></tr>
<tr><td>Import firing data</td><td>Import this loss data from a file. For example, open <code><matlabroot>/toolbox/autoblks/autodemos/projectsrc/CIDynamometer/CalMappedEng/CiEngineData.xlsx</code>.<br><br>For more information, see “Using Data” (Model-Based Calibration Toolbox).<br><br>
<table>
<tr><th>Input command</th><th>Required Data</th><th>Optional Data</th></tr>
<tr><td><code>Fuel mass</code></td><td>• Engine speed, rpm<br>• Commanded fuel mass per injection, mg<br>• Engine torque, N·m</td><td rowspan="2">• Air mass flow rate, kg/s<br>• Brake specific fuel consumption, g/(kW·h)<br>• CO2 mass flow rate, kg/s<br>• CO mass flow rate, kg/s<br>• Exhaust temperature, K<br>• Fuel mass flow rate, kg/s<br>• HC mass flow rate, kg/s<br>• NOx mass flow rate, kg/s<br>• Particulate matter mass flow rate, kg/s</td></tr>
<tr><td><code>Torque</code></td><td>• Engine speed, rpm<br>• Engine torque, N·m</td></tr>
</table>
<br>Collect firing data at steady-state operating conditions when injectors deliver the fuel. Data should cover the engine speed and torque operating range. Model-Based Calibration Toolbox uses the firing data boundary as the maximum torque.<br><br>To filter or edit the data, select Edit in Application. The Model-Based Calibration Toolbox Data Editor opens.</td></tr>
<tr><td>Import non-firing data</td><td>Import this non-firing data from a file. For example, open <code><matlabroot>/toolbox/autoblks/autodemos/projectsrc/CIDynamometer/CalMappedEng/CiEngineData.xlsx</code>.<br><br>• Engine speed, rpm<br>• Engine torque, N·m<br><br>Collect non-firing (motoring) data at steady-state operating conditions when fuel is cut off. All non-firing torque points must be less than zero. Non-firing data is a function of engine speed only.</td></tr>
</table>

| Task | Description |
|---|---|
| Generate response models | For both firing and non-firing data, the Model-Based Calibration Toolbox uses test plans to fit data to Gaussian process models (GPMs).<br><br>To assess or adjust the response model fit, select **Edit in Application**. The Model-Based Calibration Toolbox Model Browser opens. For more information, see “Model Assessment” (Model-Based Calibration Toolbox). |
| Generate calibration | Model-Based Calibration Toolbox calibrates the firing and non-firing response models and generates calibrated tables.<br><br>To assess or adjust the calibration, select **Edit in Application**. The Model-Based Calibration Toolbox CAGE Browser opens. For more information, see “Calibration Lookup Tables” (Model-Based Calibration Toolbox). |
| Update block parameters | Update the block lookup table and breakpoint parameters with the calibration. |

### Cylinder Air Mass

The block calculates the normalized cylinder air mass using these equations.

$$M_{Nom} = \frac{P_{std} V_d}{N_{cyl} R_{air} T_{std}}$$

$$L = \frac{\left(\frac{60s}{min}\right) Cps \cdot \dot{m}_{air}}{\left(\frac{1000g}{Kg}\right) N_{cyl} \cdot N \cdot M_{Nom}}$$

The equations use these variables.

| | |
|---|---|
| $L$ | Normalized cylinder air mass |
| $M_{Nom}$ | Nominal engine cylinder air mass at standard temperature and pressure, piston at bottom dead center (BDC) maximum volume, in kg |
| $Cps$ | Crankshaft revolutions per power stroke, rev/stroke |
| $P_{std}$ | Standard pressure |
| $T_{std}$ | Standard temperature |
| $R_{air}$ | Ideal gas constant for air and burned gas mixture |
| $V_d$ | Displaced volume |
| $N_{cyl}$ | Number of engine cylinders |
| $N$ | Engine speed |
| $\dot{m}_{intk}$ | Engine air mass flow, in g/s |

### Turbocharger Lag

To model turbocharger lag, select **Include turbocharger lag effect**. Turbocharger lag limits the maximum fuel mass per injection. To model the maximum fuel mass per injection, the block uses a first-order system with a time constant. At low torque, the engine does not require boost to provide sufficient air flow. When the requested fuel mass requires boost, the block uses a time constant to determine the maximum fuel mass per injection. The block uses these equations for the specified **Input command** setting.

| Calculation | Input command Parameter Setting | |
|---|---|---|
| | `Fuel mass` | `Torque` |
| Dynamic torque | $\frac{dF_{max}}{dt} = \frac{1}{\tau_{eng}}(F_{cmd} - F_{max})$ | $\frac{dT_{max}}{dt} = \frac{1}{\tau_{eng}}(T_{cmd} - T_{max})$ |
| Fuel mass per injection or torque - with turbocharger lag | $F = \begin{cases} F_{cmd} & \text{when } F_{cmd} < F_{\max} \\ F_{max} & \text{when } F_{cmd} \geq F_{\max} \end{cases}$ | $T_{target} = \begin{cases} T_{cmd} & \text{when } T_{cmd} < T_{\max} \\ T_{max} & \text{when } T_{cmd} \geq T_{\max} \end{cases}$ |
| Fuel mass per injection or torque- without turbocharger lag | $F = F_{cmd} = F_{max}$ | $T_{target} = T_{cmd} = T_{max}$ |
| Boost time constant | $\tau_{bst} = \begin{cases} \tau_{bst,rising} & \text{when } F_{cmd} > F_{\max} \\ \tau_{bst,falling} & \text{when } F_{cmd} \leq F_{\max} \end{cases}$ | $\tau_{bst} = \begin{cases} \tau_{bst,rising} & \text{when } T_{cmd} > T_{\max} \\ \tau_{bst,falling} & \text{when } T_{cmd} \leq T_{\max} \end{cases}$ |
| Final time constant | $\tau_{eng} = \begin{cases} \tau_{nat} & \text{when } T_{brake} < f_{bst}(N) \\ \tau_{bst} & \text{when } T_{brake} \geq f_{bst}(N) \end{cases}$ | |

The equations use these variables.

| | |
|---|---|
| $T_{brake}$ | Brake torque |
| $F$ | Fuel mass per injection |
| $F_{cmd}$, $F_{max}$ | Commanded and maximum fuel mass per injection, respectively |
| $T_{target}$, $T_{cmd}$, $T_{max}$ | Target, commanded, and maximum torque, respectively |
| $\tau_{bst}$ | Boost time constant |
| $\tau_{bst,rising}$, $\tau_{bst,falling}$ | Boost rising and falling time constant, respectively |
| $\tau_{eng}$ | Final time constant |
| $\tau_{nat}$ | Time constant below the boost torque speed line |
| $f_{bst}(N)$ | Boost torque/speed line |
| $N$ | Engine speed |

### Fuel Flow

To calculate the fuel economy for high-fidelity models, the block uses the volumetric fuel flow.

$$Q_{fuel} = \frac{\dot{m}_{fuel}}{\left(\frac{1000kg}{m^3}\right)Sg_{fuel}}$$

The equation uses these variables.

| | |
|---|---|
| $\dot{m}_{fuel}$ | Fuel mass flow |
| $Sg_{fuel}$ | Specific gravity of fuel |

$Q_{fuel}$ Volumetric fuel flow

### Power Accounting

For the power accounting, the block implements these equations.

| Bus Signal | | | Description | Equations |
|---|---|---|---|---|
| `PwrInfo` | `PwrTrnsfrd` — Power transferred between blocks<br>• Positive signals indicate flow into block<br>• Negative signals indicate flow out of block | `PwrCrkshft` | Crankshaft power | $-\tau_{eng}\omega$ |
| | `PwrNotTrnsfrd` — Power crossing the block boundary, but not transferred<br>• Positive signals indicate an input<br>• Negative signals indicate a loss | `PwrFuel` | Fuel input power | $\dot{m}_{fuel}LHV$ |
| | | `PwrLoss` | Power loss | $\tau_{eng}\omega - \dot{m}_{fuel}LHV$ |
| | `PwrStored` — Stored energy rate of change<br>• Positive signals indicate an increase<br>• Negative signals indicate a decrease | | *Not used* | |

The equations use these variables.

| | |
|---|---|
| $LHV$ | Fuel lower heating value |
| $\omega$ | Engine speed, rad/s |
| $\dot{m}_{fuel}$ | Fuel mass flow |
| $\tau_{eng}$ | Fuel mass per injection time constant |

## Ports

### Input

#### `FuelMassCmd` — Injected fuel mass command

`scalar`

Injected fuel mass command, $F$, in mg/inj.

##### Dependencies

To enable this port, for **Input command**, select `Fuel mass`.

#### `TrqCmd` — Torque command

`scalar`

Torque command, $T$, in N·m.

##### Dependencies

To enable this port, for **Input command**, select `Torque`.

### EngSpd — Engine speed

scalar

Engine speed, $N$, in rpm.

### EngTemp — Engine temperature

scalar

Engine temperature, $Temp_{Eng}$, in K.

#### Dependencies

To enable this port, select **Input engine temperature**.

## Output

### Info — Bus signal

bus

Bus signal containing these block calculations.

| Signal | Description | Units |
|---|---|---|
| IntkGasMassFlw | Engine air mass flow output | kg/s |
| NrmlzdAirChrg | Normalized engine cylinder air mass | N/A |
| Afr | Air-fuel ratio (AFR) | N/A |
| FuelMassFlw | Engine fuel flow output | kg/s |
| FuelVolFlw | Volumetric fuel flow | $m^3/s$ |
| ExhManGasTemp | Engine exhaust gas temperature | K |
| EngTrq | Engine torque output | N·m |
| EngSpd | Engine speed | rpm |
| CrkAng | Engine crankshaft absolute angle<br>$\int_0^{(360)Cps} EngSpd \frac{180}{30} d\theta$<br>where $Cps$ is crankshaft revolutions per power stroke. | degrees crank angle |
| Bsfc | Engine brake-specific fuel consumption (BSFC) | g/kWh |
| EoHC | Engine out hydrocarbon emission mass flow | kg/s |
| EoCO | Engine out carbon monoxide emission mass flow rate | kg/s |
| EoNOx | Engine out nitric oxide and nitrogen dioxide emissions mass flow | kg/s |
| EoCO2 | Engine out carbon dioxide emission mass flow | kg/s |

| Signal | | | Description | Units |
|---|---|---|---|---|
| EoPM | | | Engine out particulate matter emission mass flow | kg/s |
| PwrInfo | PwrTrnsfrd | PwrCrkshft | Crankshaft power | W |
| | PwrNotTrnsfrd | PwrFuel | Fuel input power | W |
| | | PwrLoss | Power loss | W |
| | PwrStored | | *Not used* | |

**EngTrq — Power**
scalar

Engine power, $T_{brake}$, in N·m.

## Parameters

### Block Options

**Input command — Table functions**
Fuel mass (default) | Torque

The lookup tables, developed with the Model-Based Calibration Toolbox, are functions of injected fuel mass, $F$, engine torque, $T$, engine speed, $N$, and engine temperature, $Temp_{Eng}$.

| Input Command Setting | Input Engine Temperature Parameter Setting | Lookup Tables |
|---|---|---|
| Fuel mass | off | $f(F,N)$ |
| | on | $f(F,N,Temp_{Eng})$ |
| Torque | off | $f(T,N)$ |
| | on | $f(T,N,Temp_{Eng})$ |

#### Dependencies

- Selecting Fuel mass enables **Breakpoints for commanded fuel mass input, f_tbrake_f_bpt**.
- Selecting Torque enables **Breakpoints for commanded torque input, f_tbrake_t_bpt**.
- Selecting **Input engine temperature** enables **Breakpoints for temperature input, f_tbrake_engtmp_bpt**.

**Include turbocharger lag effect — Increase time constant**
off (default)

To model turbocharger lag, select **Include turbocharger lag effect**. Turbocharger lag limits the maximum fuel mass per injection. To model the maximum fuel mass per injection, the block uses a first-order system with a time constant. At low torque, the engine does not require boost to provide sufficient air flow. When the requested fuel mass requires boost, the block uses a time constant to determine the maximum fuel mass per injection. The block uses these equations for the specified **Input command** setting.

| Calculation | Input command Parameter Setting | |
|---|---|---|
| | `Fuel mass` | `Torque` |
| Dynamic torque | $\frac{dF_{max}}{dt} = \frac{1}{\tau_{eng}}(F_{cmd} - F_{max})$ | $\frac{dT_{max}}{dt} = \frac{1}{\tau_{eng}}(T_{cmd} - T_{max})$ |
| Fuel mass per injection or torque - with turbocharger lag | $F = \begin{cases} F_{cmd} & \text{when } F_{cmd} < F_{\max} \\ F_{max} & \text{when } F_{cmd} \geq F_{\max} \end{cases}$ | $T_{target} = \begin{cases} T_{cmd} & \text{when } T_{cmd} < T_{\max} \\ T_{max} & \text{when } T_{cmd} \geq T_{\max} \end{cases}$ |
| Fuel mass per injection or torque- without turbocharger lag | $F = F_{cmd} = F_{max}$ | $T_{target} = T_{cmd} = T_{max}$ |
| Boost time constant | $\tau_{bst} = \begin{cases} \tau_{bst,rising} & \text{when } F_{cmd} > F_{\max} \\ \tau_{bst,falling} & \text{when } F_{cmd} \leq F_{\max} \end{cases}$ | $\tau_{bst} = \begin{cases} \tau_{bst,rising} & \text{when } T_{cmd} > T_{\max} \\ \tau_{bst,falling} & \text{when } T_{cmd} \leq T_{\max} \end{cases}$ |
| Final time constant | $\tau_{eng} = \begin{cases} \tau_{nat} & \text{when } T_{brake} < f_{bst}(N) \\ \tau_{bst} & \text{when } T_{brake} \geq f_{bst}(N) \end{cases}$ | |

The equations use these variables.

| | |
|---|---|
| $T_{brake}$ | Brake torque |
| $F$ | Fuel mass per injection |
| $F_{cmd}$, $F_{max}$ | Commanded and maximum fuel mass per injection, respectively |
| $T_{target}$, $T_{cmd}$, $T_{max}$ | Target, commanded, and maximum torque, respectively |
| $\tau_{bst}$ | Boost time constant |
| $\tau_{bst,rising}$, $\tau_{bst,falling}$ | Boost rising and falling time constant, respectively |
| $\tau_{eng}$ | Final time constant |
| $\tau_{nat}$ | Time constant below the boost torque speed line |
| $f_{bst}(N)$ | Boost torque/speed line |
| $N$ | Engine speed |

#### Dependencies

Selecting **Include turbocharger lag effect** enables these parameters:

- **Boost torque line, f_tbrake_bst**
- **Time constant below boost line, tau_nat**
- **Rising maximum fuel mass boost time constant, tau_bst_rising**
- **Falling maximum fuel mass boost time constant, tau_bst_falling**

**`Input engine temperature` — Create input port**
`off` (default) | `on`

Select this to create the `EngTemp` input port.

The lookup tables, developed with the Model-Based Calibration Toolbox, are functions of injected fuel mass, $F$, engine torque, $T$, engine speed, $N$, and engine temperature, $Temp_{Eng}$.

| Input Command Setting | Input Engine Temperature Parameter Setting | Lookup Tables |
|---|---|---|
| `Fuel mass` | `off` | $f(F,N)$ |
| | `on` | $f(F,N,Temp_{Eng})$ |
| `Torque` | `off` | $f(T,N)$ |
| | `on` | $f(T,N,Temp_{Eng})$ |

**Configuration**

**`Calibrate Maps` — Calibrate tables with measured data**
`selection`

If you have Model-Based Calibration Toolbox, click **Calibrate Maps** to virtually calibrate the 2D lookup tables using measured data. The dialog box steps through these tasks.

<table>
<tr><th>Task</th><th>Description</th></tr>
<tr><td>Import firing data</td><td>Import this loss data from a file. For example, open <code>matlabroot/toolbox/autoblks/autodemos/projectsrc/CIDynamometer/CalMappedEng/CiEngineData.xlsx</code>.<br><br>For more information, see “Using Data” (Model-Based Calibration Toolbox).<br><br>
<table>
<tr><th>Input command</th><th>Required Data</th><th>Optional Data</th></tr>
<tr><td><code>Fuel mass</code></td><td>• Engine speed, rpm<br>• Commanded fuel mass per injection, mg<br>• Engine torque, N·m</td><td rowspan="2">• Air mass flow rate, kg/s<br>• Brake specific fuel consumption, g/(kW·h)<br>• CO2 mass flow rate, kg/s<br>• CO mass flow rate, kg/s<br>• Exhaust temperature, K<br>• Fuel mass flow rate, kg/s<br>• HC mass flow rate, kg/s<br>• NOx mass flow rate, kg/s<br>• Particulate matter mass flow rate, kg/s</td></tr>
<tr><td><code>Torque</code></td><td>• Engine speed, rpm<br>• Engine torque, N·m</td></tr>
</table>
Collect firing data at steady-state operating conditions when injectors deliver the fuel. Data should cover the engine speed and torque operating range. Model-Based Calibration Toolbox uses the firing data boundary as the maximum torque.<br><br>To filter or edit the data, select Edit in Application. The Model-Based Calibration Toolbox Data Editor opens.</td></tr>
</table>

| Task | Description |
|---|---|
| Import non-firing data | Import this non-firing data from a file. For example, open `<matlabroot>/toolbox/autoblks/autodemos/projectsrc/CIDynamometer/CalMappedEng/CiEngineData.xlsx`.<br>• Engine speed, rpm<br>• Engine torque, N·m<br>Collect non-firing (motoring) data at steady-state operating conditions when fuel is cut off. All non-firing torque points must be less than zero. Non-firing data is a function of engine speed only. |
| Generate response models | For both firing and non-firing data, the Model-Based Calibration Toolbox uses test plans to fit data to Gaussian process models (GPMs).<br>To assess or adjust the response model fit, select **Edit in Application**. The Model-Based Calibration Toolbox Model Browser opens. For more information, see “Model Assessment” (Model-Based Calibration Toolbox). |
| Generate calibration | Model-Based Calibration Toolbox calibrates the firing and non-firing response models and generates calibrated tables.<br>To assess or adjust the calibration, select **Edit in Application**. The Model-Based Calibration Toolbox CAGE Browser opens. For more information, see “Calibration Lookup Tables” (Model-Based Calibration Toolbox). |
| Update block parameters | Update the block lookup table and breakpoint parameters with the calibration. |

#### Dependencies

To enable this parameter, clear **Input engine temperature**.

### `Breakpoints for commanded fuel mass input, f_tbrake_f_bpt` — Breakpoints

1-by-`M` vector

Breakpoints, in mg/inj.

#### Dependencies

Setting **Input command** to `Fuel mass` enables this parameter.

### `Breakpoints for commanded torque input, f_tbrake_t_bpt` — Breakpoints

1-by-`M` vector

Breakpoints, in N·m.

#### Dependencies

Setting **Input command** to `Torque` enables this parameter.

### `Breakpoints for engine speed input, f_tbrake_n_bpt` — Breakpoints

1-by-`N` vector

Breakpoints, in rpm.

**`Breakpoints for temperature input, f_tbrake_engtmp_bpt` — Breakpoints**
`[233.15 273.15 373.15]` (default) | 1-by-L vector

Breakpoints, in K.

**Dependencies**

To enable this parameter, select **Input engine temperature**.

**`Number of cylinders, NCyl` — Number**
4 (default) | `scalar`

Number of cylinders.

**`Crank revolutions per power stroke, Cps` — Crank revolutions**
2 (default) | `scalar`

Crank revolutions per power stroke.

**`Total displaced volume, Vd` — Volume**
`0.0015` (default) | `scalar`

Volume displaced by engine, in m^3.

**`Fuel lower heating value, Lhv` — Heating value**
`45e6` (default) | `scalar`

Fuel lower heating value, *LHV*, in J/kg.

**`Fuel specific gravity, Sg` — Specific gravity**
`0.832` (default) | `scalar`

Specific gravity of fuel, $Sg_{fuel}$, dimensionless.

**`Ideal gas constant air, Rair` — Constant**
287 (default) | `scalar`

Ideal gas constant of air and residual gas entering the engine intake port, in J/(kg·K).

**`Air standard pressure, Pstd` — Pressure**
`101325` (default) | `scalar`

Standard air pressure, in Pa.

**`Air standard temperature, Tstd` — Temperature**
`293.15` (default) | `scalar`

Standard air temperature, in K.

**`Boost torque line, f_tbrake_bst` — Boost lag**
`[90,95,95,95,96,100,104,104,104,100,95,85,75,67,60,55]` (default) | 1-by-M vector

Boost torque line, $f_{bst}(N)$, in N·m.

**Dependencies**

To enable this parameter, select **Include turbocharger lag effect**.

**`Time constant below boost line` — Time constant below**
`0.1` (default) | `scalar`

Time constant below boost line, $\tau_{nat}$, in s.

**Dependencies**

To enable this parameter, select **Include turbocharger lag effect**.

**`Rising maximum fuel mass boost time constant, tau_bst_rising` — Rising time constant**
`1.0` (default) | `scalar`

Rising maximum fuel mass boost time constant, $\tau_{bst,rising}$, in s.

**Dependencies**

To enable this parameter, select **Include turbocharger lag effect**.

**`Falling maximum fuel mass boost time constant, tau_bst_falling` — Falling time constant**
`0.7` (default) | `scalar`

Falling maximum fuel mass boost time constant, $\tau_{bst,falling}$, in s.

**Dependencies**

To enable this parameter, select **Include turbocharger lag effect**.

**`Turbocharger time constant blend fuel mass fraction, f_blend_frac` — Time constant**
`0.01` (default) | `scalar`

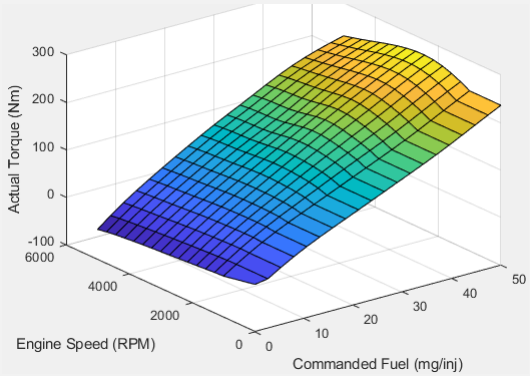
Turbocharger time constant blend fuel mass fraction, in s.

**Dependencies**

To enable this parameter, select **Include turbocharger lag effect**.

**Power**

**`Brake torque map, f_tbrake` — 2D lookup table**
`M`-by-`N` matrix

| Input Command Setting | Description |
|---|---|
| `Fuel mass` | The engine brake torque lookup table is a function of commanded fuel mass and engine speed, $T_{brake} = f(F, N)$, where:<br>• $T_{brake}$ is engine torque, in N·m.<br>• $F$ is commanded fuel mass, in mg per injection.<br>• $N$ is engine speed, in rpm.<br>  |
| `Torque` | The engine brake torque lookup table is a function of target torque and engine speed, $T_{brake} = f(T_{target}, N)$, where:<br>• $T_{brake}$ is engine torque, in N·m.<br>• $T_{target}$ is target torque, in N·m.<br>• $N$ is engine speed, in rpm. |

**Dependencies**

To enable this parameter, clear **Input engine temperature**.

### `Plot brake torque map` — Plot table

button

Click to plot table.

**Dependencies**

To enable this parameter, clear **Input engine temperature**.

### `Brake torque map, f_tbrake_3d` — 3D lookup table

M-by-N-by-L array

| Input Command Setting | Description |
|---|---|
| `Fuel mass` | The engine brake torque lookup table is a function of commanded fuel mass and engine speed, $T_{brake} = f(F, N, Temp_{Eng})$, where:<br>• $T_{brake}$ is engine torque, in N·m.<br>• $F$ is commanded fuel mass, in mg per injection.<br>• $Temp_{Eng}$ is engine temperature, in K. |

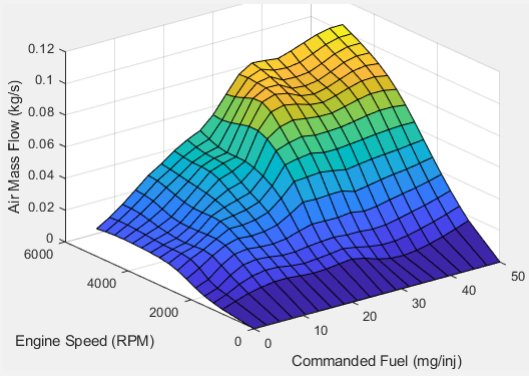
| Input Command Setting | Description |
|---|---|
| `Torque` | The engine brake torque lookup table is a function of target torque and engine speed, $T_{brake} = f(T_{target}, N, Temp_{Eng})$, where:<br>• $T_{brake}$ is engine torque, in N·m.<br>• $T_{target}$ is target torque, in N·m.<br>• $N$ is engine speed, in rpm.<br>• $Temp_{Eng}$ is engine temperature, in K. |

**Dependencies**

To enable this parameter, select **Input engine temperature**.

**Air**

**`Air mass flow map, f_air` — 2D lookup table**
M-by-N matrix

| Input Command Setting | Description |
|---|---|
| `Fuel mass` | The air mass flow lookup table is a function of commanded fuel mass and engine speed, $\dot{m}_{intk} = f(F_{max}, N)$, where:<br>• $\dot{m}_{intk}$ is engine air mass flow, in kg/s.<br>• $F_{max}$ is commanded fuel mass, in mg per injection.<br>• $N$ is engine speed, in rpm.<br>  |
| `Torque` | The air mass flow lookup table is a function of maximum torque and engine speed, $\dot{m}_{intk} = f(T_{max}, N)$, where:<br>• $\dot{m}_{intk}$ is engine air mass flow, in kg/s.<br>• $T_{max}$ is maximum torque, in N·m.<br>• $N$ is engine speed, in rpm. |

**Dependencies**

To enable this parameter, clear **Input engine temperature**.

**`Plot air mass map` — Plot table**
button

Click to plot table.

**Dependencies**

To enable this parameter, clear **Input engine temperature**.

### `Air mass flow map, f_air_3d` — 3D lookup table

M-by-N-by-L array

| Input Command Setting | Description |
|---|---|
| `Fuel mass` | The air mass flow lookup table is a function of commanded fuel mass and engine speed, $\dot{m}_{intk} = f(F_{max}, N, Temp_{Eng})$, where:<br>• $\dot{m}_{intk}$ is engine air mass flow, in kg/s.<br>• $F_{max}$ is commanded fuel mass, in mg per injection.<br>• $N$ is engine speed, in rpm.<br>• $Temp_{Eng}$ is engine temperature, in K. |
| `Torque` | The air mass flow lookup table is a function of maximum torque and engine speed, $\dot{m}_{intk} = f(T_{max}, N, Temp_{Eng})$, where:<br>• $\dot{m}_{intk}$ is engine air mass flow, in kg/s.<br>• $T_{max}$ is maximum torque, in N·m.<br>• $N$ is engine speed, in rpm.<br>• $Temp_{Eng}$ is engine temperature, in K. |

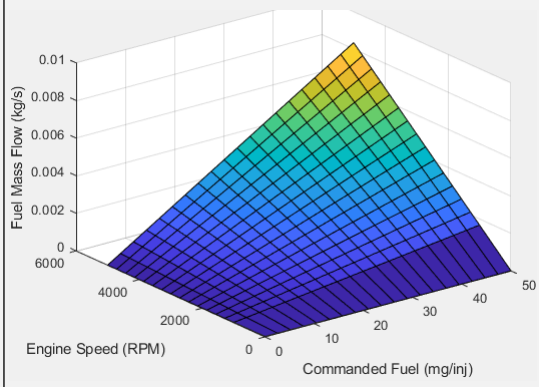
**Dependencies**

To enable this parameter, select **Input engine temperature**.

**Fuel**

### `Fuel flow map, f_fuel` — 2D lookup table

M-by-N matrix

| Input Command Setting | Description |
|---|---|
| `Fuel mass` | The engine fuel flow lookup table is a function of commanded fuel mass and engine speed, $MassFlow = f(F, N)$, where:<br>• $MassFlow$ is engine fuel mass flow, in kg/s.<br>• $F$ is commanded fuel mass, in mg per injection.<br>• $N$ is engine speed, in rpm.<br>  |
| `Torque` | The engine fuel flow lookup table is a function of target torque and engine speed, $MassFlow = f(T_{target}, N)$, where:<br>• $MassFlow$ is engine fuel mass flow, in kg/s.<br>• $T_{target}$ is target torque, in N·m.<br>• $N$ is engine speed, in rpm. |

**Dependencies**

To enable this parameter, clear **Input engine temperature**.

**`Plot fuel flow map` — Plot table**
button

Click to plot table.

**Dependencies**

To enable this parameter, clear **Input engine temperature**.

**`Fuel flow map, f_fuel_3d` — 3D lookup table**
M-by-N-by-L array

| Input Command Setting | Description |
|---|---|
| `Fuel mass` | The engine fuel flow lookup table is a function of commanded fuel mass, engine speed, and engine temperature, $MassFlow = f(F, N, Temp_{Eng})$, where:<br>• $MassFlow$ is engine fuel mass flow, in kg/s.<br>• $F$ is commanded fuel mass, in mg per injection.<br>• $N$ is engine speed, in rpm.<br>• $Temp_{Eng}$ is engine temperature, in K. |
| `Torque` | The engine fuel flow lookup table is a function of target torque and engine speed, and engine temperature, $MassFlow = f(T_{target}, N, Temp_{Eng})$, where:<br>• $MassFlow$ is engine fuel mass flow, in kg/s.<br>• $T_{target}$ is target torque, in N·m.<br>• $N$ is engine speed, in rpm.<br>• $Temp_{Eng}$ is engine temperature, in K. |

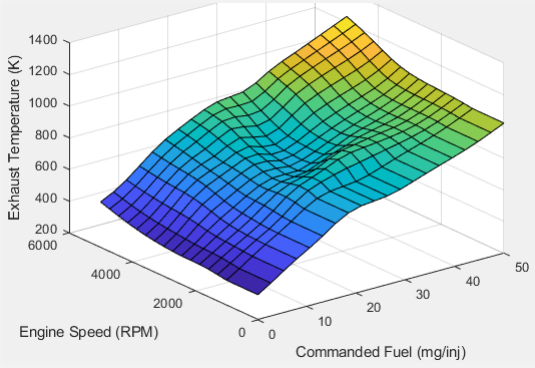
#### Dependencies

To enable this parameter, select **Input engine temperature**.

#### Temperature

### `Exhaust temperature map, f_texh` — 2D lookup table

M-by-N matrix

| Input Command Setting | Description |
|---|---|
| `Fuel mass` | The engine exhaust temperature table is a function of commanded fuel mass and engine speed, $T_{exh} = f(F, N)$, where:<br>• $T_{exh}$is exhaust temperature, in K.<br>• $F$ is commanded fuel mass, in mg per injection.<br>• $N$ is engine speed, in rpm.<br>  |

| Input Command Setting | Description |
|---|---|
| `Torque` | The engine exhaust temperature table is a function of target torque and engine speed, $T_{exh} = f(T_{target}, N)$, where:<br>• $T_{exh}$ is exhaust temperature, in K.<br>• $T_{target}$ is target torque, in N·m.<br>• $N$ is engine speed, in rpm. |

**Dependencies**

To enable this parameter, clear **Input engine temperature**.

**`Plot exhaust temperature map` — Plot table**
button

Click to plot table.

**Dependencies**

To enable this parameter, clear **Input engine temperature**.

**`Exhaust temperature map, f_texh_3d` — 3D lookup table**
M-by-N-by-L array

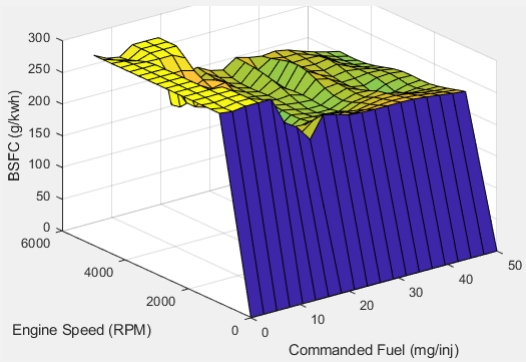
| Input Command Setting | Description |
|---|---|
| `Fuel mass` | The engine exhaust temperature table is a function of commanded fuel mass and engine speed, $T_{exh} = f(F, N, Temp_{Eng})$, where:<br>• $T_{exh}$is exhaust temperature, in K.<br>• $F$ is commanded fuel mass, in mg per injection.<br>• $N$ is engine speed, in rpm.<br>• $Temp_{Eng}$ is engine temperature, in K. |
| `Torque` | The engine exhaust temperature table is a function of target torque and engine speed, $T_{exh} = f(T_{target}, N, Temp_{Eng})$, where:<br>• $T_{exh}$ is exhaust temperature, in K.<br>• $T_{target}$ is target torque, in N·m.<br>• $N$ is engine speed, in rpm.<br>• $Temp_{Eng}$ is engine temperature, in K. |

**Dependencies**

To enable this parameter, select **Input engine temperature**.

**Efficiency**

**`BSFC map, f_eff` — 2D lookup table**
M-by-N matrix

| Input Command Setting | Description |
|---|---|
| `Fuel mass` | The brake-specific fuel consumption (BSFC) efficiency is a function of commanded fuel mass and engine speed, $BSFC = f(F, N)$, where:<br>• $BSFC$ is BSFC, in g/kWh.<br>• $F$ is commanded fuel mass, in mg per injection.<br>• $N$ is engine speed, in rpm.<br>  |
| `Torque` | The brake-specific fuel consumption (BSFC) efficiency is a function of target torque and engine speed, $BSFC = f(T_{target}, N)$, where:<br>• $BSFC$ is BSFC, in g/kWh.<br>• $T_{target}$ is target torque, in N·m.<br>• $N$ is engine speed, in rpm. |

**Dependencies**

To enable this parameter, clear **Input engine temperature**.

**`Plot BSFC map` — Plot table**
button

Click to plot table.

**Dependencies**

To enable this parameter, clear **Input engine temperature**.

**`BSFC map, f_eff_3d` — 3D lookup table**
M-by-N-by-L array

| Input Command Setting | Description |
|---|---|
| `Fuel mass` | The brake-specific fuel consumption (BSFC) efficiency is a function of commanded fuel mass and engine speed, $BSFC = f(F, N, Temp_{Eng})$, where:<br>• $BSFC$ is BSFC, in g/kWh.<br>• $F$ is commanded fuel mass, in mg per injection.<br>• $N$ is engine speed, in rpm.<br>• $Temp_{Eng}$ is engine temperature, in K. |

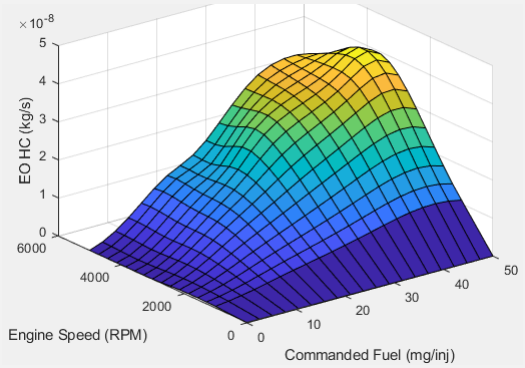
| Input Command Setting | Description |
|---|---|
| `Torque` | The brake-specific fuel consumption (BSFC) efficiency is a function of target torque and engine speed, $BSFC = f(T_{target}, N, Temp_{Eng})$, where:<br>• $BSFC$ is BSFC, in g/kWh.<br>• $T_{target}$ is target torque, in N·m.<br>• $N$ is engine speed, in rpm.<br>• $Temp_{Eng}$ is engine temperature, in K. |

**Dependencies**

To enable this parameter, select **Input engine temperature**.

**HC**

**`EO HC map, f_hc` — 2D lookup table**
M-by-N matrix

| Input Command Setting | Description |
|---|---|
| `Fuel mass` | The engine-out hydrocarbon emissions are a function of commanded fuel mass and engine speed, $EO\ HC = f(F, N)$, where:<br>• $EO\ HC$ is engine-out hydrocarbon emissions, in kg/s.<br>• $F$ is commanded fuel mass, in mg per injection.<br>• $N$ is engine speed, in rpm.<br>  |
| `Torque` | The engine-out hydrocarbon emissions are a function of target torque and engine speed, $EO\ HC = f(T_{target}, N)$, where:<br>• $EO\ HC$ is engine-out hydrocarbon emissions, in kg/s.<br>• $T_{target}$ is target torque, in N·m.<br>• $N$ is engine speed, in rpm. |

**Dependencies**

To enable this parameter, clear **Input engine temperature**.

**`Plot EO HC map` — Plot table**
button

Click to plot table.

**Dependencies**

To enable this parameter, clear **Input engine temperature**.

### `EO HC map, f_hc_3d` — 3D lookup table

M-by-N-by-L array

| Input Command Setting | Description |
|---|---|
| `Fuel mass` | The engine-out hydrocarbon emissions are a function of commanded fuel mass and engine speed, $EO\ HC = f(F, N, Temp_{Eng})$, where:<br>• $EO\ HC$ is engine-out hydrocarbon emissions, in kg/s.<br>• $F$ is commanded fuel mass, in mg per injection.<br>• $N$ is engine speed, in rpm.<br>• $Temp_{Eng}$ is engine temperature, in K. |
| `Torque` | The engine-out hydrocarbon emissions are a function of target torque and engine speed, $EO\ HC = f(T_{target}, N, Temp_{Eng})$, where:<br>• $EO\ HC$ is engine-out hydrocarbon emissions, in kg/s.<br>• $T_{target}$ is target torque, in N·m.<br>• $N$ is engine speed, in rpm.<br>• $Temp_{Eng}$ is engine temperature, in K. |

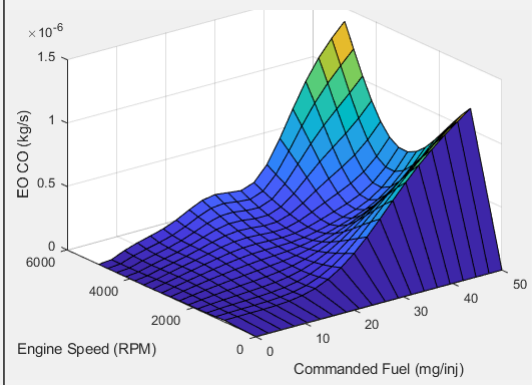
**Dependencies**

To enable this parameter, select **Input engine temperature**.

**CO**

### `EO CO map, f_co` — 2D lookup table

M-by-N matrix

| Input Command Setting | Description |
|---|---|
| `Fuel mass` | The engine-out carbon monoxide emissions are a function of commanded fuel mass and engine speed, $EO\ CO = f(F, N)$, where:<br>• $EO\ CO$ is engine-out carbon monoxide emissions, in kg/s.<br>• $F$ is commanded fuel mass, in mg per injection.<br>• $N$ is engine speed, in rpm.<br>  |
| `Torque` | The engine-out carbon monoxide emissions are a function of target torque and engine speed, $EO\ CO = f(T_{target}, N)$, where:<br>• $EO\ CO$ is engine-out carbon monoxide emissions, in kg/s.<br>• $T_{target}$ is target torque, in N·m.<br>• $N$ is engine speed, in rpm. |

**Dependencies**

To enable this parameter, clear **Input engine temperature**.

**`Plot EO CO map` — Plot table**
button

Click to plot table.

**Dependencies**

To enable this parameter, clear **Input engine temperature**.

**`EO CO map, f_co_3d` — 3D lookup table**
M-by-N-by-L array

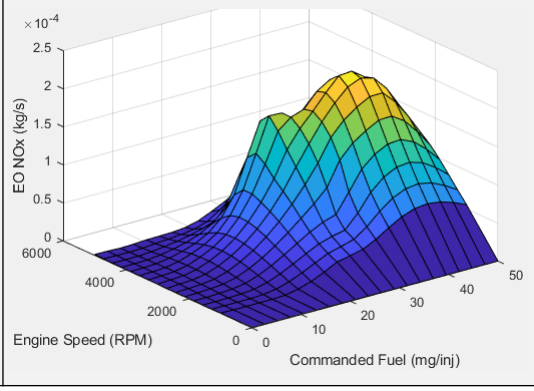
| Input Command Setting | Description |
|---|---|
| `Fuel mass` | The engine-out carbon monoxide emissions are a function of commanded fuel mass and engine speed, $EO\ CO = f(F, N, Temp_{Eng})$, where:<br>• $EO\ CO$ is engine-out carbon monoxide emissions, in kg/s.<br>• $F$ is commanded fuel mass, in mg per injection.<br>• $N$ is engine speed, in rpm.<br>• $Temp_{Eng}$ is engine temperature, in K. |
| `Torque` | The engine-out carbon monoxide emissions are a function of target torque and engine speed, $EO\ CO = f(T_{target}, N, Temp_{Eng})$, where:<br>• $EO\ CO$ is engine-out carbon monoxide emissions, in kg/s.<br>• $T_{target}$ is target torque, in N·m.<br>• $N$ is engine speed, in rpm.<br>• $Temp_{Eng}$ is engine temperature, in K. |

**Dependencies**

To enable this parameter, select **Input engine temperature**.

**NOx**

**`EO NOx map, f_nox` — 2D lookup table**
M-by-N matrix

| Input Command Setting | Description |
|---|---|
| `Fuel mass` | The engine-out nitric oxide and nitrogen dioxide emissions are a function of commanded fuel mass and engine speed, $EO\ NOx = f(F, N)$, where:<br>• $EO\ NOx$ is engine-out nitric oxide and nitrogen dioxide emissions, in kg/s.<br>• $F$ is commanded fuel mass, in mg per injection.<br>• $N$ is engine speed, in rpm.<br>  |

| Input Command Setting | Description |
|---|---|
| `Torque` | The engine-out nitric oxide and nitrogen dioxide emissions are a function of target torque and engine speed, $EO\ NOx = f(T_{target}, N)$, where:<br>• $EO\ NOx$ is engine-out nitric oxide and nitrogen dioxide emissions, in kg/s.<br>• $T_{target}$ is target torque, in N·m.<br>• $N$ is engine speed, in rpm. |

**Dependencies**

To enable this parameter, clear **Input engine temperature**.

**`Plot EO NOx map` — Plot table**
button

Click to plot table.

**Dependencies**

To enable this parameter, clear **Input engine temperature**.

**`EO NOx map, f_nox_3d` — 3D lookup table**
`M`-by-`N`-by-`L` array

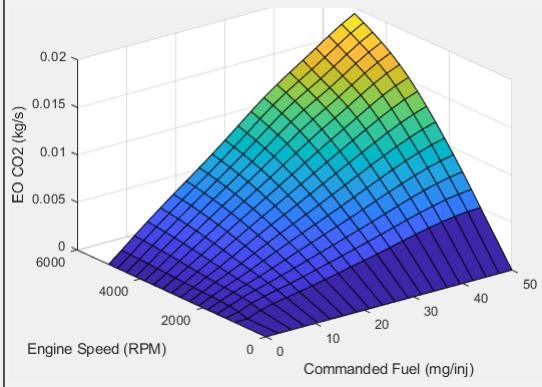
| Input Command Setting | Description |
|---|---|
| `Fuel mass` | The engine-out nitric oxide and nitrogen dioxide emissions are a function of commanded fuel mass, engine speed, and engine temperature, $EO\ NOx = f(F, N, Temp_{Eng})$, where:<br>• $EO\ NOx$ is engine-out nitric oxide and nitrogen dioxide emissions, in kg/s.<br>• $F$ is commanded fuel mass, in mg per injection.<br>• $N$ is engine speed, in rpm.<br>• $Temp_{Eng}$ is engine temperature, in K. |
| `Torque` | The engine-out nitric oxide and nitrogen dioxide emissions are a function of target torque, engine speed, and engine temperature, $EO\ NOx = f(T_{target}, N, Temp_{Eng})$, where:<br>• $EO\ NOx$ is engine-out nitric oxide and nitrogen dioxide emissions, in kg/s.<br>• $T_{target}$ is target torque, in N·m.<br>• $N$ is engine speed, in rpm.<br>• $Temp_{Eng}$ is engine temperature, in K. |

**Dependencies**

To enable this parameter, select **Input engine temperature**.

### CO2

**`EO CO2 map, f_co2` — 2D lookup table**
M-by-N matrix

| Input Command Setting | Description |
|---|---|
| `Fuel mass` | The engine-out carbon dioxide emissions are a function of commanded fuel mass and engine speed, $EO\ CO2 = f(F, N)$, where:<br>• $EO\ CO2$ is engine-out carbon dioxide emissions, in kg/s.<br>• $F$ is commanded fuel mass, in mg per injection.<br>• $N$ is engine speed, in rpm.<br>  |
| `Torque` | The engine-out carbon dioxide emissions are a function of target torque and engine speed, $EO\ CO2 = f(T_{target}, N)$, where:<br>• $EO\ CO2$ is engine-out carbon dioxide emissions, in kg/s.<br>• $T_{target}$ is target torque, in N·m.<br>• $N$ is engine speed, in rpm. |

#### Dependencies

To enable this parameter, clear **Input engine temperature**.

**`Plot CO2 map` — Plot table**
button

Click to plot table.

#### Dependencies

To enable this parameter, clear **Input engine temperature**.

**`EO CO2 map, f_co2_3d` — 3D lookup table**
M-by-N-by-L array

| Input Command Setting | Description |
|---|---|
| `Fuel mass` | The engine-out carbon dioxide emissions are a function of commanded fuel mass, engine speed, and engine temperature, $EO\ CO2 = f(F, N, Temp_{Eng})$, where:<br>• $EO\ CO2$ is engine-out carbon dioxide emissions, in kg/s.<br>• $F$ is commanded fuel mass, in mg per injection.<br>• $N$ is engine speed, in rpm.<br>• $Temp_{Eng}$ is engine temperature, in K. |
| `Torque` | The engine-out carbon dioxide emissions are a function of target torque, engine speed, and engine temperature, $EO\ CO2 = f(T_{target}, N, Temp_{Eng})$, where:<br>• $EO\ CO2$ is engine-out carbon dioxide emissions, in kg/s.<br>• $T_{target}$ is target torque, in N·m.<br>• $N$ is engine speed, in rpm.<br>• $Temp_{Eng}$ is engine temperature, in K. |

**Dependencies**

To enable this parameter, select **Input engine temperature**.

**PM**

**`EO PM map, f_pm` — 2D lookup table**
`M-by-N` matrix

| Input Command Setting | Description |
|---|---|
| `Fuel mass` | The engine-out PM emissions are a function of commanded fuel mass and engine speed, where:<br>• $EO\ PM$ is engine-out PM emissions, in kg/s.<br>• $F$ is commanded fuel mass, in mg per injection.<br>• $N$ is engine speed, in rpm. |
| `Torque` | The engine-out PM emissions are a function of target torque and engine speed, $EO\ PM = f(T_{target}, N)$, where:<br>• $EO\ PM$ is engine-out PM emissions, in kg/s.<br>• $T_{target}$ is target torque, in N·m.<br>• $N$ is engine speed, in rpm. |

**Dependencies**

To enable this parameter, clear **Input engine temperature**.

**`Plot EO PM map` — Plot table**
button

Click to plot table.

#### Dependencies

To enable this parameter, clear **Input engine temperature**.

### `EO PM map, f_pm_3d` — 3D lookup table

M-by-N-by-L array

| Input Command Setting | Description |
|---|---|
| `Fuel mass` | The engine-out PM emissions are a function of commanded fuel mass, engine speed, and engine temperature, where:<br>• *EO PM* is engine-out PM emissions, in kg/s.<br>• *F* is commanded fuel mass, in mg per injection.<br>• *N* is engine speed, in rpm.<br>• $Temp_{Eng}$ is engine temperature, in K. |
| `Torque` | The engine-out PM emissions are a function of target torque, engine speed, and engine temperature, $EO\ PM = f(T_{target}, N, T)$, where:<br>• *EO PM* is engine-out PM emissions, in kg/s.<br>• $T_{target}$ is target torque, in N·m.<br>• *N* is engine speed, in rpm.<br>• $Temp_{Eng}$ is engine temperature, in K. |

#### Dependencies

To enable this parameter, select **Input engine temperature**.

## Extended Capabilities

### C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

## See Also

CI Core Engine | Mapped Motor | Mapped SI Engine

### Topics

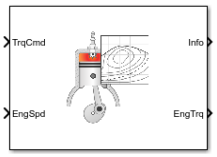
“Generate Mapped CI Engine from a Spreadsheet”
“Engine Calibration Maps”
“Model-Based Calibration Toolbox”

### Introduced in R2017a

# Mapped SI Engine

Spark-ignition engine model using lookup tables

**Library:** Powertrain Blockset / Propulsion / Combustion Engines
Vehicle Dynamics Blockset / Powertrain / Propulsion




## Description

The Mapped SI Engine block implements a mapped spark-ignition (SI) engine model using power, air mass flow, fuel flow, exhaust temperature, efficiency, and emission performance lookup tables. You can use the block for:

- Hardware-in-the-loop (HIL) engine control design
- Vehicle-level fuel economy and performance simulations

The block enables you to specify lookup tables for these engine characteristics. The lookup tables, developed with the Model-Based Calibration Toolbox, are functions of commanded torque, $T_{cmd}$, brake torque, $T_{brake}$, and engine speed, $N$. If you select **Input engine temperature**, the tables are also a function of engine temperature, $Temp_{Eng}$.

| Table | Input Engine Temperature Parameter Setting | |
|---|---|---|
| | `off` | `on` |
| Power | $f(T_{cmd},N)$ | $f(T_{cmd},N,Temp_{Eng})$ |
| Air | $f(T_{brake},N)$ | $f(T_{brake},N,Temp_{Eng})$ |
| Fuel | | |
| Temperature | | |
| Efficiency | | |
| HC | | |
| CO | | |
| NOx | | |
| CO2 | | |
| PM | | |

To bound the Mapped SI Engine block output, the block does not extrapolate the lookup table data.

### Virtual Calibration

If you have Model-Based Calibration Toolbox, click **Calibrate Maps** to virtually calibrate the 2D lookup tables using measured data. The dialog box steps through these tasks.

<table>
<tr><th>Task</th><th>Description</th></tr>
<tr><td>Import firing data</td><td>Import this loss data from a file. For example, open <code><matlabroot>/toolbox/autoblks/autodemos/projectsrc/SIDynamometer/CalMappedEng/SiEngineData.xlsx</code>.<br><br>For more information, see “Using Data” (Model-Based Calibration Toolbox).<br><br>
<table>
<tr><th>Required Data</th><th>Optional Data</th></tr>
<tr><td>• Engine speed, rpm<br>• Engine torque, N·m</td><td>• Air mass flow rate, kg/s<br>• Brake specific fuel consumption, g/(kW·h)<br>• CO2 mass flow rate, kg/s<br>• CO mass flow rate, kg/s<br>• Exhaust temperature, K<br>• Fuel mass flow rate, kg/s<br>• HC mass flow rate, kg/s<br>• NOx mass flow rate, kg/s<br>• Particulate matter mass flow rate, kg/s</td></tr>
</table>
<br>Collect firing data at steady-state operating conditions when injectors deliver the fuel. Data should cover the engine speed and torque operating range. Model-Based Calibration Toolbox uses the firing data boundary as the maximum torque.<br><br>To filter or edit the data, select Edit in Application. The Model-Based Calibration Toolbox Data Editor opens.</td></tr>
<tr><td>Import non-firing data</td><td>Import this non-firing data from a file. For example, open <code><matlabroot>/toolbox/autoblks/autodemos/projectsrc/SIDynamometer/CalMappedEng/SiEngineData.xlsx</code>.<br><br>• Engine speed, rpm<br>• Engine torque, N·m<br><br>Collect non-firing (motoring) data at steady-state operating conditions when fuel is cut off. All non-firing torque points must be less than zero. Non-firing data is a function of engine speed only.</td></tr>
<tr><td>Generate response models</td><td>For both firing and non-firing data, the Model-Based Calibration Toolbox uses test plans to fit data to Gaussian process models (GPMs).<br><br>To assess or adjust the response model fit, select Edit in Application. The Model-Based Calibration Toolbox Model Browser opens. For more information, see “Model Assessment” (Model-Based Calibration Toolbox).</td></tr>
<tr><td>Generate calibration</td><td>Model-Based Calibration Toolbox calibrates the firing and non-firing response models and generates calibrated tables.<br><br>To assess or adjust the calibration, select Edit in Application. The Model-Based Calibration Toolbox CAGE Browser opens. For more information, see “Calibration Lookup Tables” (Model-Based Calibration Toolbox).</td></tr>
</table>

| Task | Description |
|---|---|
| Update block parameters | Update the block lookup table and breakpoint parameters with the calibration. |

### Cylinder Air Mass

The block calculates the normalized cylinder air mass using these equations.

$$M_{Nom} = \frac{P_{std} V_d}{N_{cyl} R_{air} T_{std}}$$

$$L = \frac{\left(\frac{60s}{min}\right) Cps \cdot \dot{m}_{air}}{\left(\frac{1000g}{Kg}\right) N_{cyl} \cdot N \cdot M_{Nom}}$$

The equations use these variables.

| | |
|---|---|
| $L$ | Normalized cylinder air mass |
| $M_{Nom}$ | Nominal engine cylinder air mass at standard temperature and pressure, piston at bottom dead center (BDC) maximum volume, in kg |
| $Cps$ | Crankshaft revolutions per power stroke, rev/stroke |
| $P_{std}$ | Standard pressure |
| $T_{std}$ | Standard temperature |
| $R_{air}$ | Ideal gas constant for air and burned gas mixture |
| $V_d$ | Displaced volume |
| $N_{cyl}$ | Number of engine cylinders |
| $N$ | Engine speed |
| $\dot{m}_{intk}$ | Engine air mass flow, in g/s |

### Turbocharger Lag

To model turbocharger lag, select **Include turbocharger lag effect**. During throttle control, the time constant models the manifold filling and emptying dynamics. When the torque request requires a turbocharger boost, the block uses a larger time constant to represent the turbocharger lag. The block uses these equations.

| | |
|---|---|
| Dynamic torque | $\frac{dT_{brake}}{dt} = \frac{1}{\tau_{eng}}(T_{stdy} - T_{brake})$ |
| Boost time constant | $\tau_{bst} = \begin{cases} \tau_{bst,\,rising} & \text{when } T_{stdy} > T_{brake} \\ \tau_{bst,\,falling} & \text{when } T_{stdy} \le T_{brake} \end{cases}$ |
| Final time constant | $\tau_{eng} = \begin{cases} \tau_{thr} & \text{when } T_{brake} < f_{bst}(N) \\ \tau_{bst} & \text{when } T_{brake} \ge f_{bst}(N) \end{cases}$ |

The equations use these variables.

| | |
|---|---|
| $T_{brake}$ | Brake torque |

| | |
|---|---|
| $T_{stdy}$ | Steady-state target torque |
| $\tau_{bst}$ | Boost time constant |
| $\tau_{bst,rising}$, $\tau_{bst,falling}$ | Boost rising and falling time constant, respectively |
| $\tau_{eng}$ | Final time constant |
| $\tau_{thr}$ | Time constant during throttle control |
| $f_{bst}(N)$ | Boost torque speed line |
| $N$ | Engine speed |

### Fuel Flow

To calculate the fuel economy for high-fidelity models, the block uses the volumetric fuel flow.

$$Q_{fuel} = \frac{\dot{m}_{fuel}}{\left(\frac{1000kg}{m^3}\right)Sg_{fuel}}$$

The equation uses these variables.

| | |
|---|---|
| $\dot{m}_{fuel}$ | Fuel mass flow |
| $Sg_{fuel}$ | Specific gravity of fuel |
| $Q_{fuel}$ | Volumetric fuel flow |

### Power Accounting

For the power accounting, the block implements these equations.

| Bus Signal | | | Description | Equations |
|---|---|---|---|---|
| `PwrInfo` | `PwrTrnsfrd` — Power transferred between blocks<br>• Positive signals indicate flow into block<br>• Negative signals indicate flow out of block | `PwrCrkshft` | Crankshaft power | $-\tau_{eng}\omega$ |
| | `PwrNotTrnsfrd` — Power crossing the block boundary, but not transferred<br>• Positive signals indicate an input<br>• Negative signals indicate a loss | `PwrFuel` | Fuel input power | $\dot{m}_{fuel}LHV$ |
| | | `PwrLoss` | Power loss | $\tau_{eng}\omega - \dot{m}_{fuel}LHV$ |
| | `PwrStored` — Stored energy rate of change<br>• Positive signals indicate an increase<br>• Negative signals indicate a decrease | | *Not used* | |

The equations use these variables.

| | |
|---|---|
| $LHV$ | Fuel lower heating value |

| | |
|---|---|
| $\omega$ | Engine speed, rad/s |
| $\dot{m}_{fuel}$ | Fuel mass flow |
| $\tau_{eng}$ | Fuel mass per injection time constant |

## Ports

### Input

#### `TrqCmd` — Commanded torque

`scalar`

Torque, $T_{cmd}$, in N·m.

#### `EngSpd` — Engine speed

`scalar`

Engine speed, $N$, in rpm.

#### `EngTemp` — Engine temperature

`scalar`

Engine temperature, $Temp_{Eng}$, in K.

##### Dependencies

To enable this port, select **Input engine temperature**.

### Output

#### `Info` — Bus signal

bus

Bus signal containing these block calculations.

| Signal | Description | Units |
|---|---|---|
| `IntkGassMassFlw` | Engine air mass flow output | kg/s |
| `NrmlzdAirChrg` | Normalized engine cylinder air mass | N/A |
| `Afr` | Air-fuel ratio (AFR) | N/A |
| `FuelMassFlw` | Engine fuel flow output | kg/s |
| `FuelVolFlw` | Volumetric fuel flow | $m^3/s$ |
| `ExhManGasTemp` | Engine exhaust gas temperature | K |
| `EngTrq` | Engine torque output | N·m |
| `EngSpd` | Engine speed | rpm |

| Signal | | | Description | Units |
|---|---|---|---|---|
| `CrkAng` | | | Engine crankshaft absolute angle<br>$\int_0^{(360)Cps} EngSpd\frac{180}{30}d\theta$<br>where *Cps* is crankshaft revolutions per power stroke. | degrees crank angle |
| `Bsfc` | | | Engine brake-specific fuel consumption (BSFC) | g/kWh |
| `EoHC` | | | Engine out hydrocarbon emission mass flow | kg/s |
| `EoCO` | | | Engine out carbon monoxide emission mass flow rate | kg/s |
| `EoNOx` | | | Engine out nitric oxide and nitrogen dioxide emissions mass flow | kg/s |
| `EoCO2` | | | Engine out carbon dioxide emission mass flow | kg/s |
| `EoPM` | | | Engine out particulate matter emission mass flow | kg/s |
| `PwrInfo` | `PwrTrnsfrd` | `PwrCrkshft` | Crankshaft power | W |
| | `PwrNotTrnsfrd` | `PwrFuel` | Fuel input power | W |
| | | `PwrLoss` | Power loss | W |
| | `PwrStored` | | *Not used* | |

#### `EngTrq` — Engine brake torque

`scalar`

Engine brake torque, $T_{brake}$, in N·m.

## Parameters

#### Block Options

#### `Include turbocharger lag effect` — Increase time constant

`off` (default)

To model turbocharger lag, select **Include turbocharger lag effect**. During throttle control, the time constant models the manifold filling and emptying dynamics. When the torque request requires a turbocharger boost, the block uses a larger time constant to represent the turbocharger lag. The block uses these equations.

| | |
|---|---|
| Dynamic torque | $\frac{dT_{brake}}{dt} = \frac{1}{\tau_{eng}}(T_{stdy} - T_{brake})$ |

| | |
|---|---|
| Boost time constant | $\tau_{bst} = \begin{cases} \tau_{bst,\,rising} & \text{when } T_{stdy} > T_{brake} \\ \tau_{bst,\,falling} & \text{when } T_{stdy} \le T_{brake} \end{cases}$ |
| Final time constant | $\tau_{eng} = \begin{cases} \tau_{thr} & \text{when } T_{brake} < f_{bst}(N) \\ \tau_{bst} & \text{when } T_{brake} \ge f_{bst}(N) \end{cases}$ |

The equations use these variables.

| | |
|---|---|
| $T_{brake}$ | Brake torque |
| $T_{stdy}$ | Steady-state target torque |
| $\tau_{bst}$ | Boost time constant |
| $\tau_{bst,rising}$, $\tau_{bst,falling}$ | Boost rising and falling time constant, respectively |
| $\tau_{eng}$ | Final time constant |
| $\tau_{thr}$ | Time constant during throttle control |
| $f_{bst}(N)$ | Boost torque speed line |
| $N$ | Engine speed |

#### Dependencies

Selecting **Include turbocharger lag effect** enables these parameters:

- **Boost torque line, f_tbrake_bst**
- **Time constant below boost line, tau_thr**
- **Rising torque boost time constant, tau_bst_rising**
- **Falling torque boost time constant, tau_bst_falling**

### `Input engine temperature` — Create input port

`off` (default) | `on`

Select this to create the `EngTemp` input port.

The block enables you to specify lookup tables for these engine characteristics. The lookup tables, developed with the Model-Based Calibration Toolbox, are functions of commanded torque, $T_{cmd}$, brake torque, $T_{brake}$, and engine speed, $N$. If you select **Input engine temperature**, the tables are also a function of engine temperature, $Temp_{Eng}$.

| Table | Input Engine Temperature Parameter Setting | |
|---|---|---|
| | `off` | `on` |
| Power | $f(T_{cmd},N)$ | $f(T_{cmd},N,Temp_{Eng})$ |
| Air | $f(T_{brake},N)$ | $f(T_{brake},N,Temp_{Eng})$ |
| Fuel | | |
| Temperature | | |
| Efficiency | | |
| HC | | |
| CO | | |

| Table | Input Engine Temperature Parameter Setting | |
|---|---|---|
| | `off` | `on` |
| NOx | | |
| CO2 | | |
| PM | | |

**Configuration**

### `Calibrate Maps` — Calibrate tables with measured data

`selection`

If you have Model-Based Calibration Toolbox, click **Calibrate Maps** to virtually calibrate the 2D lookup tables using measured data. The dialog box steps through these tasks.

| Task | Description |
|---|---|
| Import firing data | Import this loss data from a file. For example, open `<matlabroot>/toolbox/autoblks/autodemos/projectsrc/SIDynamometer/CalMappedEng/SiEngineData.xlsx`.<br><br>For more information, see “Using Data” (Model-Based Calibration Toolbox).<br><br>**Required Data**<br>• Engine speed, rpm<br>• Engine torque, N·m<br><br>**Optional Data**<br>• Air mass flow rate, kg/s<br>• Brake specific fuel consumption, g/(kW·h)<br>• CO2 mass flow rate, kg/s<br>• CO mass flow rate, kg/s<br>• Exhaust temperature, K<br>• Fuel mass flow rate, kg/s<br>• HC mass flow rate, kg/s<br>• NOx mass flow rate, kg/s<br>• Particulate matter mass flow rate, kg/s<br><br>Collect firing data at steady-state operating conditions when injectors deliver the fuel. Data should cover the engine speed and torque operating range. Model-Based Calibration Toolbox uses the firing data boundary as the maximum torque.<br><br>To filter or edit the data, select **Edit in Application**. The Model-Based Calibration Toolbox Data Editor opens. |

| Task | Description |
|---|---|
| Import non-firing data | Import this non-firing data from a file. For example, open `<matlabroot>/toolbox/autoblks/autodemos/projectsrc/SIDynamometer/CalMappedEng/SiEngineData.xlsx`.<br>• Engine speed, rpm<br>• Engine torque, N·m<br>Collect non-firing (motoring) data at steady-state operating conditions when fuel is cut off. All non-firing torque points must be less than zero. Non-firing data is a function of engine speed only. |
| Generate response models | For both firing and non-firing data, the Model-Based Calibration Toolbox uses test plans to fit data to Gaussian process models (GPMs).<br>To assess or adjust the response model fit, select **Edit in Application**. The Model-Based Calibration Toolbox Model Browser opens. For more information, see “Model Assessment” (Model-Based Calibration Toolbox). |
| Generate calibration | Model-Based Calibration Toolbox calibrates the firing and non-firing response models and generates calibrated tables.<br>To assess or adjust the calibration, select **Edit in Application**. The Model-Based Calibration Toolbox CAGE Browser opens. For more information, see “Calibration Lookup Tables” (Model-Based Calibration Toolbox). |
| Update block parameters | Update the block lookup table and breakpoint parameters with the calibration. |

**Dependencies**

To enable this parameter, clear **Input engine temperature**.

**`Breakpoints for commanded torque, f_tbrake_t_bpt` — Breakpoints**
1-by-`M` vector

Breakpoints, in N·m.

**`Breakpoints for engine speed input, f_tbrake_n_bpt` — Breakpoints**
1-by-`N` vector

Breakpoints, in rpm.

**`Breakpoints for temperature input, f_tbrake_engtmp_bpt` — Breakpoints**
`[233.15 273.15 373.15]` (default) | 1-by-`L` vector

Breakpoints, in K.

**Dependencies**

To enable this parameter, select **Input engine temperature**.

**`Number of cylinders, NCyl` — Number**
4 (default) | `scalar`

Number of cylinders.

**`Crank revolutions per power stroke, Cps` — Crank revolutions**
`2` (default) | `scalar`

Crank revolutions per power stroke.

**`Total displaced volume, Vd` — Volume**
`0.0015` (default) | `scalar`

Volume displaced by engine, in m^3.

**`Fuel lower heating value, Lhv` — Heating value**
`45e6` (default) | `scalar`

Fuel lower heating value, *LHV*, in J/kg.

**`Fuel specific gravity, Sg` — Specific gravity**
`0.745` (default) | `scalar`

Specific gravity of fuel, $Sg_{fuel}$, dimensionless.

**`Ideal gas constant air, Rair` — Constant**
`287` (default) | `scalar`

Ideal gas constant of air and residual gas entering the engine intake port, in J/(kg*K).

**`Air standard pressure, Pstd` — Pressure**
`101325` (default) | `scalar`

Standard air pressure, in Pa.

**`Air standard temperature, Tstd` — Temperature**
`293.15` (default) | `scalar`

Standard air temperature, in K.

**`Boost torque line, f_tbrake_bst` — Boost lag**
1-by-`M` vector

Boost torque line, $f_{bst}(N)$, in N·m.

**Dependencies**

To enable this parameter, select **Include turbocharger lag effect**.

**`Time constant below boost line` — Time constant below**
`0.2` (default) | `scalar`

Time constant below boost line, $\tau_{thr}$, in s.

**Dependencies**

To enable this parameter, select **Include turbocharger lag effect**.

**`Rising torque boost time constant, tau_bst_rising` — Rising time constant**
`1.5` (default) | `scalar`

Rising torque boost time constant, $\tau_{bst,rising}$, in s.

**Dependencies**

To enable this parameter, select **Include turbocharger lag effect**.

### `Falling torque boost time constant, tau_bst_falling` — Falling time constant

1 (default) | `scalar`

Falling torque boost time constant, $\tau_{bst,falling}$, in s.

**Dependencies**

To enable this parameter, select **Include turbocharger lag effect**.

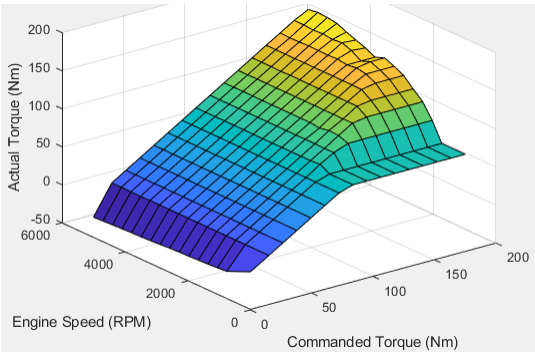
**Power**

### `Brake torque map, f_tbrake` — 2D lookup table

M-by-N matrix

The engine torque lookup table is a function of commanded engine torque and engine speed, $T = f(T_{cmd}, N)$, where:

- $T$ is engine torque, in N·m.
- $T_{cmd}$ is commanded engine torque, in N·m.
- $N$ is engine speed, in rpm.




### `Plot brake torque map` — Plot table

button

Click to plot table.

**Dependencies**

To enable this parameter, clear **Input engine temperature**.

### `Brake torque map, f_tbrake_3d` — 3D lookup table

M-by-N-by-L array

The engine torque lookup table is a function of commanded engine torque, engine speed, and engine temperature, $T = f(T_{cmd}, N, Temp_{Eng})$, where:

- $T$ is engine torque, in N·m.
- $T_{cmd}$ is commanded engine torque, in N·m.

- $N$ is engine speed, in rpm.
- $Temp_{Eng}$ is engine temperature, in K.

**Dependencies**

To enable this parameter, select **Input engine temperature**.

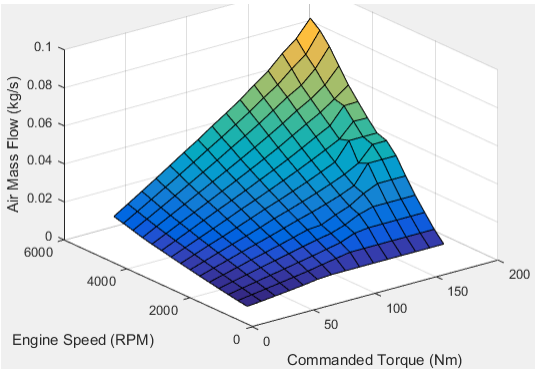
**Air**

### `Air mass flow map, f_air` — 2D lookup table

M-by-N matrix

The engine air mass flow lookup table is a function of commanded engine torque and engine speed, $\dot{m}_{intk} = f(T_{cmd}, N)$, where:

- $\dot{m}_{intk}$ is engine air mass flow, in kg/s.
- $T_{cmd}$ is commanded engine torque, in N·m.
- $N$ is engine speed, in rpm.




**Dependencies**

To enable this parameter, clear **Input engine temperature**.

### `Plot air mass map` — Plot table

button

Click to plot table.

**Dependencies**

To enable this parameter, clear **Input engine temperature**.

### `Air mass flow map, f_air_3d` — 3D lookup table

M-by-N-by-L array

The engine air mass flow lookup table is a function of commanded engine torque, engine speed, and engine temperature, $\dot{m}_{intk} = f(T_{cmd}, N, Temp_{Eng})$, where:

- $\dot{m}_{intk}$ is engine air mass flow, in kg/s.
- $T_{cmd}$ is commanded engine torque, in N·m.

- $N$ is engine speed, in rpm.
- $Temp_{Eng}$ is engine temperature, in K.

**Dependencies**

To enable this parameter, select **Input engine temperature**.

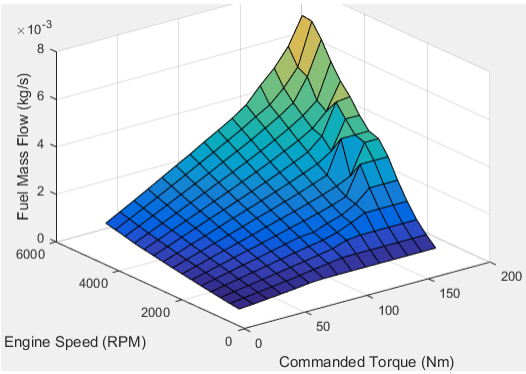
**Fuel**

### `Fuel flow map, f_fuel` — 2D lookup table

M-by-N matrix

The engine fuel mass flow lookup table is a function of commanded engine torque and engine speed, $MassFlow = f(T_{cmd}, N)$, where:

- $MassFlow$ is engine fuel mass flow, in kg/s.
- $T_{cmd}$ is commanded engine torque, in N·m.
- $N$ is engine speed, in rpm.




**Dependencies**

To enable this parameter, clear **Input engine temperature**.

### `Plot fuel flow map` — Plot table

button

Click to plot table.

**Dependencies**

To enable this parameter, clear **Input engine temperature**.

### `Fuel flow map, f_fuel_3d` — 3D lookup table

M-by-N-by-L array

The engine fuel mass flow lookup table is a function of commanded engine torque, engine speed, and engine temperature, $MassFlow = f(T_{cmd}, N, Temp_{Eng})$, where:

- $MassFlow$ is engine fuel mass flow, in kg/s.
- $T_{cmd}$ is commanded engine torque, in N·m.
- $N$ is engine speed, in rpm.

- $Temp_{Eng}$ is engine temperature, in K.

**Dependencies**

To enable this parameter, select **Input engine temperature**.

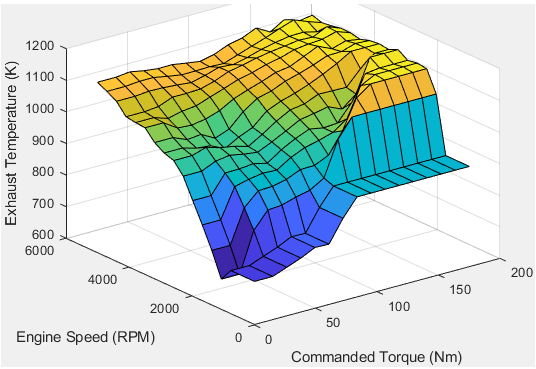
**Temperature**

### `Exhaust temperature map, f_texh` — 2D lookup table

`M-by-N` matrix

The engine exhaust temperature lookup table is a function of commanded engine torque and engine speed, $T_{exh} = f(T_{cmd}, N)$, where:

- $T_{exh}$ is exhaust temperature, in K.
- $T_{cmd}$ is commanded engine torque, in N·m.
- $N$ is engine speed, in rpm.




**Dependencies**

To enable this parameter, clear **Input engine temperature**.

### `Plot exhaust temperature map` — Plot table

button

Click to plot table.

**Dependencies**

To enable this parameter, clear **Input engine temperature**.

### `Exhaust temperature map, f_texh_3d` — 3D lookup table

`array`

The engine exhaust temperature lookup table is a function of commanded engine torque, engine speed, and engine temperature, $T_{exh} = f(T_{cmd}, N, Temp_{Eng})$, where:

- $T_{exh}$ is exhaust temperature, in K.
- $T_{cmd}$ is commanded engine torque, in N·m.
- $N$ is engine speed, in rpm.
- $Temp_{Eng}$ is engine temperature, in K.

**Dependencies**

To enable this parameter, select **Input engine temperature**.

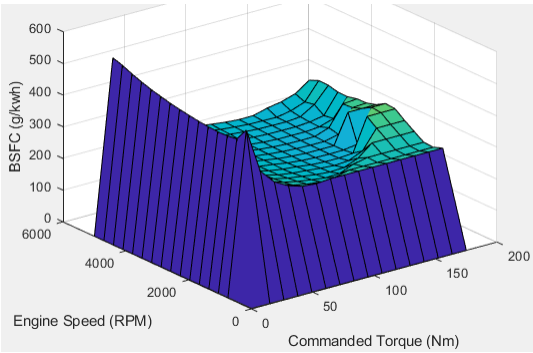
**Efficiency**

### `BSFC map, f_eff` — 2D lookup table

M-by-N-by-L array

The brake-specific fuel consumption (BSFC) efficiency is a function of commanded engine torque and engine speed, $BSFC = f(T_{cmd}, N)$, where:

- $BSFC$ is BSFC, in g/kWh.
- $T_{cmd}$ is commanded engine torque, in N·m.
- $N$ is engine speed, in rpm.




**Dependencies**

To enable this parameter, clear **Input engine temperature**.

### `Plot BSFC map` — Plot table

button

Click to plot table.

**Dependencies**

To enable this parameter, clear **Input engine temperature**.

### `BSFC map, f_eff_3d` — 3D lookup table

M-by-N-by-L array

The brake-specific fuel consumption (BSFC) efficiency is a function of commanded engine torque, engine speed, and engine temperature, $BSFC = f(T_{cmd}, N, Temp_{Eng})$, where:

- $BSFC$ is BSFC, in g/kWh.
- $T_{cmd}$ is commanded engine torque, in N·m.
- $N$ is engine speed, in rpm.
- $Temp_{Eng}$ is engine temperature, in K.

**Dependencies**

To enable this parameter, select **Input engine temperature**.

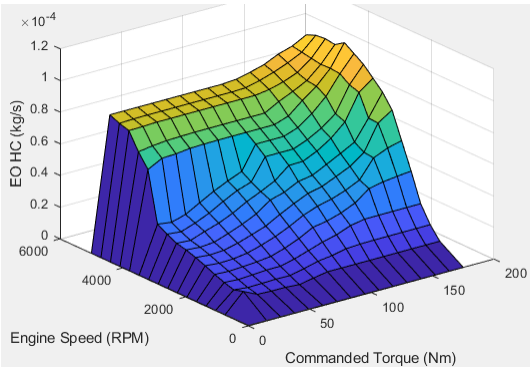
### HC

#### `EO HC map, f_hc` — 2D lookup table

M-by-N matrix

The engine-out hydrocarbon emissions are a function of commanded engine torque and engine speed, $EO\ HC = f(T_{cmd}, N)$, where:

- $EO\ HC$ is engine-out hydrocarbon emissions, in kg/s.
- $T_{cmd}$ is commanded engine torque, in N·m.
- $N$ is engine speed, in rpm.




**Dependencies**

To enable this parameter, clear **Input engine temperature**.

#### `Plot EO HC map` — Plot table

button

Click to plot table.

**Dependencies**

To enable this parameter, clear **Input engine temperature**.

#### `EO HC map, f_hc_3d` — 3D lookup table

M-by-N-by-L array

The engine-out hydrocarbon emissions are a function of commanded engine torque, engine speed, and engine temperature, $EO\ HC = f(T_{cmd}, N, Temp_{Eng})$, where:

- $EO\ HC$ is engine-out hydrocarbon emissions, in kg/s.
- $T_{cmd}$ is commanded engine torque, in N·m.
- $N$ is engine speed, in rpm.
- $Temp_{Eng}$ is engine temperature, in K.

**Dependencies**

To enable this parameter, select **Input engine temperature**.

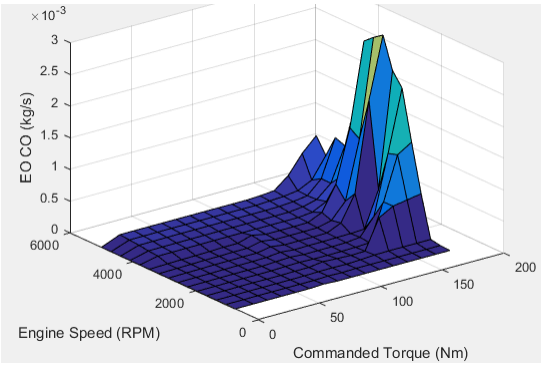
### CO

#### `EO CO map, f_co` — 2D lookup table

M-by-N matrix

The engine-out carbon monoxide emissions are a function of commanded engine torque and engine speed, $EO\ CO = f(T_{cmd}, N)$, where:

- $EO\ CO$ is engine-out carbon monoxide emissions, in kg/s.
- $T_{cmd}$ is commanded engine torque, in N·m.
- $N$ is engine speed, in rpm.




**Dependencies**

To enable this parameter, clear **Input engine temperature**.

#### `Plot EO CO map` — Plot table

button

Click to plot table.

**Dependencies**

To enable this parameter, clear **Input engine temperature**.

#### `EO HC map, f_hc_3d` — 3D lookup table

M-by-N-by-L array

The engine-out hydrocarbon emissions are a function of commanded engine torque, engine speed, and engine temperature, $EO\ HC = f(T_{cmd}, N, Temp_{Eng})$, where:

- $EO\ HC$ is engine-out hydrocarbon emissions, in kg/s.
- $T_{cmd}$ is commanded engine torque, in N·m.
- $N$ is engine speed, in rpm.
- $Temp_{Eng}$ is engine temperature, in K.

**Dependencies**

To enable this parameter, select **Input engine temperature**.

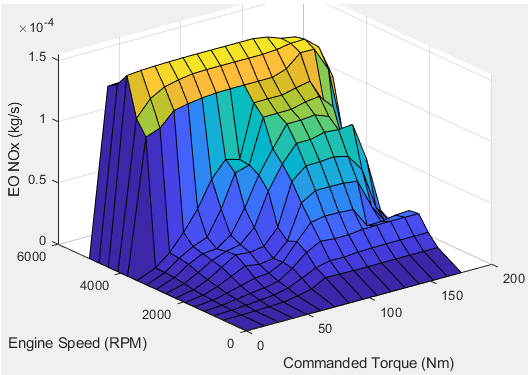
**NOx**

### `EO NOx map, f_nox` — 2D lookup table

M-by-N matrix

The engine-out nitric oxide and nitrogen dioxide emissions are a function of commanded engine torque and engine speed, $EO\ NOx = f(T_{cmd}, N)$, where:

- $EO\ NOx$ is engine-out nitric oxide and nitrogen dioxide emissions, in kg/s.
- $T_{cmd}$ is commanded engine torque, in N·m.
- $N$ is engine speed, in rpm.




**Dependencies**

To enable this parameter, clear **Input engine temperature**.

### `Plot EO NOx map` — Plot table

button

Click to plot table.

**Dependencies**

To enable this parameter, clear **Input engine temperature**.

### `EO NOx map, f_nox_3d` — 3D lookup table

M-by-N-by-L array

The engine-out nitric oxide and nitrogen dioxide emissions are a function of commanded engine torque, engine speed, and engine temperature, $EO\ NOx = f(T_{cmd}, N, Temp_{Eng})$, where:

- $EO\ NOx$ is engine-out nitric oxide and nitrogen dioxide emissions, in kg/s.
- $T_{cmd}$ is commanded engine torque, in N·m.
- $N$ is engine speed, in rpm.
- $Temp_{Eng}$ is engine temperature, in K.

**Dependencies**

To enable this parameter, select **Input engine temperature**.

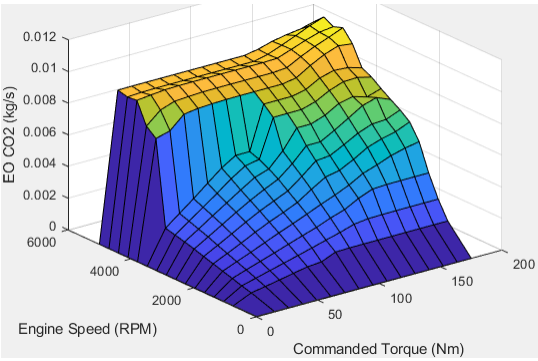
### CO2

#### `EO CO2 map, f_co2` — 2D lookup table

M-by-N matrix

The engine-out carbon dioxide emissions are a function of commanded engine torque and engine speed, $EO\ CO2 = f(T_{cmd}, N)$, where:

- $EO\ CO2$ is engine-out carbon dioxide emissions, in kg/s.
- $T_{cmd}$ is commanded engine torque, in N·m.
- $N$ is engine speed, in rpm.




**Dependencies**

To enable this parameter, clear **Input engine temperature**.

#### `Plot CO2 map` — Plot table

button

Click to plot table.

**Dependencies**

To enable this parameter, clear **Input engine temperature**.

#### `EO CO2 map, f_co2_3d` — 3D lookup table

M-by-N-by-L array

The engine-out carbon dioxide emissions are a function of commanded engine torque, engine speed, and engine temperature, $EO\ CO2 = f(T_{cmd}, N, Temp_{Eng})$, where:

- $EO\ CO2$ is engine-out carbon dioxide emissions, in kg/s.
- $T_{cmd}$ is commanded engine torque, in N·m.
- $N$ is engine speed, in rpm.
- $Temp_{Eng}$ is engine temperature, in K.

**Dependencies**

To enable this parameter, select **Input engine temperature**.

#### PM

**`EO PM map, f_pm` — 2D lookup table**
M-by-N matrix

The engine-out particulate matter emissions are a function of commanded engine torque and engine speed, where:

- *EO PM* is engine-out PM emissions, in kg/s.
- $T_{cmd}$ is commanded engine torque, in N·m.
- $N$ is engine speed, in rpm.

**Dependencies**

To enable this parameter, clear **Input engine temperature**.

**`Plot EO PM map` — Plot table**
button

Click to plot table.

**Dependencies**

To enable this parameter, clear **Input engine temperature**.

**`EO PM map, f_pm_3d` — 3D lookup table**
M-by-N-by-L array

The engine-out particulate matter emissions are a function of commanded engine torque, engine speed, and engine temperature, where:

- *EO PM* is engine-out PM emissions, in kg/s.
- $T_{cmd}$ is commanded engine torque, in N·m.
- $N$ is engine speed, in rpm.
- $Temp_{Eng}$ is engine temperature, in K.

**Dependencies**

To enable this parameter, select **Input engine temperature**.

## Extended Capabilities

**C/C++ Code Generation**
Generate C and C++ code using Simulink® Coder™.

## See Also

Mapped CI Engine | Mapped Motor | SI Core Engine

**Topics**
“Generate Mapped SI Engine from a Spreadsheet”
“Engine Calibration Maps”
“Model-Based Calibration Toolbox”

**Introduced in R2017a**

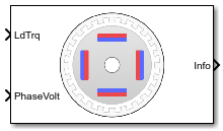
5

# Electric Motor, Converters, Inverter Blocks

# Interior PMSM

Three-phase interior permanent magnet synchronous motor with sinusoidal back electromotive force

**Library:** Powertrain Blockset / Propulsion / Electric Motors and Inverters
Motor Control Blockset / Electrical Systems / Motors




## Description

The Interior PMSM block implements a three-phase interior permanent magnet synchronous motor (PMSM) with sinusoidal back electromotive force. The block uses the three-phase input voltages to regulate the individual phase currents, allowing control of the motor torque or speed.

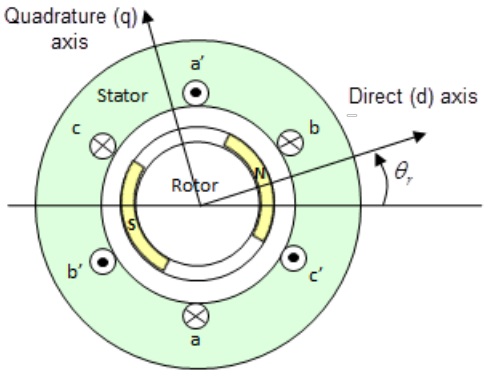
By default, the block sets the **Simulation type** parameter to `Continuous` to use a continuous sample time during simulation. If you want to generate code for fixed-step double- and single-precision targets, considering setting the parameter to `Discrete`. Then specify a **Sample Time, Ts** parameter.

On the **Parameters** tab, if you select `Back-emf`, the block implements this equation to calculate the permanent flux linkage constant.

$$\lambda_{pm} = \frac{1}{\sqrt{3}} \cdot \frac{K_e}{1000P} \cdot \frac{60}{2\pi}$$

### Motor Construction

This figure shows the motor construction with a single pole pair on the motor.




The motor magnetic field due to the permanent magnets creates a sinusoidal rate of change of flux with motor angle.

For the axes convention, the *a*-phase and permanent magnet fluxes are aligned when motor angle $\theta_r$ is zero.

### Three-Phase Sinusoidal Model Electrical System

The block implements these equations, expressed in the motor flux reference frame (dq frame). All quantities in the motor reference frame are referred to the stator.

$$\omega_e = P\omega_m$$

$$\frac{d}{dt}i_d = \frac{1}{L_d}v_d - \frac{R}{L_d}i_d + \frac{L_q}{L_d}P\omega_m i_q$$

$$\frac{d}{dt}i_q = \frac{1}{L_q}v_q - \frac{R}{L_q}i_q - \frac{L_d}{L_q}P\omega_m i_d - \frac{\lambda_{pm}P\omega_m}{L_q}$$

$$T_e = 1.5P[\lambda_{pm}i_q + (L_d - L_q)i_d i_q]$$

The $L_q$ and $L_d$ inductances represent the relation between the phase inductance and the motor position due to the saliency of the motor.

The equations use these variables.

| | |
|---|---|
| $L_q$, $L_d$ | q- and d-axis inductances |
| $R$ | Resistance of the stator windings |
| $i_q$, $i_d$ | q- and d-axis currents |
| $v_q$, $v_d$ | q- and d-axis voltages |
| $\omega_m$ | Angular mechanical velocity of the motor |
| $\omega_e$ | Angular electrical velocity of the motor |
| $\lambda_{pm}$ | Permanent flux linkage constant |
| $K_e$ | Back electromotive force (EMF) |
| $P$ | Number of pole pairs |
| $T_e$ | Electromagnetic torque |
| $\Theta_e$ | Electrical angle |

### Mechanical System

The motor angular velocity is given by:

$$\frac{d}{dt}\omega_m = \frac{1}{J}\left(T_e - T_f - F\omega_m - T_m\right)$$

$$\frac{d\theta_m}{dt} = \omega_m$$

The equations use these variables.

| | |
|---|---|
| $J$ | Combined inertia of motor and load |
| $F$ | Combined viscous friction of motor and load |
| $\theta_m$ | Motor mechanical angular position |
| $T_m$ | Motor shaft torque |
| $T_e$ | Electromagnetic torque |

| | |
|---|---|
| $T_f$ | Motor shaft static friction torque |
| $\omega_m$ | Angular mechanical velocity of the motor |

### Power Accounting

For the power accounting, the block implements these equations.

| Bus Signal | | | Description | Variable | Equations |
|---|---|---|---|---|---|
| `PwrInfo` | `PwrTrnsfrd` — Power transferred between blocks<br>• Positive signals indicate flow into block<br>• Negative signals indicate flow out of block | `PwrMtr` | Mechanical power | $P_{mot}$ | $P_{mot} = \quad -\omega_m T_e$ |
| | | `PwrBus` | Electrical power | $P_{bus}$ | $P_{bus} = \quad v_{an} i_a + \quad v_{bn} i_b + v_{cn} i_c$ |
| | `PwrNotTrnsfrd` — Power crossing the block boundary, but not transferred<br>• Positive signals indicate an input<br>• Negative signals indicate a loss | `PwrElecLoss` | Resistive power loss | $P_{elec}$ | $P_{elec} = \quad -\frac{3}{2}(R_s i_{sd}^2 + R_s i_{sq}^2)$ |
| | | `PwrMechLoss` | Mechanical power loss | $P_{mech}$ | When **Port Configuration** is set to `Torque`:<br>$P_{mech} = \quad -\left(\omega_m^2 F + \quad \lvert\omega_m\rvert T_f\right)$<br>When **Port Configuration** is set to `Speed`:<br>$P_{mech} = \quad 0$ |
| | `PwrStored` — Stored energy rate of change<br>• Positive signals indicate an increase<br>• Negative signals indicate a decrease | `PwrMtrStored` | Stored motor power | $P_{str}$ | $P_{str} = \quad P_{bus} + \quad P_{mot} + \quad P_{elec} \quad + \quad P_{mech}$ |

The equations use these variables.

| | |
|---|---|
| $R_s$ | Stator resistance |
| $i_a$, $i_b$, $i_c$ | Stator phase a, b, and c current |
| $i_{sq}$, $i_{sd}$ | Stator q- and d-axis currents |
| $v_{an}$, $v_{bn}$, $v_{cn}$ | Stator phase a, b, and c voltage |
| $\omega_m$ | Angular mechanical velocity of the rotor |
| $F$ | Combined motor and load viscous damping |

| | |
|---|---|
| $T_e$ | Electromagnetic torque |
| $T_f$ | Combined motor and load friction torque |

## Ports

### Input

#### `LdTrq` — Motor shaft torque

`scalar`

Motor shaft input torque, $T_m$, in N·m.

##### Dependencies

To create this port, select `Torque` for the **Port Configuration** parameter.

#### `Spd` — Motor shaft speed

`scalar`

Angular velocity of the motor, $\omega_m$, in rad/s.

##### Dependencies

To create this port, select `Speed` for the **Port Configuration** parameter.

#### `PhaseVolt` — Stator terminal voltages

1-by-3 array

Stator terminal voltages, $V_a$, $V_b$, and $V_c$, in V.

##### Dependencies

To create this port, select `Speed` or `Torque` for the **Port Configuration** parameter.

### Output

#### `Info` — Bus signal

bus

The bus signal contains these block calculations.

| Signal | Description | Variable | Units |
|---|---|---|---|
| `IaStator` | Stator phase current A | $i_a$ | A |
| `IbStator` | Stator phase current B | $i_b$ | A |
| `IcStator` | Stator phase current C | $i_c$ | A |
| `IdSync` | Direct axis current | $i_d$ | A |
| `IqSync` | Quadrature axis current | $i_q$ | A |
| `VdSync` | Direct axis voltage | $v_d$ | V |
| `VqSync` | Quadrature axis voltage | $v_q$ | V |
| `MtrSpd` | Angular mechanical velocity of the motor | $\omega_m$ | rad/s |

| Signal | | | Description | Variable | Units |
|---|---|---|---|---|---|
| MtrPos | | | Motor mechanical angular position | $\theta_m$ | rad |
| MtrTrq | | | Electromagnetic torque | $T_e$ | N·m |
| PwrInfo | PwrTrnsfrd | PwrMtr | Mechanical power | $P_{mot}$ | W |
| | | PwrBus | Electrical power | $P_{bus}$ | W |
| | PwrNotTrnsfrd | PwrElecLoss | Resistive power loss | $P_{elec}$ | W |
| | | PwrMechLoss | Mechanical power loss | $P_{mech}$ | W |
| | PwrStored | PwrMtrStored | Stored motor power | $P_{str}$ | W |

**`PhaseCurr` — Phase a, b, c current**
1-by-3 array

Phase a, b, c current, $i_a$, $i_b$, and $i_c$, in A.

**`MtrTrq` — Motor torque**
`scalar`

Motor torque, $T_{mtr}$, in N·m.

**Dependencies**

To create this port, select `Speed` for the **Mechanical input configuration** parameter.

**`MtrSpd` — Motor speed**
`scalar`

Angular speed of the motor, $\omega_{mtr}$, in rad/s.

**Dependencies**

To create this port, select `Torque` for the **Mechanical input configuration** parameter.

## Parameters

**Block Options**

**`Mechanical input configuration` — Select port configuration**
`Torque` (default) | `Speed`

This table summarizes the port configurations.

| Port Configuration | Creates Input Port | Creates Output Port |
|---|---|---|
| Torque | LdTrq | MtrSpd |
| Speed | Spd | MtrTrq |

**`Simulation type` — Select simulation type**
`Continuous` (default) | `Discrete`

By default, the block uses a continuous sample time during simulation. If you want to generate code for single-precision targets, considering setting the parameter to `Discrete`.

#### Dependencies

Setting **Simulation type** to `Discrete` creates the **Sample Time, Ts** parameter.

### `Sample Time (Ts)` — Sample time for discrete integration

`scalar`

Integration sample time for discrete simulation, in s.

#### Dependencies

Setting **Simulation type** to `Discrete` creates the **Sample Time, Ts** parameter.

#### Parameters

### `Number of pole pairs (P)` — Pole pairs

`scalar`

Motor pole pairs, *P*.

### `Stator phase resistance per phase (Rs)` — Resistance

`scalar`

Stator phase resistance per phase, $R_s$, in ohm.

### `Stator d-axis and q-axis inductance (Ldq)` — Inductance

`vector`

Stator d-axis and q-axis inductance, $L_d$, $L_q$, in H.

### `Permanent flux linkage constant (lambda_pm)` — Flux

`scalar`

Permanent flux linkage constant, $\lambda_{pm}$, in Wb.

### `Back-emf constant (Ke)` — Back electromotive force

`scalar`

Back electromotive force, EMF, $K_e$, in Vpk_LL/krpm. Vpk_LL is the peak voltage line-to-line measurement.

To calculate the permanent flux linkage constant, the block implements this equation.

$$\lambda_{pm} = \frac{1}{\sqrt{3}} \cdot \frac{K_e}{1000P} \cdot \frac{60}{2\pi}$$

### `Physical inertia, viscous damping, and static friction (mechanical)` — Inertia, damping, friction

`vector`

Mechanical properties of the motor:

- Inertia, *J*, in kgm^2

- Viscous damping, $F$, in N·m/(rad/s)
- Static friction, $T_f$, in N·m

#### Dependencies

To enable this parameter, select the `Torque` configuration parameter.

#### Initial Values

**`Initial d-axis and q-axis current (idq0)` — Current**
`vector`

Initial q- and d-axis currents, $i_q$, $i_d$, in A.

**`Initial mechanical position (theta_init)` — Angle**
`scalar`

Initial motor angular position, $\theta_{m0}$, in rad.

**`Initial mechanical speed (omega_init)` — Speed**
`scalar`

Initial angular velocity of the motor, $\omega_{m0}$, in rad/s.

#### Dependencies

To enable this parameter, select the `Torque` configuration parameter.

## References

[1] Kundur, P. *Power System Stability and Control*. New York, NY: McGraw Hill, 1993.

[2] Anderson, P. M. *Analysis of Faulted Power Systems*. Hoboken, NJ: Wiley-IEEE Press, 1995.

## Extended Capabilities

#### C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

## See Also

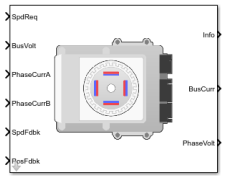
Flux-Based PMSM | Induction Motor | Interior PM Controller | Mapped Motor | Surface Mount PMSM

#### Introduced in R2017a

# Interior PM Controller

Torque-based, field-oriented controller for an internal permanent magnet synchronous motor

**Library:** Powertrain Blockset / Propulsion / Electric Motor Controllers

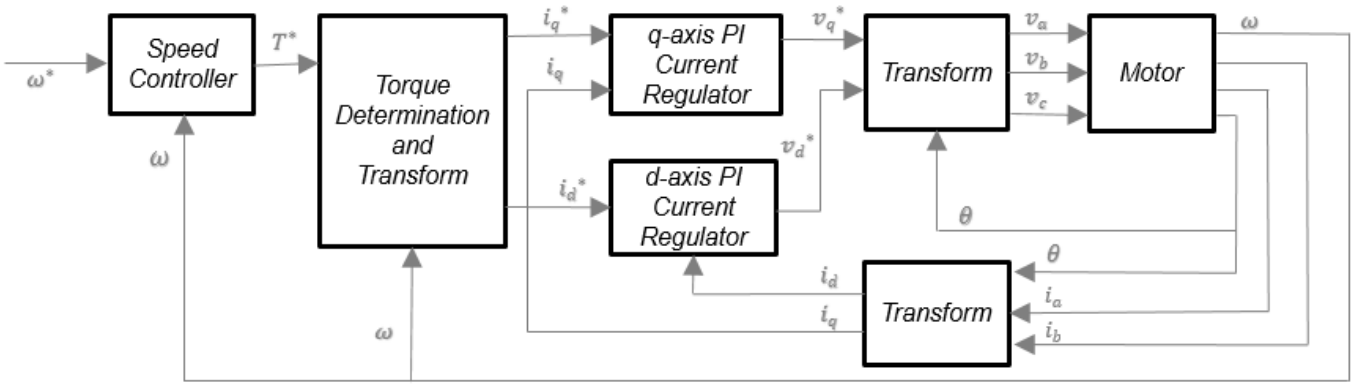



## Description

The Interior PM Controller block implements a torque-based, field-oriented controller for an internal permanent magnet synchronous motor (PMSM) with an optional outer-loop speed controller. The internal torque control implements strategies for achieving maximum torque per ampere (MTPA) and weakening the magnetic flux. You can specify either the speed or torque control type.

The Interior PM Controller implements equations for speed control, torque determination, regulators, transforms, and motors.

The figure illustrates the information flow in the block.




The block implements equations that use these variables.

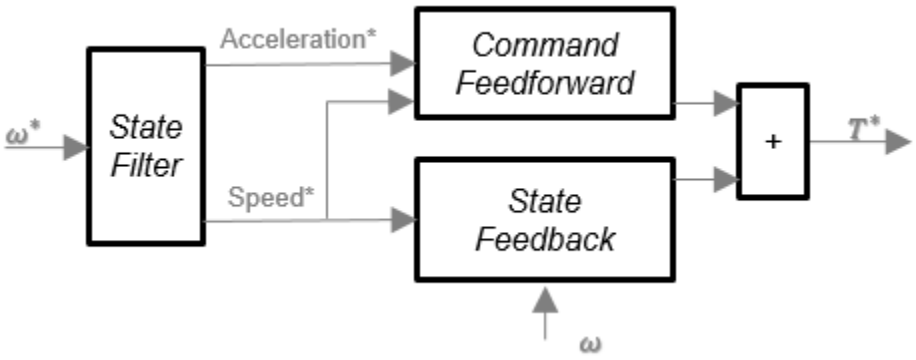
| | |
|---|---|
| $\omega$ | Rotor speed |
| $\omega$* | Rotor speed command |
| $T$* | Torque command |
| $i_d$ | d-axis current |
| $i^*_d$ | d-axis current command |
| $i_q$ | q-axis current |
| $i^*_q$ | q-axis current command |

| | |
|---|---|
| $v_d$, | d-axis voltage |
| $v^*_d$ | d-axis voltage command |
| $v_q$ | q-axis voltage |
| $v^*_q$ | q-axis voltage command |
| $v_a$, $v_b$, $v_c$ | Stator phase a, b, c voltages |
| $i_a$, $i_b$, $i_c$ | Stator phase a, b, c currents |

### Speed Controller

To implement the speed controller, select the **Control Type** parameter `Speed Control`. If you select the **Control Type** parameter `Torque Control`, the block does not implement the speed controller.

The speed controller determines the torque command by implementing a state filter, and calculating the feedforward and feedback commands. If you do not implement the speed controller, input a torque command to the Interior PM Controller block.




#### State Filter

The state filter is a low-pass filter that generates the acceleration command based on the speed command. On the **Speed Controller** tab:

- To make the speed-command lag time negligible, specify a **Bandwidth of the state filter** parameter.
- To calculate a **Speed regulation time constant, Ksf** gain based on the state filter bandwidth, select **Calculate Speed Regulator Gains**.

The discrete form of characteristic equation is given by:

$$z + K_{sf}T_{sm} - 1$$

The filter calculates the gain using this equation.

$$K_{sf} = \frac{1 - \exp\left(-T_{sm}2\pi EV_{sf}\right)}{T_{sm}}$$

The equations use these variables.

| | |
|---|---|
| $EV_{sf}$ | Bandwidth of the speed command filter |
| $T_{sm}$ | Motion controller sample time |
| $K_{sf}$ | Speed regulator time constant |

#### State Feedback

To generate the state feedback torque, the block uses the filtered speed error signal from the state filter. The feedback torque calculation also requires gains for speed regulator.

On the **Speed Controller** tab, select **Calculate Speed Regulator Gains** to calculate:

- **Proportional gain, ba**
- **Angular gain, Ksa**
- **Rotational gain, Kisa**

For the gain calculations, the block uses the inertia from the **Physical inertia, viscous damping, static friction** parameter value on the **Motor Parameters** tab.

The gains for the state feedback are calculated using these equations.

| Calculation | Equations |
|---|---|
| Discrete forms of characteristic equation | $z^3 + \frac{\left(-3J_p + T_s b_a + \quad T_s^2 K_{sa} + T_s^3 K_{isa}\right)}{J_p} z^2 + \frac{\left(3J_p - \quad 2T_s b_a - \quad T_s^2 K_{sa}\right)}{J_p} z + \frac{-J_p + T_s b_a}{J_p}$<br><br>$(z - p_1)(z - p_2)(z - p_3) = z^3 + (p_1 + p_2 + p_3)z^2 + (p_1 p_2 + p_2 p_3 + p_1 3)z^2 - p_1 p_2 p_3$ |
| Speed regulator proportional gain | $b_a = \frac{J_p - J_p p_1 p_2 p_3}{T_{sm}}$ |
| Speed regulator integral gain | $K_{sa} = \frac{J_p(p_1 p_2 + p_2 p_3 + p_3 p_1) - 3J_p + 2b_a T_{sm}}{T_{sm}^2}$ |
| Speed regulator double integral gain | $K_{isa} = \frac{-J_p(p_1 + p_2 + p_3) + 3J_p - b_a T_{sm} - K_{sa} T_{sm}^2}{T_{sm}^3}$ |

The equations use these variables.

| | |
|---|---|
| $P$ | Motor pole pairs |
| $b_a$ | Speed regulator proportional gain |
| $K_{sa}$ | Speed regulator integral gain |
| $K_{isa}$ | Speed regulator double integral gain |
| $J_p$ | Motor inertia |
| $T_{sm}$ | Motion controller sample time |

#### Command Feedforward

To generate the state feedforward torque, the block uses the filtered speed and acceleration from the state filter. Also, the feedforward torque calculation uses the inertia, viscous damping, and static friction. To achieve zero tracking error, the torque command is the sum of the feedforward and feedback torque commands.

Selecting **Calculate Speed Regulator Gains** on the **Speed Controller** tab updates the inertia, viscous damping, and static friction with the **Physical inertia, viscous damping, static friction** parameter values on the **Motor Parameters** tab.

The feedforward torque command uses this equation.

$$T_{cmd_ff} = J_p\dot{\omega}_m + F_v\omega_m + F_s\frac{\omega_m}{|\omega_m|}$$

where:

| | |
|---|---|
| $J_p$ | Motor inertia |
| $T_{cmd_ff}$ | Torque command feedforward |
| $F_s$ | Static friction torque constant |
| $F_v$ | Viscous friction torque constant |
| $F_s$ | Static friction torque constant |
| $\omega_m$ | Rotor speed |

### Torque Determination

The block uses a maximum torque per ampere (MTPA) trajectory to calculate the base speed and the current commands. The available bus voltage determines the base speed. The direct (d) and quadrature (q) permanent magnet (PM) determines the induced voltage.

| Calculation | Equations |
|---|---|
| Electrical base speed transition into field weakening | $\omega_{base} = \frac{v_{max}}{\sqrt{(L_q i_q)^2 + (L_d i_d + \lambda_{pm})^2}}$ |
| d-axis voltage | $v_d = -\omega_e L_q i_{q_{max}}$ |
| q-axis voltage | $v_q = \omega_e(L_d i_{d_max} + \lambda_{pm})$ |
| Maximum phase current | $i_{max}{}^2 = i_{d_\max}^2 + i_{q_\max}^2$ |
| Maximum line to neutral voltage | $v_{max} = \frac{v_{bus}}{\sqrt{3}}$ |
| d-axis phase current MTPA table | $I_m = \frac{2T_{max}}{3P\lambda_{pm}}$<br>$i_{d_mtpa} = \frac{\lambda_{pm}}{4(L_q - L_d)} - \sqrt{\frac{\lambda_{pm}^2}{16(L_q - L_d)^2} + \frac{I_m^2}{2}}$ |
| q-axis phase current MTPA table | $i_{q_mtpa} = \sqrt{I_m^2 - (i_{mtpa})^2}$ |
| Torque MTPA breakpoints | $T_{mtpa} = \frac{3}{2}P(\lambda_{pm} i_q + (L_d - L_q) i_d i_q)$ |

| Calculation | Equations |
|---|---|
| Field weakening, using the speed-based voltage limits | $\left(L_q i_q\right)^2 + \left(L_d i_d + \quad \lambda_{pm}\right)^2 \le \quad \frac{v_{max}^2}{\omega_e^2}$<br>$i_q = \quad \sqrt{i_{max}^2 - \quad i_d^2}$<br>$\left(L_d^2 - \quad L_q^2\right) i_d^2 + 2\lambda_{pm} L_d i_d + \quad \lambda_{pm} + \quad L_q^2 i_{max}^2 - \frac{v_{max}^2}{\omega_e^2} = 0$<br>$i_{dfw} = \quad \frac{-\lambda_{pm} L_d + \quad \sqrt{\left(\lambda_{pm} L_d\right)^2 - \quad \left(L_d^2 - \quad L_q^2\right)\left(\lambda_{pm}^2 + \quad L_q^2 i_{max}^2 - \quad \frac{v_{max}^2}{\omega_e^2}\right.}}{\left(L_d^2 - \quad L_q^2\right)}$<br>$T_{fw} = \quad \frac{3}{2} P\left(\lambda_{pm} i_{qfw} + \left(L_d - \quad L_q\right) i_{dfw} i_{qfw}\right)$ |
| Current command | `If` $\lvert\omega_e\rvert \le \quad \omega_{base}$<br>$i_{dref} = \quad i_{d_{mtpa}}(T_{ref})$<br>$i_{qref} = \quad i_{q_{mtpa}}(T_{ref})$<br>`Else`<br>$i_{dfw} = \quad \max\left(i_{dfw}, \quad - i_{max}\right)$<br>$i_{qfw} = \quad \sqrt{i_{max}^2 - \quad i_d^2}$<br>`If` $\quad T_{fw} < T_{ref}$<br>$i_{dref} = \quad i_{d_{fw}}$<br>$i_{qref} = \quad i_{q_{fw}}$<br>`Else`<br>$i_{dref} = \quad i_{d_{fw}}$<br>$i_{qref} = \quad \frac{T_{ref}}{\frac{3}{2} P\left(\lambda_{pm} + \left(L_d - \quad L_q\right) i_{dfw}\right)}$<br>`End`<br>`End` |

The equations use these variables.

| | |
|---|---|
| $i_{max}$ | Maximum phase current |
| $i_d$ | d-axis current |
| $i_q$ | q-axis current |
| $i_{d_max}$ | Maximum d-axis phase current |
| $i_{q_max}$ | Maximum q-axis phase current |
| $i_{d_mtpa}$ | d-axis phase current MTPA table |
| $i_{q_mtpa}$ | q-axis phase current MTPA table |
| $I_m$ | Estimated maximum current |
| $i_{dfw}$ | d-axis field weakening current |
| $i_{qfw}$ | q-axis field weakening current |

| | |
|---|---|
| $\omega_e$ | Rotor electrical speed |
| $\lambda_{pm}$ | Permanent magnet flux linkage |
| $v_d$ | d-axis voltage |
| $v_q$ | q-axis voltage |
| $v_{max}$ | Maximum line to neutral voltage |
| $v_{bus}$ | DC bus voltage |
| $L_d$ | d-axis winding inductance |
| $L_q$ | q-axis winding inductance |
| $P$ | Motor pole pairs |
| $T_{fw}$ | Field weakening torque |
| $T_{mtpa}$ | Torque MTPA breakpoints |

### Current Regulators

The block regulates the current with an anti-windup feature. Classic proportional-integrator (PI) current regulators do not consider the d-axis and q-axis coupling or the back-electromagnetic force (EMF) coupling. As a result, transient performance deteriorates. To account for the coupling, the block implements the complex vector current regulator (CVCR) in the scalar format of the rotor reference frame. The CVCR decouples:

- d-axis and q-axis current cross-coupling
- Back-EMF cross-coupling

The current frequency response is a first-order system, with a bandwidth of $EV_{current}$.

The block implements these equations.

| Calculation | Equations |
|---|---|
| Motor voltage, in the rotor reference frame | $L_d\frac{di_d}{dt} = v_d - R_s i_d + p\omega_m L_q i_q$<br>$L_d\frac{di_q}{dt} = v_q - R_s i_q - p\omega_m L_d i_d - p\omega_m \lambda_{pm}$ |
| Current regulator gains | $\omega_b = 2\pi EV_{current}$<br>$K_{p_d} = L_d\omega_b$<br>$K_{p_q} = L_q\omega_b$<br>$K_i = R_s\omega_b$ |
| Transfer functions | $\frac{i_d}{i_{dref}} = \frac{\omega_b}{s+\omega_b}$<br>$\frac{i_q}{i_{qref}} = \frac{\omega_b}{s+\omega_b}$ |

The equations use these variables.

| | |
|---|---|
| $EV_{current}$ | Current regulator bandwidth |
| $i_d$ | d-axis current |

| | |
|---|---|
| $i_q$ | q-axis current |
| $K_{p_d}$ | Current regulator d-axis gain |
| $K_{p_q}$ | Current regulator q-axis gain |
| $L_d$ | d-axis winding inductance |
| $L_q$ | q-axis winding inductance |
| $R_s$ | Stator phase winding resistance |
| $\omega_m$ | Rotor speed |
| $v_d$ | d-axis voltage |
| $v_q$ | q-axis voltage |
| $\lambda_{pm}$ | Permanent magnet flux linkage |
| $P$ | Motor pole pairs |

### Transforms

To calculate the voltages and currents in balanced three-phase ($a$, $b$) quantities, quadrature two-phase ($\alpha$, $\beta$) quantities, and rotating ($d$, $q$) reference frames, the block uses the Clarke and Park Transforms.

In the transform equations.

$$\omega_e = P\omega_m$$

$$\frac{d\theta_e}{dt} = \omega_e$$

| Transform | Description | Equations |
|---|---|---|
| Clarke | Converts balanced three-phase quantities ($a$, $b$) into balanced two-phase quadrature quantities ($\alpha$, $\beta$). | $x_\alpha = \frac{2}{3}x_a - \frac{1}{3}x_b - \frac{1}{3}x_c$<br>$x_\beta = \frac{\sqrt{3}}{2}x_b - \frac{\sqrt{3}}{2}x_c$ |
| Park | Converts balanced two-phase orthogonal stationary quantities ($\alpha$, $\beta$) into an orthogonal rotating reference frame ($d$, $q$). | $x_d = x_\alpha\cos\theta_e + x_\beta\sin\theta_e$<br>$x_q = -x_\alpha\sin\theta_e + x_\beta\cos\theta_e$ |
| Inverse Clarke | Converts balanced two-phase quadrature quantities ($\alpha$, $\beta$) into balanced three-phase quantities ($a$, $b$). | $x_a = x_a$<br>$x_b = -\frac{1}{2}x_\alpha + \frac{\sqrt{3}}{2}x_\beta$<br>$x_c = -\frac{1}{2}x_\alpha - \frac{\sqrt{3}}{2}x_\beta$ |
| Inverse Park | Converts an orthogonal rotating reference frame ($d$, $q$) into balanced two-phase orthogonal stationary quantities ($\alpha$, $\beta$). | $x_\alpha = x_d\cos\theta_e - x_q\sin\theta_e$<br>$x_\beta = x_d\sin\theta_e + x_q\cos\theta_e$ |

The transforms use these variables.

| | |
|---|---|
| $\omega_m$ | Rotor speed |
| $P$ | Motor pole pairs |

| | |
|---|---|
| $\omega_e$ | Rotor electrical speed |
| $\Theta_e$ | Rotor electrical angle |
| $x$ | Phase current or voltage |

### Motor

The block uses the phase currents and phase voltages to estimate the DC bus current. Positive current indicates battery discharge. Negative current indicates battery charge. The block uses these equations.

| | |
|---|---|
| Load power | $Ld_{Pwr} = v_a \quad i_a + \quad v_b \quad i_b + \quad v_c \quad i_c$ |
| Source power | $Src_{Pwr} = \quad Ld_{Pwr} + Pwr_{Loss}$ |
| DC bus current | $i_{bus} = \quad \frac{Src_{Pwr}}{v_{bus}}$ |
| Estimated rotor torque | $MtrTrq_{est} = 1.5P[\lambda i_q + (L_d - L_q) i_d i_q]$ |
| Power loss for single efficiency source to load | $Pwr_{Loss} = \frac{100 - Eff}{Eff} \cdot Ld_{Pwr}$ |
| Power loss for single efficiency load to source | $Pwr_{Loss} = \frac{100 - Eff}{100} \cdot \lvert Ld_{Pwr} \rvert$ |
| Power loss for tabulated efficiency | $Pwr_{Loss} = \quad f(\omega_m, MtrTrq_{est})$ |

The equations use these variables.

| | |
|---|---|
| $v_a$, $v_b$, $v_c$ | Stator phase a, b, c voltages |
| $v_{bus}$ | Estimated DC bus voltage |
| $i_a$, $i_b$, $i_c$ | Stator phase a, b, c currents |
| $i_{bus}$ | Estimated DC bus current |
| $Eff$ | Overall inverter efficiency |
| $\omega_m$ | Rotor mechanical speed |
| $L_q$ | q-axis winding inductance |
| $L_d$ | d-axis winding inductance |
| $i_q$ | q-axis current |
| $i_d$ | d-axis current |
| $\lambda$ | Permanent magnet flux linkage |
| $P$ | Motor pole pairs |

### Electrical Losses

To specify the electrical losses, on the **Electrical Losses** tab, for **Parameterize losses by**, select one of these options.

| Setting | Block Implementation |
|---|---|
| `Single efficiency measurement` | Electrical loss calculated using a constant value for inverter efficiency. |

| Setting | Block Implementation |
|---|---|
| `Tabulated loss data` | Electrical loss calculated as a function of motor speeds and load torques. |
| `Tabulated efficiency data` | Electrical loss calculated using inverter efficiency that is a function of motor speeds and load torques.<br>• Converts the efficiency values you provide into losses and uses the tabulated losses for simulation.<br>• Ignores efficiency values you provide for zero speed or zero torque. Losses are assumed zero when either torque or speed is zero.<br>• Uses linear interpolation to determine losses. Provide tabulated data for low speeds and low torques, as required, to get the desired level of accuracy for lower power conditions.<br>• Does not extrapolate loss values for speed and torque magnitudes that exceed the range of the table. |

For best practice, use `Tabulated loss data` instead of `Tabulated efficiency data`:

- Efficiency becomes ill defined for zero speed or zero torque.
- You can account for fixed losses that are still present for zero speed or torque.

## Ports

### Input

#### SpdReq — Rotor speed command

`scalar`

Rotor speed command, $\omega^*_m$, in rad/s.

##### Dependencies

To create this port, select `Speed Control` for the **Control Type** parameter.

#### TrqCmd — Torque command

`scalar`

Torque command, $T^*$, in N·m.

##### Dependencies

To create this port, select `Torque Control` for the **Control Type** parameter.

#### BusVolt — DC bus voltage

`scalar`

DC bus voltage, $v_{bus}$, in V.

#### PhaseCurrA — Current

`scalar`

Stator current phase a, $i_a$, in A.

**PhaseCurrB — Current**
scalar

Stator current phase b, $i_b$, in A.

**SpdFdbk — Rotor speed**
scalar

Rotor speed, $\omega_m$, in rad/s.

**PosFdbk — Rotor electrical angle**
scalar

Rotor electrical angle, $\Theta_m$, in rad.

**Output**

**Info — Bus signal**
bus

Bus signal containing these block calculations.

| Signal | Description | Units |
|---|---|---|
| SrcPwr | Source power | W |
| LdPwr | Load power | W |
| PwrLoss | Power loss | W |
| MtrTrqEst | Estimated motor torque | N·m |

**BusCurr — Bus current**
scalar

Estimated DC bus current, $i_{bus}$, in A.

**PhaseVolt — Stator terminal voltages**
array

Stator terminal voltages, $V_a$, $V_b$, and $V_c$, in V.

## Parameters

**Block Options**

**Control Type — Select control**
Speed Control (default) | Torque Control

If you select Torque Control, the block does not implement the speed controller.

This table summarizes the port configurations.

| Port Configuration | Creates Ports |
|---|---|
| Speed Control | SpdReq |

| Port Configuration | Creates Ports |
|---|---|
| `Torque Control` | `TrqCmd` |

#### Motor Parameters

### `Stator resistance, Rs` — Resistance

`0.02` (default) | `scalar`

Stator phase winding resistance, $R_s$, in ohm.

#### Dependencies

This table summarizes the parameter dependencies.

| Parameter | Used to Derive | |
|---|---|---|
| | Parameter | Tab |
| **Stator resistance, Rs** | **D and Q axis integral gain, Ki** | **Current Controller** |

### `D-axis inductance, Ld` — Inductance

`1.7e-3` (default) | `scalar`

D-axis winding inductance, $L_d$, in H.

#### Dependencies

This table summarizes the parameter dependencies.

| Parameter | Used to Derive | |
|---|---|---|
| | Parameter | Tab |
| **D-axis inductance, Ld** | **Torque Breakpoints, T_mtpa**<br>**D-axis table data, id_mtpa**<br>**Q-axis table data, iq_mtpa**<br>**D, q, and max current limits, idq_limits** | **Id and Iq Calculation** |

### `Q-axis inductance, Lq` — Inductance

`3.2e-3` (default) | `scalar`

Q-axis winding inductance, $L_q$, in H.

#### Dependencies

This table summarizes the parameter dependencies.

| Parameter | Used to Derive | |
|---|---|---|
| | Parameter | Tab |
| Q-axis inductance, Lq | Torque Breakpoints, T_mtpa<br>D-axis table data, id_mtpa<br>Q-axis table data, iq_mtpa<br>D, Q, and max current limits, idq_limits | Id and Iq Calculation |

**`Permanent magnet flux, lambda_pm` — Flux**
`0.2205` (default) | `scalar`

Permanent magnet flux, $\lambda_{pm}$, in Wb.

#### Dependencies

This table summarizes the parameter dependencies.

| Parameter | Used to Derive | |
|---|---|---|
| | Parameter | Tab |
| Permanent magnet flux, lambda_pm | Torque Breakpoints, T_mtpa<br>D-axis table data, id_mtpa<br>Q-axis table data, iq_mtpa<br>D, Q, and max current limits, idq_limits | Id and Iq Calculation |

**`Number of pole pairs, PolePairs` — Poles**
`4` (default) | `scalar`

Motor pole pairs, *P*.

#### Dependencies

This table summarizes the parameter dependencies.

| Parameter | Used to Derive | |
|---|---|---|
| | Parameter | Tab |
| Number of pole pairs, PolePairs | Torque Breakpoints, T_mtpa<br>D-axis table data, id_mtpa<br>Q-axis table data, iq_mtpa<br>D, Q, and max current limits, idq_limits | Id and Iq Calculation |

**`Physical inertia, viscous damping, static friction, Mechanical` — Inertia, damping, friction**
`[0.0027, 4.924e-4, 0]` (default) | `vector`

Mechanical properties of the motor:

- Motor inertia, $F_v$, in kgm^2
- Viscous friction torque constant, $F_v$, in N·m/(rad/s)
- Static friction torque constant, $F_s$, in N·m

#### Dependencies

To enable this parameter, set the **Control Type** parameter to `Speed Control`.

For the gain calculations, the block uses the inertia from the **Physical inertia, viscous damping, static friction** parameter value that is on the **Motor Parameters** tab.

This table summarizes the parameter dependencies.

| Parameter | Used to Derive | |
|---|---|---|
| | Parameter | Tab |
| **Physical inertia, viscous damping, static friction, Mechanical** | **Proportional gain, ba**<br>**Angular gain, Ksa**<br>**Rotational gain, Kisa**<br>**Inertia compensation, Jcomp**<br>**Viscous damping compensation, Fv**<br>**Static friction, Fs** | **Speed Controller** |

#### Id and Iq Calculation

**`Maximum torque, T_max` — Torque**
`60` (default) | `scalar`

Maximum torque, in N·m.

#### Dependencies

This table summarizes the parameter dependencies.

| Parameter | Used to Derive | |
|---|---|---|
| | Parameter | Tab |
| Maximum torque, T_max | Torque Breakpoints, T_mtpa<br><br>D-axis table data, id_mtpa<br><br>Q-axis table data, iq_mtpa<br><br>D, Q, and max current limits, idq_limits | Id and Iq Calculation |

**`MTPA table breakpoints, bp` — Number of breakpoints**
`10` (default) | `scalar`

**Dependencies**

This table summarizes the parameter dependencies.

| Parameter | Used to Derive | |
|---|---|---|
| | Parameter | Tab |
| MTPA table breakpoints, pb | Torque Breakpoints, T_mtpa<br><br>D-axis table data, id_mtpa<br><br>Q-axis table data, iq_mtpa<br><br>D, Q, and max current limits, idq_limits | Id and Iq Calculation |

**`Calculate MTPA Table Data` — Derive parameters**
button

Click to derive parameters.

**Dependencies**

On the **Id and Iq Calculation** tab, when you select **Calculate MPTA Table data**, the block calculates derived parameters. The table summarizes the derived parameter dependencies on other block parameters.

| Derived Parameter on Id and Iq Calculation tab | | Depends On | |
|---|---|---|---|
| | | Parameter | Tab |
| Torque Breakpoints, T_mtpa | $T_{mtpa} = \frac{3}{2}P\left(\lambda_{pm}i_q + \left(L_d - L_q\right)i_d i_q\right)$ | Maximum torque, T_max<br><br>MTPA table breakpoints, pb | Id and Iq Calculation |

| Derived Parameter on Id and Iq Calculation tab | | Depends On | |
|---|---|---|---|
| | | **Parameter** | **Tab** |
| **D-axis table data, id_mtpa** | $I_m = \frac{2T_{max}}{3P\lambda_{pm}}$<br>$i_{d_mtpa} = \frac{\lambda_{pm}}{4(L_q - L_d)} - \sqrt{\frac{\lambda_{pm}^2}{16(L_q - L_d)^2} + \frac{I_m^2}{2}}$ | **Permanent magnet flux, lambda_pm**<br>**D-axis inductance, Ld** | **Motor Parameters** |
| **Q-axis table data, iq_mtpa** | $i_{q_mtpa} = \sqrt{I_m^2 - (i_{mtpa})^2}$ | **Q-axis inductance, Lq** | |
| **D, Q, and max current limits, idq_limits** | | **Number of pole pairs, PolePairs** | |

The equations use these variables.

| | |
|---|---|
| $i_{max}$ | Maximum phase current |
| $i_d$ | d-axis current |
| $i_q$ | q-axis current |
| $i_{d_max}$ | Maximum d-axis phase current |
| $i_{q_max}$ | Maximum q-axis phase current |
| $i_{d_mtpa}$ | d-axis phase current MTPA table |
| $i_{q_mtpa}$ | q-axis phase current MTPA table |
| $\lambda_{pm}$ | Permanent magnet flux linkage |
| $L_d$ | d-axis winding inductance |
| $L_q$ | q-axis winding inductance |
| $P$ | Motor pole pairs |
| $T_{mtpa}$ | Torque MTPA breakpoints |
| $I_m$ | Estimated maximum current |

**`Torque Breakpoints, T_mtpa` — Derived**

`[0 6.41323967543524 12.8472271930531 19.3221671098192 25.8572437875407 32.4702594835269 39.177408529382 45.9931820911486 52.930379967864 60.0001984561834]` (default) | `vector`

Derived torque breakpoints, in N·m.

#### Dependencies

This table summarizes the parameter dependencies.

| Parameter | Dependency | |
|---|---|---|
| | **Parameter** | **Tab** |
| **Torque Breakpoints, T_mtpa** | **Maximum torque, T_max**<br>**MTPA table breakpoints, pb** | **Id and Iq Calculation** |

| Parameter | Dependency | |
|---|---|---|
| | Parameter | Tab |
| | **Permanent magnet flux, lambda_pm**<br><br>**D-axis inductance, Ld**<br><br>**Q-axis inductance, Lq**<br><br>**Number of pole pairs, PolePairs** | **Motor Parameters** |

**`D-axis table data, id_mtpa` — Derived**
`[0 -0.159333276810563 -0.633258709677809 -1.41005695027301 -2.47173666500257 -3.79592548539108 -5.35786489234899 -7.13217478652462 -9.09420364751938 -11.2209236729158]` (default) | `vector`

Derived d-axis table data, in A.

**Dependencies**

This table summarizes the parameter dependencies.

| Parameter | Dependency | |
|---|---|---|
| | Parameter | Tab |
| **D-axis table data, id_mtpa** | **Maximum torque, T_max**<br><br>**MTPA table breakpoints, pb** | **Id and Iq Calculation** |
| | **Permanent magnet flux, lambda_pm**<br><br>**D-axis inductance, Ld**<br><br>**Q-axis inductance, Lq**<br><br>**Number of pole pairs, PolePairs** | **Motor Parameters** |

**`Q-axis table data, iq_mtpa` — Derived**
`[0 4.84224935172196 9.66902512748937 14.4660510262181 19.2212063070062 23.9250934510846 28.5711892538614 33.1556572971289 37.6769488702032 42.1353166357157]` (default) | `vector`

Derived q-axis table data, in A.

**Dependencies**

This table summarizes the parameter dependencies.

| Parameter | Dependency | |
|---|---|---|
| | Parameter | Tab |
| D-axis table data, id_mtpa | Maximum torque, T_max<br><br>MTPA table breakpoints, pb | Id and Iq Calculation |
| | Permanent magnet flux, lambda_pm<br><br>D-axis inductance, Ld<br><br>Q-axis inductance, Lq<br><br>Number of pole pairs, PolePairs | Motor Parameters |

**`D, Q, and max current limits, idq_limits` — Derived**
`[-11.2210468862948 42.1352838229553 43.6038305205566]` (default) | `array`

Derived d, q, and maximum current limits, in A.

#### Dependencies

This table summarizes the parameter dependencies.

| Parameter | Dependency | |
|---|---|---|
| | Parameter | Tab |
| D, Q, and max current limits, idq_limits | Maximum torque, T_max<br><br>MTPA table breakpoints, pb | Id and Iq Calculation |
| | Permanent magnet flux, lambda_pm<br><br>D-axis inductance, Ld<br><br>Q-axis inductance, Lq<br><br>Number of pole pairs, PolePairs | Motor Parameters |

### Current Controller

**`Bandwidth of the current regulator, EV_current` — Bandwidth**
`200` (default) | `scalar`

Derived current regulator bandwidth, in Hz.

#### Dependencies

This table summarizes the parameter dependencies.

| Parameter | Used to Derive | |
|---|---|---|
| | Parameter | Tab |
| Bandwidth of the current regulator, EV_current | D-axis proportional gain, Kp_d<br><br>Q-axis proportional gain, Kp_q<br><br>D and Q axis proportional gain, Ki | Current Controller |

**`Sample time for the torque control, Tst` — Time**
`5e-5` (default) | `scalar`

Derived torque control sample time, in s.

**Dependencies**

This table summarizes the parameter dependencies.

| Parameter | Used to Derive | |
|---|---|---|
| | Parameter | Tab |
| Sample time for the torque control, Tst | Speed regulation time constant, Ksf | Speed Controller |

**`Calculate Current Regulator Gains` — Derive parameters**
button

Click to derive parameters.

**Dependencies**

On the **Current Controller** tab, when you select **Calculate Current Regulator Gains**, the block calculates derived parameters. The table summarizes the derived parameter dependencies on other block parameters.

| Derived Parameter on Current Controller tab | Dependency | |
|---|---|---|
| | Parameter | Tab |
| D-axis proportional gain, Kp_d | Bandwidth of the current regulator, EV_current | Current Controller |
| Q-axis proportional gain, Kp_q<br><br>D and Q axis integral gain, Ki | Stator resistance, Rs | Motor Parameters |

**`D-axis proportional gain, Kp_d` — Derived**
`2.1363` (default) | `scalar`

Derived d-axis proportional gain, in V/A.

**Dependencies**

This table summarizes the parameter dependencies.

| Parameter | Dependency | |
|---|---|---|
| | Parameter | Tab |
| D-axis proportional gain, Kp_d | Bandwidth of the current regulator, EV_current | Current Controller |

**`Q-axis proportional gain, Kp_q` — Derived**
`4.0212` (default) | `scalar`

Derived q-axis proportional gain, in V/A.

#### Dependencies

This table summarizes the parameter dependencies.

| Parameter | Dependency | |
|---|---|---|
| | Parameter | Tab |
| Q-axis proportional gain, Kp_q | Bandwidth of the current regulator, EV_current | Current Controller |

**`D and Q axis integral gain, Ki` — Derived**
`25.1327` (default) | `scalar`

Derived d- and q- axis integral gains, in V/A·s.

#### Dependencies

This table summarizes the parameter dependencies.

| Parameter | Dependency | |
|---|---|---|
| | Parameter | Tab |
| D and Q axis integral gain, Ki | Stator resistance, Rs | Motor Parameters |

### Speed Controller

**`Bandwidth of the motion controller, EV_motion` — Bandwidth**
`[20, 4, 0.8]` (default) | `vector`

Motion controller bandwidth, in Hz. Set the first element of the vector to the desired cutoff frequency. Set the second and third elements of the vector to the higher-order cut off frequencies. You can set the value of the next element to `1/5` the value of the previous element. For example, if the desired cutoff frequency is `20` Hz, specify `[20 4 0.8]`.

#### Dependencies

The parameter is enabled when the **Control Type** parameter is set to `Speed Control`.

| Parameter | Used to Derive | |
|---|---|---|
| | Parameter | Tab |
| Bandwidth of the motion controller, EV_motion | Proportional gain, ba<br><br>Angular gain, Ksa<br><br>Rotational gain, Kisa | Speed Controller |

### `Bandwidth of the state filter, EV_sf` — Bandwidth

`200` (default) | `scalar`

State filter bandwidth, in Hz.

#### Dependencies

The parameter is enabled when the **Control Type** parameter is set to `Speed Control`.

| Parameter | Used to Derive | |
|---|---|---|
| | Parameter | Tab |
| Bandwidth of the state filter, EV_sf | Speed regulation time constant, Ksf | Speed Controller |

### `Calculate Speed Regulator Gains` — Derive parameters

button

Click to derive parameters.

#### Dependencies

On the **Speed Controller** tab, when you select **Calculate Speed Regulator Gains**, the block calculates derived parameters. The table summarizes the derived parameters that depend on other block parameters.

| Derived Parameter on Speed Controller tab | | Depends On | |
|---|---|---|---|
| | | Parameter | Tab |
| Proportional gain, ba | $b_a = \frac{J_p - J_p p_1 p_2 p_3}{T_{sm}}$ | Bandwidth of the motion controller, EV_motion<br><br>Bandwidth of the state filter, EV_sf | Speed Controller |
| Angular gain, Ksa | $K_{sa} = \frac{J_p(p_1p_2 + p_2p_3 + p_3p_1) - 3J_p + 2b_aT_{sm}}{T_{sm}^2}$ | Sample time for the torque control, Tst | Current Controller |
| Rotational gain, Kisa | $K_{isa} = \frac{-J_p(p_1 + p_2 + p_3) + 3J_p - b_aT_{sm} - K_{sa}T_{sm}^2}{T_{sm}^3}$ | Physical inertia, viscous damping, static friction, Mechanical | Motor Parameters |

| Derived Parameter on Speed Controller tab | | Depends On | |
|---|---|---|---|
| | | **Parameter** | **Tab** |
| **Speed regulation time constant, Ksf** | $K_{sf} = \frac{1 - \exp\left(-T_{sm} 2\pi EV_{sf}\right)}{T_{sm}}$ | | |
| **Inertia compensation, Jcomp** | $J_{comp} = J_p$ | **Physical inertia, viscous damping, static friction, Mechanical** | **Motor Parameters** |
| **Viscous damping compensation, Fv** | $F_v$ | | |
| **Static friction, Fs** | $F_s$ | | |

The equations use these variables.

| | |
|---|---|
| $P$ | Motor pole pairs |
| $b_a$ | Speed regulator proportional gain |
| $K_{sa}$ | Speed regulator integral gain |
| $K_{isa}$ | Speed regulator double integral gain |
| $K_{sf}$ | Speed regulator time constant |
| $J_p$ | Motor inertia |
| $EV_{sf}$ | State filter bandwidth |
| $EV_{motion}$ | Motion controller bandwidth |

### `Proportional gain, ba` — Derived

`3.7477` (default) | `scalar`

Derived proportional gain, in N·m/(rad/s).

#### Dependencies

This table summarizes the parameter dependencies.

| Parameter | Dependency | |
|---|---|---|
| | **Parameter** | **Tab** |
| **Proportional gain, ba** | **Physical inertia, viscous damping, static friction, Mechanical** | **Motor Parameters** |
| | **Bandwidth of the motion controller, EV_motion** | **Speed Controller** |

### `Angular gain, Ksa` — Derived

`94.0877` (default) | `scalar`

Derived angular gain, in N·m/rad.

#### Dependencies

This table summarizes the parameter dependencies.

| Parameter | Dependency | |
|---|---|---|
| | Parameter | Tab |
| Angular gain, Ksa | Physical inertia, viscous damping, static friction, Mechanical | Motor Parameters |
| | Bandwidth of the motion controller, EV_motion | Speed Controller |

**`Rotational gain, Kisa` — Derived**
`381.7822` (default) | `scalar`

Derived rotational gain, in N·m/(rad*s).

#### Dependencies

This table summarizes the parameter dependencies.

| Parameter | Dependency | |
|---|---|---|
| | Parameter | Tab |
| Rotational gain, Kisa | Physical inertia, viscous damping, static friction, Mechanical | Motor Parameters |
| | Bandwidth of the motion controller, EV_motion | Speed Controller |

**`Speed regulation time constant, Ksf` — Derived**
`1217.9727` (default) | `scalar`

Derived speed regulation time constant, in 1/s.

#### Dependencies

This table summarizes the parameter dependencies.

| Parameter | Dependency | |
|---|---|---|
| | Parameter | Tab |
| Speed regulation time constant, Ksf | Sample time for the torque control, Tst | Current Controller |
| | Bandwidth of the state filter, EV_sf | Speed Controller |

**`Inertia compensation, Jcomp` — Derived**
`0.025` (default) | `scalar`

Derived inertia compensation, in kg·m^2.

#### Dependencies

This table summarizes the parameter dependencies.

| Parameter | Dependency | |
|---|---|---|
| | Parameter | Tab |
| **Inertia compensation, Jcomp** | **Physical inertia, viscous damping, static friction, Mechanical** | **Motor Parameters** |

### `Viscous damping compensation, Fv` — Derived

`0` (default) | `scalar`

#### Dependencies

This table summarizes the parameter dependencies.

| Parameter | Dependency | |
|---|---|---|
| | Parameter | Tab |
| **Viscous damping compensation, Fv** | **Physical inertia, viscous damping, static friction, Mechanical** | **Motor Parameters** |

### `Static friction, Fs` — Derived

`0` (default) | `scalar`

Derived static friction, in N·m/(rad/s).

#### Dependencies

This table summarizes the parameter dependencies.

| Parameter | Dependency | |
|---|---|---|
| | Parameter | Tab |
| **Static friction, Fs** | **Physical inertia, viscous damping, static friction, Mechanical** | **Motor Parameters** |

#### Electrical Losses

### `Parameterize losses by` — Select type

`Single efficiency measurement` (default) | `Tabulated loss data` | `Tabulated efficiency data`

| Setting | Block Implementation |
|---|---|
| `Single efficiency measurement` | Electrical loss calculated using a constant value for inverter efficiency. |
| `Tabulated loss data` | Electrical loss calculated as a function of motor speeds and load torques. |

| Setting | Block Implementation |
|---|---|
| `Tabulated efficiency data` | Electrical loss calculated using inverter efficiency that is a function of motor speeds and load torques.<br>• Converts the efficiency values you provide into losses and uses the tabulated losses for simulation.<br>• Ignores efficiency values you provide for zero speed or zero torque. Losses are assumed zero when either torque or speed is zero.<br>• Uses linear interpolation to determine losses. Provide tabulated data for low speeds and low torques, as required, to get the desired level of accuracy for lower power conditions.<br>• Does not extrapolate loss values for speed and torque magnitudes that exceed the range of the table. |

For best practice, use `Tabulated loss data` instead of `Tabulated efficiency data`:

- Efficiency becomes ill defined for zero speed or zero torque.
- You can account for fixed losses that are still present for zero speed or torque.

**`Overall inverter efficiency, eff` — Constant**
98 (default) | `scalar`

Overall inverter efficiency, *Eff*, in %.

**Dependencies**

To enable this parameter, for **Parameterize losses by**, select `Tabulated loss data`.

**`Vector of speeds (w) for tabulated loss, w_loss_bp` — Breakpoints**
`[0 200 400 600 800 1000]` (default) | 1-by-`M` vector

Speed breakpoints for lookup table when calculating losses, in rad/s.

**Dependencies**

To enable this parameter, for **Parameterize losses by**, select `Tabulated loss data`.

**`Vector of torques (T) for tabulated loss, T_loss_bp` — Breakpoints**
`[0 25 50 75 100]` (default) | 1-by-`N` vector

Torque breakpoints for lookup table when calculating losses, in N·m.

**Dependencies**

To enable this parameter, for **Parameterize losses by**, select `Tabulated loss data`.

**`Corresponding losses, losses_table` — Table**
`[100 100 100 100 100;100 150 200 250 300;100 200 300 400 500;100 250 400 550 700;100 300 500 700 900;100 350 600 850 1100]` (default) | `M`-by-`N` array

Array of values for electrical losses as a function of `M` speeds and `N` torques, in W. Each value specifies the losses for a specific combination of speed and torque. The matrix size must match the dimensions defined by the speed and torque vectors.

**Dependencies**

To enable this parameter, for **Parameterize losses by**, select `Tabulated loss data`.

**`Vector of speeds (w) for tabulated efficiency, w_eff_bp` — Breakpoints**
`[200 400 600 800 1000]` (default) | 1-by-M vector

Speed breakpoints for lookup table when calculating efficiency, in rad/s.

**Dependencies**

To enable this parameter, for **Parameterize losses by**, select `Tabulated efficiency data`.

**`Vector of torques (T) for tabulated efficiency, T_eff_bp` — Breakpoints**
`[25 50 75 100]` (default) | 1-by-N vector

Torque breakpoints for lookup table when calculating efficiency, in N·m.

**Dependencies**

To enable this parameter, for **Parameterize losses by**, select `Tabulated efficiency data`.

**`Corresponding efficiency, efficiency_table` — Table**
`[96.2 98.1 98.7 99;98.1 99 99.4 99.5;98.7 99.4 99.6 99.7;99 99.5 99.7 99.8;99.2 99.6 99.7 99.8]` (default) | M-by-N array

Array of efficiency as a function of `M` speeds and `N` torque, in %. Each value specifies the efficiency for a specific combination of speed and torque. The matrix size must match the dimensions defined by the speed and torque vectors.

The block ignores efficiency values for zero speed or zero torque. Losses are zero when either torque or speed is zero. The block uses linear interpolation.

To get the desired level of accuracy for lower power conditions, you can provide tabulated data for low speeds and low torques.

**Dependencies**

To enable this parameter, for **Parameterize losses by**, select `Tabulated efficiency data`.

## References

[1] Lorenz, Robert D., Thomas Lipo, and Donald W. Novotny. “Motion control with induction motors.” *Proceedings of the IEEE*, Vol. 82, Issue 8, August 1994, pp. 1215–1240.

[2] Morimoto, Shigeo, Masayuka Sanada, and Yoji Takeda. “Wide-speed operation of interior permanent magnet synchronous motors with high-performance current regulator.” *IEEE Transactions on Industry Applications*, Vol. 30, Issue 4, July/August 1994, pp. 920–926.

[3] Li, Muyang. “Flux-Weakening Control for Permanent-Magnet Synchronous Motors Based on Z-Source Inverters.” Master’s Thesis, Marquette University, e-Publications@Marquette, Fall 2014.

[4] Briz, Fernando, Michael W. Degner, and Robert D. Lorenz. "Analysis and design of current regulators using complex vectors." *IEEE Transactions on Industry Applications*, Vol. 36, Issue 3, May/June 2000, pp. 817–825.

[5] Briz, Fernando, et al. "Current and flux regulation in field-weakening operation [of induction motors]."*IEEE Transactions on Industry Applications*, Vol. 37, Issue 1, Jan/Feb 2001, pp. 42–50.

## Extended Capabilities

### C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

## See Also

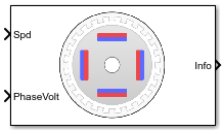
Flux-Based PM Controller | IM Controller | Interior PMSM | Surface Mount PM Controller

### Introduced in R2017a

# Flux-Based PMSM

Flux-based permanent magnet synchronous motor

**Library:** Powertrain Blockset / Propulsion / Electric Motors and Inverters




## Description

The Flux-Based PMSM block implements a flux-based three-phase permanent magnet synchronous motor (PMSM) with a tabular-based electromotive force. The block uses the three-phase input voltages to regulate the individual phase currents, allowing control of the motor torque or speed.

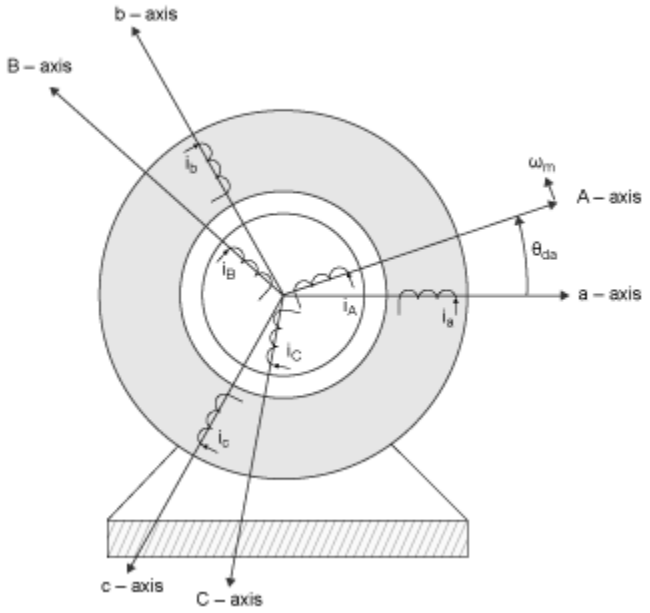
Flux-based motor models take into account magnetic saturation and iron losses. To calculate the magnetic saturation and iron loss, the Flux-Based PMSM block uses the inverse of the flux linkages. To obtain the block parameters, you can use finite-element analysis (FEA) or measure phase voltages using a dynamometer.

By default, the block sets the **Simulation Type** parameter to `Continuous` to use a continuous sample time during simulation. If you want to generate code for fixed-step double- and single-precision targets, considering setting the parameter to `Discrete`. Then specify a **Sample Time, Ts** parameter.

To enable power loss calculations suitable for code generation targets that limit memory, select **Enable memory optimized 2D LUT**.

### Three-Phase Sinusoidal Model Electrical System

The block implements equations that are expressed in a stationary rotor reference (dq) frame. The *d*-axis aligns with the *a*-axis. All quantities in the rotor reference frame are referred to the stator.




The block uses these equations.

| Calculation | Equation |
|---|---|
| $q$- and $d$-axis voltage | $v_d = \frac{d\psi_d}{dt} + R_s i_d - \omega_e \psi_q$<br>$v_q = \frac{d\psi_q}{dt} + R_s i_q + \omega_e \psi_d$ |
| $q$- and $d$-axis current | $i_d = f(\psi_d, \psi_q)$<br>$i_q = g(\psi_d, \psi_q)$ |
| Electromechanical torque | $T_e = 1.5P[\psi_d i_q - \psi_q i_d]$ |

The equations use these variables.

| | |
|---|---|
| $\omega_m$ | Rotor mechanical speed |
| $\omega_e$ | Rotor electrical speed |
| $\Theta_{da}$ | dq stator electrical angle with respect to the rotor a-axis |
| $R_s$, $R_r$ | Resistance of the stator and rotor windings, respectively |
| $i_q$, $i_d$ | $q$- and $d$-axis current, respectively |
| $v_q$, $v_d$ | $q$- and $d$-axis voltage, respectively |
| $\Psi_q$, $\Psi_d$ | $q$- and $d$-axis magnet flux, respectively |
| $P$ | Number of pole pairs |
| $T_e$ | Electromagnetic torque |

### Transforms

To calculate the voltages and currents in balanced three-phase ($a$, $b$) quantities, quadrature two-phase ($\alpha$, $\beta$) quantities, and rotating ($d$, $q$) reference frames, the block uses the Clarke and Park Transforms.

In the transform equations.

$$\omega_e = P\omega_m$$
$$\frac{d\theta_e}{dt} = \omega_e$$

| Transform | Description | Equations |
|---|---|---|
| Clarke | Converts balanced three-phase quantities ($a$, $b$) into balanced two-phase quadrature quantities ($\alpha$, $\beta$). | $x_\alpha = \frac{2}{3}x_a - \frac{1}{3}x_b - \frac{1}{3}x_c$<br>$x_\beta = \frac{\sqrt{3}}{2}x_b - \frac{\sqrt{3}}{2}x_c$ |
| Park | Converts balanced two-phase orthogonal stationary quantities ($\alpha$, $\beta$) into an orthogonal rotating reference frame ($d$, $q$). | $x_d = x_\alpha\cos\theta_e + x_\beta\sin\theta_e$<br>$x_q = -x_\alpha\sin\theta_e + x_\beta\cos\theta_e$ |
| Inverse Clarke | Converts balanced two-phase quadrature quantities ($\alpha$, $\beta$) into balanced three-phase quantities ($a$, $b$). | $x_a = x_a$<br>$x_b = -\frac{1}{2}x_\alpha + \frac{\sqrt{3}}{2}x_\beta$<br>$x_c = -\frac{1}{2}x_\alpha - \frac{\sqrt{3}}{2}x_\beta$ |
| Inverse Park | Converts an orthogonal rotating reference frame ($d$, $q$) into balanced two-phase orthogonal stationary quantities ($\alpha$, $\beta$). | $x_\alpha = x_d\cos\theta_e - x_q\sin\theta_e$<br>$x_\beta = x_d\sin\theta_e + x_q\cos\theta_e$ |

The transforms use these variables.

| | |
|---|---|
| $\omega_m$ | Rotor mechanical speed |
| $P$ | Motor pole pairs |
| $\omega_e$ | Rotor electrical speed |
| $\Theta_e$ | Rotor electrical angle |
| $x$ | Phase current or voltage |

### Mechanical System

The rotor angular velocity is given by:

$$\frac{d}{dt}\omega_m = \frac{1}{J}\left(T_e - T_f - F\omega_m - T_m\right)$$
$$\frac{d\theta_m}{dt} = \omega_m$$

The equations use these variables.

| | |
|---|---|
| $J$ | Combined inertia of rotor and load |
| $F$ | Combined viscous friction of rotor and load |
| $\theta_m$ | Rotor mechanical angular position |
| $T_m$ | Rotor shaft torque |
| $T_e$ | Electromagnetic torque |
| $T_f$ | Combined rotor and load friction torque |
| $\omega_m$ | Rotor mechanical speed |

### Power Accounting

For the power accounting, the block implements these equations.

| Bus Signal | | | Description | Variable | Equations |
|---|---|---|---|---|---|
| `PwrInfo` | `PwrTrnsfrd` — Power transferred between blocks<br>• Positive signals indicate flow into block<br>• Negative signals indicate flow out of block | `PwrMtr` | Mechanical power | $P_{mot}$ | $P_{mot} = -\omega_m T_e$ |
| | | `PwrBus` | Electrical power | $P_{bus}$ | $P_{bus} = v_{an}i_a + v_{bn}i_b + v_{cn}i_c$ |
| | `PwrNotTrnsfrd` — Power crossing the block boundary, but not transferred<br>• Positive signals indicate an input<br>• Negative signals indicate a loss | `PwrElecLoss` | Resistive power loss | $P_{elec}$ | $P_{elec} = -\frac{3}{2}(R_s i_{sd}^2 + R_s i_{sq}^2)$ |
| | | `PwrMechLoss` | Mechanical power loss | $P_{mech}$ | When **Port Configuration** is set to `Torque`:<br>$P_{mech} = -\left(\omega_m^2 F + \lvert\omega_m\rvert T_f\right)$<br>When **Port Configuration** is set to `Speed`:<br>$P_{mech} = 0$ |
| | `PwrStored` — Stored energy rate of change<br>• Positive signals indicate an increase<br>• Negative signals indicate a decrease | `PwrMtrStored` | Stored motor power | $P_{str}$ | $P_{str} = P_{bus} + P_{mot} + P_{elec} + P_{mech}$ |

The equations use these variables.

| | |
|---|---|
| $R_s$ | Stator resistance |

| | |
|---|---|
| $i_a$, $i_b$, $i_c$ | Stator phase a, b, and c current |
| $i_{sq}$, $i_{sd}$ | Stator q- and d-axis currents |
| $v_{an}$, $v_{bn}$, $v_{cn}$ | Stator phase a, b, and c voltage |
| $\omega_m$ | Angular mechanical velocity of the rotor |
| $F$ | Combined motor and load viscous damping |
| $T_e$ | Electromagnetic torque |
| $T_f$ | Combined motor and load friction torque |

### Lookup Table Memory Optimization

The data for the **Corresponding d-axis current, id** and **Corresponding q-axis current, iq** lookup tables are functions of the *d*- and *q*-axis flux.

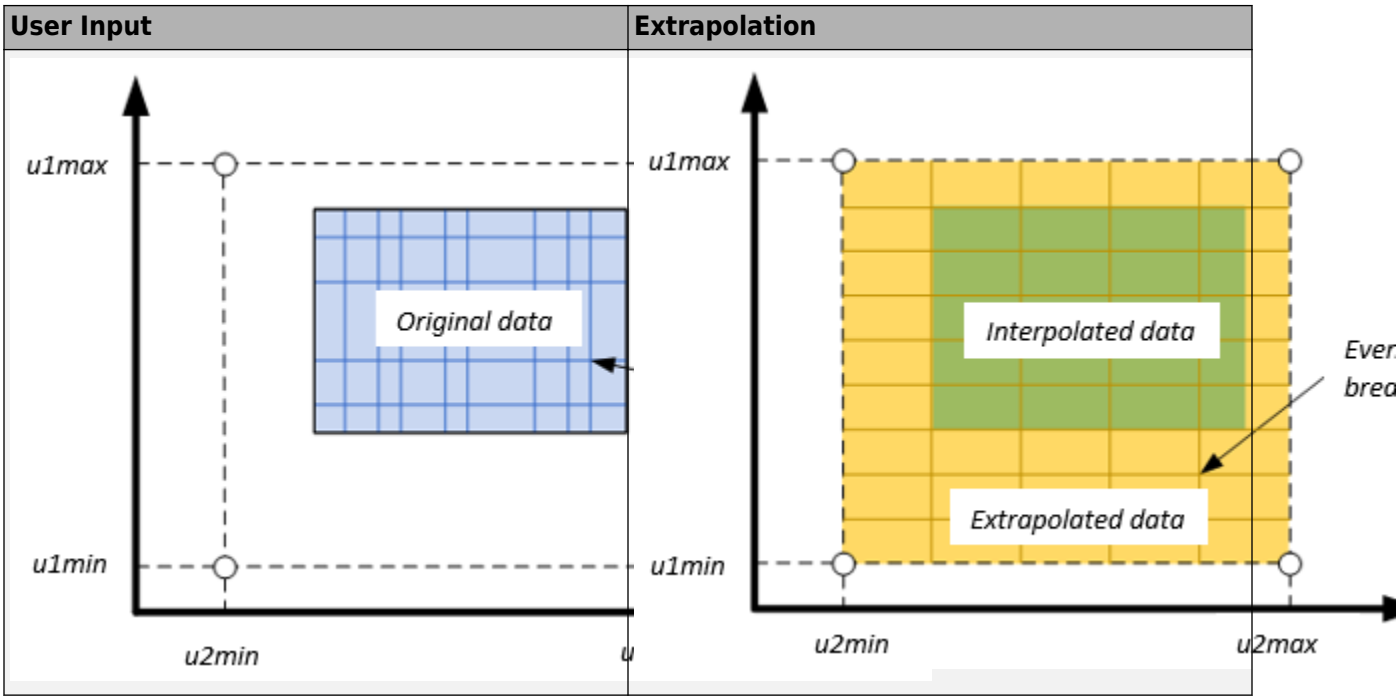
To enable current calculations suitable for code generation targets that limit memory, select **Enable memory optimized 2D LUT**. The block uses linear interpolation to optimize the current lookup table values for code generation. This table summarizes the optimization implementation.

| Use Case | Implementation |
|---|---|
| *d*- and *q*-axis flux aligns with the lookup table breakpoint values. | Memory-optimized current is current lookup table value at intersection of flux values. |
| *d*- and *q*-axis flux does not align with the lookup table breakpoint values, but is within range. | Memory-optimized current is linear interpolation between corresponding flux values. |
| *d*- and *q*-axis flux does not align with the lookup table breakpoint values, and is out of range. | Cannot compute an memory-optimized current. Block uses extrapolated data. |

#### Extrapolation

The lookup tables optimized for code generation do not support extrapolation for data that is out of range. However, you can include pre-calculated extrapolation values in the power loss lookup table by selecting **Specify Extrapolation**.

The block uses the endpoint parameters to resize the table data.




## Ports

### Input

#### `LdTrq` — Rotor shaft torque

`scalar`

Rotor shaft input torque, $T_m$, in N·m.

**Dependencies**

To create this port, select `Torque` for the **Port Configuration** parameter.

#### `Spd` — Rotor shaft speed

`scalar`

Angular velocity of the rotor, $\omega_m$, in rad/s.

**Dependencies**

To create this port, select `Speed` for the **Port Configuration** parameter.

#### `PhaseVolt` — Stator terminal voltages

1-by-3 array

Stator terminal voltages, $V_a$, $V_b$, and $V_c$, in V.

**Dependencies**

To create this port, select `Speed` or `Torque` for the **Port Configuration** parameter.

### Output

#### `Info` — Bus signal

bus

The bus signal contains these block calculations.

| Signal | | | Description | Variable | Units |
|---|---|---|---|---|---|
| `IaStator` | | | Stator phase current A | $i_a$ | A |
| `IbStator` | | | Stator phase current B | $i_b$ | A |
| `IcStator` | | | Stator phase current C | $i_c$ | A |
| `IdSync` | | | Direct axis current | $i_d$ | A |
| `IqSync` | | | Quadrature axis current | $i_q$ | A |
| `VdSync` | | | Direct axis voltage | $v_d$ | V |
| `VqSync` | | | Quadrature axis voltage | $v_q$ | V |
| `MtrSpd` | | | Angular mechanical velocity of the rotor | $\omega_m$ | rad/s |
| `MtrPos` | | | Rotor mechanical angular position | $\theta_m$ | rad |
| `MtrTrq` | | | Electromagnetic torque | $T_e$ | N·m |
| `PwrInfo` | `PwrTrnsfrd` | `PwrMtr` | Mechanical power | $P_{mot}$ | W |
| | | `PwrBus` | Electrical power | $P_{bus}$ | W |
| | `PwrNotTrnsfrd` | `PwrElecLoss` | Resistive power loss | $P_{elec}$ | W |
| | | `PwrMechLoss` | Mechanical power loss | $P_{mech}$ | W |
| | `PwrStored` | `PwrMtrStored` | Stored motor power | $P_{str}$ | W |

#### `PhaseCurr` — Phase a, b, c current

1-by-3 array

Phase a, b, c current, $i_a$, $i_b$, and $i_c$, in A.

#### `MtrTrq` — Motor torque

`scalar`

Motor torque, $T_{mtr}$, in N·m.

##### Dependencies

To create this port, select `Speed` for the **Port configuration** parameter.

#### `MtrSpd` — Motor speed

`scalar`

Angular speed of the motor, $\omega_{mtr}$, in rad/s.

##### Dependencies

To create this port, select `Torque` for the **Port configuration** parameter.

## Parameters

#### Block Options

**`Simulation type` — Select simulation type**
`Continuous` (default) | `Discrete`

By default, the block uses a continuous sample time during simulation. If you want to generate code for single-precision targets, considering setting the parameter to `Discrete`.

#### Dependencies

Setting **Simulation Type** to `Discrete` creates the **Sample Time, Ts** parameter.

**`Sample time, Ts` — Sample time for discrete integration**
`0.001` (default) | `scalar`

Integration sample time for discrete simulation, in s.

#### Dependencies

Setting **Simulation Type** to `Discrete` creates the **Sample Time, Ts** parameter.

**`Port Configuration` — Select port configuration**
`Torque` (default) | `Speed`

This table summarizes the port configurations.

| Port Configuration | Creates Input Port | Creates Output Port |
|---|---|---|
| `Torque` | `LdTrq` | `MtrSpd` |
| `Speed` | `Spd` | `MtrTrq` |

**`Enable memory optimized 2D LUT` — Selection**
`off` (default) | `on`

Enable generation of optimized lookup tables, suitable code generation targets that limit memory.

**`Vector of d-axis flux, flux_d` — Flux breakpoints**
1-by-`M` vector

*d*-axis flux, $\Psi_d$, breakpoints, in Wb.

**`Resample storage size for flux_d, n1` — Flux bit size**
2 (default) | 4 | 8 | 16 | 32 | 64 | 128 | 256

Flux breakpoint storage size, *n1*, dimensionless. The block resamples the **Corresponding d-axis current, id** and **Corresponding q-axis current, iq** data based on the storage size.

#### Dependencies

To create this parameter, select **Enable memory optimized 2D LUT**.

**`Vector of q-axis flux, flux_q` — Flux breakpoints**
1-by-`N` vector

*q*-axis flux, $\Psi_q$, breakpoints, in Wb.

**`Resample storage size for flux_q, n2` — Flux bit size**
`2` (default) | `4` | `8` | `16` | `32` | `64` | `128` | `256`

Flux breakpoint storage size, *n2*, dimensionless. The block resamples the **Corresponding d-axis current, id** and **Corresponding q-axis current, iq** data based on the storage size.

**Dependencies**

To create this parameter, select **Enable memory optimized 2D LUT**.

**`Corresponding d-axis current, id` — 2D lookup table**
`M`-by-`N` array

Array of values for *d*-axis current, $i_d$, as a function of `M` *d*-fluxes, $\Psi_d$, and `N` *q*-fluxes, $\Psi_q$, in A. Each value specifies the current for a specific combination of *d*- and *q*-axis flux. The array size must match the dimensions defined by the flux vectors.

If you set **Enable memory optimized 2D LUT**, the block converts the data to single precision.

**`Corresponding q-axis current, iq` — 2D lookup table**
`M`-by-`N` array

Array of values for *q*-axis current, $i_d$, as a function of `M` *d*-fluxes, $\Psi_d$, and `N` *q*-fluxes, $\Psi_q$, in A. Each value specifies the current for a specific combination of *d*- and *q*-axis flux. The array size must match the dimensions defined by the flux vectors.

If you set **Enable memory optimized 2D LUT**, the block converts the data to single precision.

**`flux_d max endpoint, u1max` — Flux breakpoint**
`0.22457` (default) | `scalar`

Flux breakpoint maximum extrapolation endpoint, *u1max*, in Wb.

**Dependencies**

To create this parameter, select **Enable memory optimized 2D LUT** and **Specify Extrapolation**.

**`flux_d min endpoint, u1min` — Flux breakpoint**
`-0.22607` (default) | `scalar`

Flux breakpoint minimum extrapolation endpoint, *u1min*, in Wb.

**Dependencies**

To create this parameter, select **Enable memory optimized 2D LUT** and **Specify Extrapolation**.

**`flux_q max endpoint, u2max` — Flux breakpoint**
`0.42959` (default) | `scalar`

Flux breakpoint maximum extrapolation endpoint, *u2max*, in Wb.

**Dependencies**

To create this parameter, select **Enable memory optimized 2D LUT** and **Specify Extrapolation**.

**`flux_q min endpoint, u2min` — Flux breakpoint**
`-0.4296` (default) | `scalar`

Flux breakpoint minimum extrapolation endpoint, *u2min*, in Wb.

**Dependencies**

To create this parameter, select **Enable memory optimized 2D LUT** and **Specify Extrapolation**.

**`Stator phase resistance, Rs` — Resistance**
`0.02` (default) | `scalar`

Stator phase resistance, $R_s$, in ohm.

**`Number of pole pairs, P` — Pole pairs**
`4` (default) | `scalar`

Motor pole pairs, $P$.

**`Initial flux, fluxdq0` — Flux**
`[0 0]` (default) | `vector`

Initial $d$- and $q$-axis flux, $\Psi_{q0}$ and $\Psi_{d0}$, in Wb.

**`Initial mechanical position, theta_init` — Angle**
`0` (default) | `scalar`

Initial rotor angular position, $\theta_{m0}$, in rad.

**`Initial mechanical speed, omega_init` — Speed**
`0` (default) | `scalar`

Initial angular velocity of the rotor, $\omega_{m0}$, in rad/s.

**Dependencies**

To enable this parameter, select the `Torque` configuration parameter.

**`Physical inertia, viscous damping, and static friction, mechanical` — Inertia, damping, friction**
`[0.0027, 4.924e-4, 0]` (default) | `vector`

Mechanical properties of the rotor:

- Inertia, $J$, in kgm^2
- Viscous damping, $F$, in N·m/(rad/s)
- Static friction, $T_f$, in N·m

**Dependencies**

To enable this parameter, select the `Torque` configuration parameter.

## References

[1] Hu, Dakai, Yazan Alsmadi, and Longya Xu. “High fidelity nonlinear IPM modeling based on measured stator winding flux linkage.” *IEEE Transactions on Industry Applications*, Vol. 51, No. 4, July/August 2015.

[2] Chen, Xiao, Jiabin Wang, Bhaskar Sen, Panagiotis Lasari, Tianfu Sun. “A High-Fidelity and Computationally Efficient Model for Interior Permanent-Magnet Machines Considering the Magnetic Saturation, Spatial Harmonics, and Iron Loss Effect.” *IEEE Transactions on Industrial Electronics*, Vol. 62, No. 7, July 2015.

[3] Ottosson, J., M. Alakula. “A compact field weakening controller implementation.” *International Symposium on Power Electronics, Electrical Drives, Automation and Motion*, July, 2006.

## Extended Capabilities

### C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

## See Also

Flux-Based PM Controller | Induction Motor | Interior PMSM | Mapped Motor | Surface Mount PMSM

### Topics

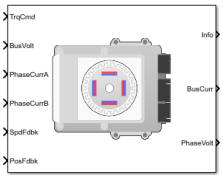
“Generate Parameters for Flux-Based Blocks”

### Introduced in R2017b

# Flux-Based PM Controller

Controller for a flux-based permanent magnet synchronous motor

**Library:** Powertrain Blockset / Propulsion / Electric Motor Controllers

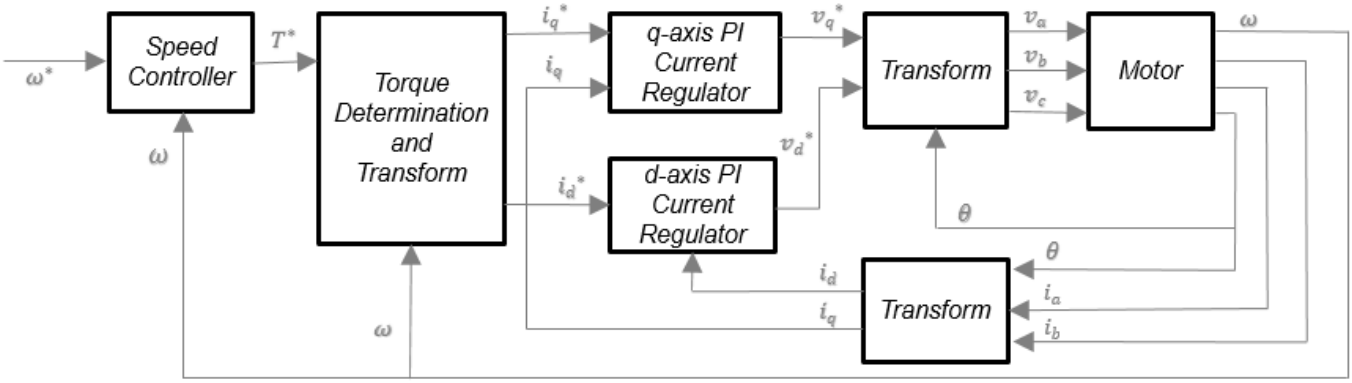



## Description

The Flux Based PM Controller block implements a flux-based, field-oriented controller for an interior permanent magnet synchronous motor (PMSM) with an optional outer-loop speed controller. The internal torque control implements strategies for achieving maximum torque per ampere (MTPA) and weakening the magnetic flux. You can specify either the speed or torque control type.

The Flux Based PM Controller implements equations for speed control, torque determination, regulators, transforms, and motors.

The figure illustrates the information flow in the block.




The block implements equations using these variables.

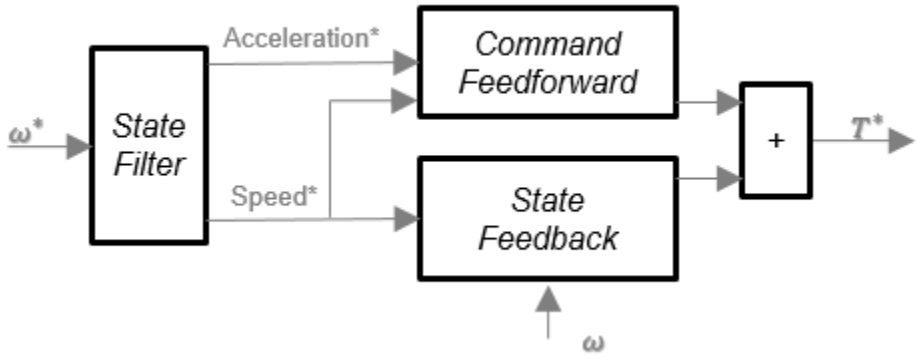
| | |
|---|---|
| $\omega$ | Rotor speed |
| $\omega$* | Rotor speed command |
| $T$* | Torque command |
| $i_d$ | $d$-axis current |
| $i^*_d$ | $d$-axis current command |
| $i_q$ | $q$-axis current |
| $i^*_q$ | $q$-axis current command |

| | |
|---|---|
| $v_d$, | $d$-axis voltage |
| $v^*_d$ | $d$-axis voltage command |
| $v_q$ | $q$-axis voltage |
| $v^*_q$ | $q$-axis voltage command |
| $v_a$, $v_b$, $v_c$ | Stator phase a, b, c voltages |
| $i_a$, $i_b$, $i_c$ | Stator phase a, b, c currents |

### Speed Controller

To implement the speed controller, select the **Control Type** parameter `Speed Control`. If you select the **Control Type** parameter `Torque Control`, the block does not implement the speed controller.

The speed controller determines the torque command by implementing a state filter, and calculating the feedforward and feedback commands. If you do not implement the speed controller, input a torque command to the Flux Based PM Controller block.




#### State Filter

The state filter is a low-pass filter that generates the acceleration command based on the speed command. The discrete form of characteristic equation is given by:

$$z + K_{sf}T_{sm} - 1$$

The filter calculates the gain using this equation.

$$K_{sf} = \frac{1 - \exp\left(-T_{sm}2\pi EV_{sf}\right)}{T_{sm}}$$

The equations use these variables.

| | |
|---|---|
| $EV_{sf}$ | Bandwidth of the speed command filter |
| $T_{sm}$ | Motion controller sample time |
| $K_{sf}$ | Speed regulator time constant |

#### State Feedback

To generate the state feedback torque, the block uses the filtered speed error signal from the state filter. To filter the speed, the block uses a proportional integral (PI) controller.

$$T_{cmd} = Kp_{\omega}(\omega_m^* - \omega_m) + Ki_{\omega}\frac{zT_{sm}}{z-1}(\omega_m^* - \omega_m)$$

The equations use these variables.

| | |
|---|---|
| $\omega_m$ | Rotor speed |
| $\omega^*_m$ | Rotor speed command |
| $T_{cmd}$ | Torque command |
| $Kp_{\omega}$ | Speed regulator proportional gain |
| $Ki_{\omega}$ | Speed regulator integral gain |
| $T_{sm}$ | Speed regulator sample rate |

#### Command Feedforward

To generate the state feedforward torque, the block uses the filtered speed and acceleration from the state filter. Also, the feedforward torque calculation uses the inertia, viscous damping, and static friction. To achieve zero tracking error, the torque command is the sum of the feedforward and feedback torque commands.

The feedforward torque command uses this equation.

$$T_{cmd_ff} = J_p\dot{\omega}_m + F_v\omega_m + F_s\frac{\omega_m}{|\omega_m|}$$

where:

| | |
|---|---|
| $J_p$ | Rotor inertia |
| $T_{cmd_ff}$ | Torque command feedforward |
| $F_s$ | Static friction torque constant |
| $F_v$ | Viscous friction torque constant |
| $F_s$ | Static friction torque constant |
| $\omega_m$ | Rotor speed |

#### Current Command

The block uses lookup tables to determine the *d*-axis and *q*-axis current commands. The lookup tables are functions of mechanical speed and torque. To determine the lookup tables, you can use an external finite element analysis (FEA) models or dynamometer test results.

$$i_{dref} = f(|\omega_m|, |T_{ref}|)$$
$$i_{qref} = sign(T_{ref}) * f(|\omega_m|, |T_{ref}|)$$

The equations use these variables.

| | |
|---|---|
| $\omega_m$ | Rotor speed |
| $T_{ref}$ | Torque command |
| $i_{dref}$, $i_{qref}$ | *d*- and *q*-axis reference current, respectively |

#### Voltage Command

The block uses these equations to calculate the voltage in the motor reference frame.

$$v_d = \frac{d\psi_d}{dt} + R_s i_d - \omega_e \psi_q$$
$$v_q = \frac{d\psi_q}{dt} + R_s i_q + \omega_e \psi_d$$

$$\frac{d\psi_d}{dt} + R_s i_d = Kp_d(i_d^* - i_d) + Ki_d \frac{zT_{st}}{z-1}(i_d^* - i_d)$$
$$\frac{d\psi_q}{dt} + R_s i_q = Kp_q(i_q^* - i_q) + Ki_q \frac{zT_{st}}{z-1}(i_q^* - i_q)$$

$$v_d = Kp_i(i_d^* - i_d) + Ki_d \frac{zT_{st}}{z-1}(i_d^* - i_d) + \omega_e \psi_q$$
$$v_q = Kp_i(i_q^* - i_q) + Ki_q \frac{zT_{st}}{z-1}(i_q^* - i_q) - \omega_e \psi_d$$

$$\psi_q = f(i_d, i_q)$$
$$\psi_d = f(i_d, i_q)$$

The equations use these variables.

| | |
|---|---|
| $\omega_m$ | Rotor mechanical speed |
| $\omega_e$ | Rotor electrical speed |
| $R_s$, $R_r$ | Resistance of the stator and rotor windings, respectively |
| $i_q$, $i_d$ | $q$- and $d$-axis current, respectively |
| $v_q$, $v_d$ | $q$- and $d$-axis voltage, respectively |
| $\Psi_q$, $\Psi_d$ | $q$- and $d$-axis magnet flux, respectively |
| $T_{st}$ | Current regulator sample rate |
| $Ki_d$ , $Ki_q$ | $d$- and $q$- axis integral gain, respectively |
| $Kp_d$ , $Kp_q$ | $d$- and $q$- axis proportional gain, respectively |

### Transforms

To calculate the voltages and currents in balanced three-phase ($a$, $b$) quantities, quadrature two-phase ($\alpha$, $\beta$) quantities, and rotating ($d$, $q$) reference frames, the block uses the Clarke and Park Transforms.

In the transform equations.

$$\omega_e = P\omega_m$$
$$\frac{d\theta_e}{dt} = \omega_e$$

| Transform | Description | Equations |
|---|---|---|
| Clarke | Converts balanced three-phase quantities ($a$, $b$) into balanced two-phase quadrature quantities ($\alpha$, $\beta$). | $x_\alpha = \frac{2}{3}x_a - \frac{1}{3}x_b - \frac{1}{3}x_c$<br>$x_\beta = \frac{\sqrt{3}}{2}x_b - \frac{\sqrt{3}}{2}x_c$ |
| Park | Converts balanced two-phase orthogonal stationary quantities ($\alpha$, $\beta$) into an orthogonal rotating reference frame ($d$, $q$). | $x_d = x_\alpha\cos\theta_e + x_\beta\sin\theta_e$<br>$x_q = -x_\alpha\sin\theta_e + x_\beta\cos\theta_e$ |
| Inverse Clarke | Converts balanced two-phase quadrature quantities ($\alpha$, $\beta$) into balanced three-phase quantities ($a$, $b$). | $x_a = x_a$<br>$x_b = -\frac{1}{2}x_\alpha + \frac{\sqrt{3}}{2}x_\beta$<br>$x_c = -\frac{1}{2}x_\alpha - \frac{\sqrt{3}}{2}x_\beta$ |
| Inverse Park | Converts an orthogonal rotating reference frame ($d$, $q$) into balanced two-phase orthogonal stationary quantities ($\alpha$, $\beta$). | $x_\alpha = x_d\cos\theta_e - x_q\sin\theta_e$<br>$x_\beta = x_d\sin\theta_e + x_q\cos\theta_e$ |

The transforms use these variables.

| | |
|---|---|
| $\omega_m$ | Rotor speed |
| $P$ | Rotor pole pairs |
| $\omega_e$ | Rotor electrical speed |
| $\Theta_e$ | Rotor electrical angle |
| $x$ | Phase current or voltage |

**Motor**

The block uses the phase currents and phase voltages to estimate the DC bus current. Positive current indicates battery discharge. Negative current indicates battery charge.

The block uses these equations.

| | |
|---|---|
| Load power | $Ld_{Pwr} = v_a\ i_a + v_b\ i_b + v_c\ i_c$ |
| Source power | $Src_{Pwr} = Ld_{Pwr} + Pwr_{Loss}$ |
| DC bus current | $i_{bus} = \frac{Src_{Pwr}}{v_{bus}}$ |
| Estimated rotor torque | $T_e = 1.5P[\psi_d i_q - \psi_q i_d]$ |
| Power loss for single efficiency source to load | $Pwr_{Loss} = \frac{100 - Eff}{Eff} \cdot Ld_{Pwr}$ |
| Power loss for single efficiency load to source | $Pwr_{Loss} = \frac{100 - Eff}{100} \cdot \lvert Ld_{Pwr}\rvert$ |
| Power loss for tabulated efficiency | $Pwr_{Loss} = f(\omega_m, MtrTrq_{est})$ |

The equations use these variables.

| | |
|---|---|
| $v_a$, $v_b$, $v_c$ | Stator phase a, b, c voltages |
| $v_{bus}$ | Estimated DC bus voltage |
| $i_a$, $i_b$, $i_c$ | Stator phase a, b, c currents |
| $i_{bus}$ | Estimated DC bus current |
| $Eff$ | Overall inverter efficiency |
| $\omega_m$ | Rotor mechanical speed |
| $L_q$, $L_d$ | $q$- and $d$-axis winding inductance, respectively |
| $\Psi_q$, $\Psi_d$ | $q$- and $d$-axis magnet flux, respectively |
| $i_q$, $i_d$ | $q$- and $d$-axis current, respectively |
| $\lambda$ | Permanent magnet flux linkage |
| $P$ | Rotor pole pairs |

#### Electrical Losses

To specify the electrical losses, on the **Electrical Losses** tab, for **Parameterize losses by**, select one of these options.

| Setting | Block Implementation |
|---|---|
| `Single efficiency measurement` | Electrical loss calculated using a constant value for inverter efficiency. |
| `Tabulated loss data` | Electrical loss calculated as a function of motor speeds and load torques. |
| `Tabulated efficiency data` | Electrical loss calculated using inverter efficiency that is a function of motor speeds and load torques.<br>• Converts the efficiency values you provide into losses and uses the tabulated losses for simulation.<br>• Ignores efficiency values you provide for zero speed or zero torque. Losses are assumed zero when either torque or speed is zero.<br>• Uses linear interpolation to determine losses. Provide tabulated data for low speeds and low torques, as required, to get the desired level of accuracy for lower power conditions.<br>• Does not extrapolate loss values for speed and torque magnitudes that exceed the range of the table. |

For best practice, use `Tabulated loss data` instead of `Tabulated efficiency data`:

- Efficiency becomes ill defined for zero speed or zero torque.
- You can account for fixed losses that are still present for zero speed or torque.

## Ports

### Input

#### SpdReq — Rotor speed command

`scalar`

Rotor speed command, $\omega^*_m$, in rad/s.

#### Dependencies

To create this port, select `Speed Control` for the **Control Type** parameter.

### `TrqCmd` — Torque command

`scalar`

Torque command, $T^*$, in N·m.

#### Dependencies

To create this port, select `Torque Control` for the **Control Type** parameter.

### `BusVolt` — DC bus voltage

`scalar`

DC bus voltage, $v_{bus}$, in V.

### `PhaseCurrA` — Current

`scalar`

Stator current phase a, $i_a$, in A.

### `PhaseCurrB` — Current

`scalar`

Stator current phase b, $i_b$, in A.

### `SpdFdbk` — Rotor speed

`scalar`

Rotor speed, $\omega_m$, in rad/s.

### `PosFdbk` — Rotor electrical angle

`scalar`

Rotor electrical angle, $\Theta_m$, in rad.

## Output

### `Info` — Bus signal

bus

Bus signal containing these block calculations.

| Signal | Description | Units |
|---|---|---|
| `SrcPwr` | Source power | W |
| `LdPwr` | Load power | W |
| `PwrLoss` | Power loss | W |
| `MtrTrqEst` | Estimated motor torque | N·m |

### `BusCurr` — Bus current

`scalar`

Estimated DC bus current, $i_{bus}$, in A.

**`PhaseVolt` — Stator terminal voltages**
`array`

Stator terminal voltages, $V_a$, $V_b$, and $V_c$, in V.

## Parameters

### Block Options

**`Control Type` — Select control**
`Torque Control` (default) | `Speed Control`

If you select `Torque Control`, the block does not implement the speed controller.

This table summarizes the port configurations.

| Port Configuration | Creates Ports |
|---|---|
| `Speed Control` | `SpdReq` |
| `Torque Control` | `TrqCmd` |

### Motor Parameters

**Number of pole pairs, `PolePairs` — Poles**
4 (default) | `scalar`

Motor pole pairs, $P$.

**Vector of d-axis current breakpoints, `id_index` — Current**
`vector`

$d$-axis current, $i_{d_index}$, in A.

**Vector of q-axis current breakpoints, `iq_index` — current**
`vector`

$q$-axis current, $i_{q_index}$, in A.

**Corresponding d-axis flux, `lambda_d` — Flux**
`vector`

$d$-axis flux, $\lambda_d$, in Wb.

**Corresponding q-axis flux, `lambda_q` — Flux**
`vector`

$q$-axis flux, $\lambda_q$, in Wb.

### Current Controller

**Sample time for the torque control, `Tst` — Time**
`1e-4` (default) | `scalar`

Torque control sample time, $T_{st}$, in s.

**D-axis proportional gain, Kp_d — Gain**
2.4056 (default) | scalar

*d*-axis proportional gain, $Kp_d$, in V/A.

**Q-axis proportional gain, Kp_q — Gain**
2.4056 (default) | scalar

*q*-axis proportional gain, $Kp_q$, in V/A.

**D-axis integral gain, Ki_d — Gain**
192.45 (default) | scalar

*d*-axis integral gain, $Ki_d$, in V/A·s.

**Q-axis integral gain, Ki_q — Gain**
192.45 (default) | scalar

*q*- axis integral gain, $Ki_q$, in V/A·s.

**Vector of speed breakpoints, wpb — Breakpoints**
vector

Speed breakpoints, $\omega_{bp}$, in rad/s.

**Vector of torque breakpoints, tpb — Breakpoints**
vector

Torque breakpoints, $T_{bp}$, in N·m.

**Corresponding d-axis current reference, id_ref — Current**
vector

*d*-axis reference current, $i_{dref}$, in A.

**Corresponding q-axis current reference, iq_ref — Current**
vector

*q*-axis reference current, $i_{qref}$, in A.

#### Speed Controller

**Speed regulation time constant, Ksf — Time**
.1 (default) | scalar

Speed regulator time constant, $K_{sf}$, in 1/s.

#### Dependencies

To enable this parameter, for the **Control Type** parameter, select `Speed Control`.

**Proportional gain, Kp_w — Gain**
0.40475 (default) | scalar

Proportional gain, $Kp_\omega$, in N·m/(rad/s).

#### Dependencies

To enable this parameter, for the **Control Type** parameter, select `Speed Control`.

### `Integral gain, Ki_w` — Gain

`10.1615` (default) | `scalar`

Integral gain, $Ki_\omega$ N·m/rad.

#### Dependencies

To enable this parameter, for the **Control Type** parameter, select `Speed Control`.

### `Inertia compensation, Jcomp` — Inertia

`0.0027` (default) | `scalar`

Inertia compensation, in kg·m^2.

#### Dependencies

To enable this parameter, for the **Control Type** parameter, select `Speed Control`.

### `Static friction, Fs` — Friction

`0` (default) | `scalar`

Static friction, in N·m.

#### Dependencies

To enable this parameter, for the **Control Type** parameter, select `Speed Control`.

### `Viscous damping compensation, Fv` — Dampint

`0.0004924` (default) | `scalar`

Viscous damping compensation, in N·m/(rad/s).

#### Dependencies

To enable this parameter, for the **Control Type** parameter, select `Speed Control`.

#### Electrical Losses

### `Parameterize losses by` — Select type

`Single efficiency measurement` (default) | `Tabulated loss data` | `Tabulated efficiency data`

| Setting | Block Implementation |
|---|---|
| `Single efficiency measurement` | Electrical loss calculated using a constant value for inverter efficiency. |
| `Tabulated loss data` | Electrical loss calculated as a function of motor speeds and load torques. |

| Setting | Block Implementation |
|---|---|
| `Tabulated efficiency data` | Electrical loss calculated using inverter efficiency that is a function of motor speeds and load torques.<br>• Converts the efficiency values you provide into losses and uses the tabulated losses for simulation.<br>• Ignores efficiency values you provide for zero speed or zero torque. Losses are assumed zero when either torque or speed is zero.<br>• Uses linear interpolation to determine losses. Provide tabulated data for low speeds and low torques, as required, to get the desired level of accuracy for lower power conditions.<br>• Does not extrapolate loss values for speed and torque magnitudes that exceed the range of the table. |

For best practice, use `Tabulated loss data` instead of `Tabulated efficiency data`:

- Efficiency becomes ill defined for zero speed or zero torque.
- You can account for fixed losses that are still present for zero speed or torque.

**`Overall inverter efficiency, eff` — Constant**
98 (default) | `scalar`

Overall inverter efficiency, *Eff*, in %.

**Dependencies**

To enable this parameter, for **Parameterize losses by**, select `Tabulated loss data`.

**`Vector of speeds (w) for tabulated loss, w_loss_bp` — Breakpoints**
`[0 200 400 600 800 1000]` (default) | 1-by-`M` vector

Speed breakpoints for lookup table when calculating losses, in rad/s.

**Dependencies**

To enable this parameter, for **Parameterize losses by**, select `Tabulated loss data`.

**`Vector of torques (T) for tabulated loss, T_loss_bp` — Breakpoints**
`[0 25 50 75 100]` (default) | 1-by-`N` vector

Torque breakpoints for lookup table when calculating losses, in N·m.

**Dependencies**

To enable this parameter, for **Parameterize losses by**, select `Tabulated loss data`.

**`Corresponding losses, losses_table` — Table**
`[100 100 100 100 100;100 150 200 250 300;100 200 300 400 500;100 250 400 550 700;100 300 500 700 900;100 350 600 850 1100]` (default) | `M`-by-`N` array

Array of values for electrical losses as a function of `M` speeds and `N` torques, in W. Each value specifies the losses for a specific combination of speed and torque. The matrix size must match the dimensions defined by the speed and torque vectors.

#### Dependencies

To enable this parameter, for **Parameterize losses by**, select `Tabulated loss data`.

**`Vector of speeds (w) for tabulated efficiency, w_eff_bp` — Breakpoints**
`[200 400 600 800 1000]` (default) | 1-by-M vector

Speed breakpoints for lookup table when calculating efficiency, in rad/s.

#### Dependencies

To enable this parameter, for **Parameterize losses by**, select `Tabulated efficiency data`.

**`Vector of torques (T) for tabulated efficiency, T_eff_bp` — Breakpoints**
`[25 50 75 100]` (default) | 1-by-N vector

Torque breakpoints for lookup table when calculating efficiency, in N·m.

#### Dependencies

To enable this parameter, for **Parameterize losses by**, select `Tabulated efficiency data`.

**`Corresponding efficiency, efficiency_table` — Table**
`[96.2 98.1 98.7 99;98.1 99 99.4 99.5;98.7 99.4 99.6 99.7;99 99.5 99.7 99.8;99.2 99.6 99.7 99.8]` (default) | M-by-N array

Array of efficiency as a function of `M` speeds and `N` torque, in %. Each value specifies the efficiency for a specific combination of speed and torque. The matrix size must match the dimensions defined by the speed and torque vectors.

The block ignores efficiency values for zero speed or zero torque. Losses are zero when either torque or speed is zero. The block uses linear interpolation.

To get the desired level of accuracy for lower power conditions, you can provide tabulated data for low speeds and low torques.

#### Dependencies

To enable this parameter, for **Parameterize losses by**, select `Tabulated efficiency data`.

## References

[1] Hu, Dakai, Yazan Alsmadi, and Longya Xu. “High fidelity nonlinear IPM modeling based on measured stator winding flux linkage.” *IEEE Transactions on Industry Applications*, Vol. 51, No. 4, July/August 2015.

[2] Chen, Xiao, Jiabin Wang, Bhaskar Sen, Panagiotis Lasari, Tianfu Sun. “A High-Fidelity and Computationally Efficient Model for Interior Permanent-Magnet Machines Considering the Magnetic Saturation, Spatial Harmonics, and Iron Loss Effect.” *IEEE Transactions on Industrial Electronics*, Vol. 62, No. 7, July 2015.

[3] Ottosson, J., M. Alakula. “A compact field weakening controller implementation.” *International Symposium on Power Electronics, Electrical Drives, Automation and Motion*, July, 2006.

## Extended Capabilities

### C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

## See Also

Flux-Based PMSM | IM Controller | Interior PM Controller | Surface Mount PM Controller

### Topics

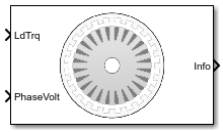
“Generate Parameters for Flux-Based Blocks”

### Introduced in R2017b

# Induction Motor

Three-phase induction motor

**Library:** Powertrain Blockset / Propulsion / Electric Motors and Inverters
Motor Control Blockset / Electrical Systems / Motors



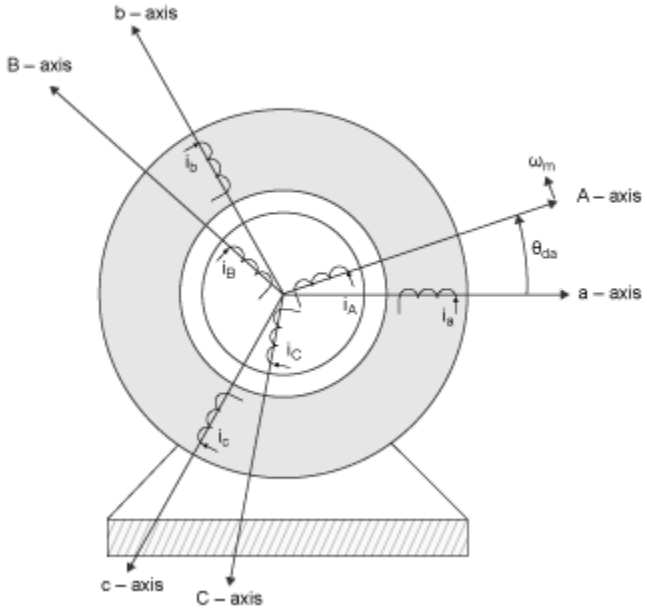

## Description

The Induction Motor block implements a three-phase induction motor. The block uses the three-phase input voltages to regulate the individual phase currents, allowing control of the motor torque or speed.

By default, the block sets the **Simulation Type** parameter to `Continuous` to use a continuous sample time during simulation. If you want to generate code for fixed-step double- and single-precision targets, considering setting the parameter to `Discrete`. Then specify a **Sample Time, Ts** parameter.

### Three-Phase Sinusoidal Model Electrical System

The block implements equations that are expressed in a stationary rotor reference (qd) frame. The d-axis aligns with the a-axis. All quantities in the rotor reference frame are referred to the stator.




The block uses these equations to calculate the electrical speed ($\omega_{em}$) and slip speed ($\omega_{slip}$).

$$\omega_{em} = P\omega_m$$
$$\omega_{slip} = \omega_{syn} - \omega_{em}$$

To calculate the dq rotor electrical speed with respect to the rotor A-axis ($dA$), the block uses the difference between the stator a-axis ($da$) speed and slip speed:

$$\omega_{dA} = \omega_{da} - \omega_{em}$$

To simplify the equations for the flux, voltage, and current transformations, the block uses a stationary reference frame:

$$\omega_{da} = 0$$
$$\omega_{dA} = -\omega_{em}$$

| Calculation | Equation |
|---|---|
| Flux | $$\frac{d}{dt}\begin{bmatrix}\lambda_{sd}\\ \lambda_{sq}\end{bmatrix} = \begin{bmatrix}v_{sd}\\ v_{sq}\end{bmatrix} - R_s\begin{bmatrix}i_{sd}\\ i_{sq}\end{bmatrix} - \omega_{da}\begin{bmatrix}0 & -1\\ 1 & 0\end{bmatrix}\begin{bmatrix}\lambda_{sd}\\ \lambda_{sq}\end{bmatrix}$$ $$\frac{d}{dt}\begin{bmatrix}\lambda_{rd}\\ \lambda_{rq}\end{bmatrix} = \begin{bmatrix}v_{rd}\\ v_{rq}\end{bmatrix} - R_r\begin{bmatrix}i_{rd}\\ i_{rq}\end{bmatrix} - \omega_{dA}\begin{bmatrix}0 & -1\\ 1 & 0\end{bmatrix}\begin{bmatrix}\lambda_{rd}\\ \lambda_{rq}\end{bmatrix}$$ $$\begin{bmatrix}\lambda_{sd}\\ \lambda_{sq}\\ \lambda_{rd}\\ \lambda_{rq}\end{bmatrix} = \begin{bmatrix}L_s & 0 & L_m & 0\\ 0 & L_s & 0 & L_m\\ L_m & 0 & L_r & 0\\ 0 & L_m & 0 & L_r\end{bmatrix}\begin{bmatrix}i_{sd}\\ i_{sq}\\ i_{rd}\\ i_{rq}\end{bmatrix}$$ |
| Current | $$\begin{bmatrix}i_{sd}\\ i_{sq}\\ i_{rd}\\ i_{rq}\end{bmatrix} = \left(\frac{1}{L_m^2 - L_rL_s}\right)\begin{bmatrix}-L_r & 0 & L_m & 0\\ 0 & -L_r & 0 & L_m\\ L_m & 0 & -L_s & 0\\ 0 & L_m & 0 & -L_s\end{bmatrix}\begin{bmatrix}\lambda_{sd}\\ \lambda_{sq}\\ \lambda_{rd}\\ \lambda_{rq}\end{bmatrix}$$ |
| Inductance | $$L_s = L_{ls} + L_m$$ $$L_r = L_{lr} + L_m$$ |
| Electromagnetic torque | $$T_e = PL_m(i_{sq}i_{rd} - i_{sd}i_{rq})$$ |
| Power invariant dq transformation to ensure that the dq and three phase powers are equal | $$\begin{bmatrix}v_{sd}\\ v_{sq}\end{bmatrix} = \sqrt{\frac{2}{3}}$$ $$\begin{bmatrix}\cos(\Theta_{da}) & \cos(\Theta_{da} - \frac{2\pi}{3}) & \cos(\Theta_{da} + \frac{2\pi}{3})\\ -\sin(\Theta_{da}) & -\sin(\Theta_{da} - \frac{2\pi}{3}) & -\sin(\Theta_{da} + \frac{2\pi}{3})\end{bmatrix}\begin{bmatrix}v_a\\ v_b\\ v_c\end{bmatrix}$$ $$\begin{bmatrix}i_a\\ i_b\\ i_c\end{bmatrix} = \sqrt{\frac{2}{3}}\begin{bmatrix}\cos(\Theta_{da}) & -\sin(\Theta_{da})\\ \cos(\Theta_{da} - \frac{2\pi}{3}) & -\sin(\Theta_{da} - \frac{2\pi}{3})\\ \cos(\Theta_{da} + \frac{2\pi}{3}) & -\sin(\Theta_{da} + \frac{2\pi}{3})\end{bmatrix}\begin{bmatrix}i_{sd}\\ i_{sq}\end{bmatrix}$$ |

The equations use these variables.

| | |
|---|---|
| $\omega_m$ | Angular velocity of the rotor |
| $\omega_{em}$ | Electrical rotor speed |
| $\omega_{slip}$ | Electrical rotor slip speed |
| $\omega_{syn}$ | Synchronous rotor speed |
| $\omega_{da}$ | dq stator electrical speed with respect to the rotor a-axis |
| $\omega_{dA}$ | dq stator electrical speed with respect to the rotor A-axis |
| $\Theta_{da}$ | dq stator electrical angle with respect to the rotor a-axis |
| $\Theta_{dA}$ | dq stator electrical angle with respect to the rotor A-axis |
| $L_q$, $L_d$ | q- and d-axis inductances |
| $L_s$ | Stator inductance |
| $L_r$ | Rotor inductance |
| $L_m$ | Magnetizing inductance |
| $L_{ls}$ | Stator leakage inductance |
| $L_{lr}$ | Rotor leakage inductance |
| $v_{sq}$, $v_{sd}$ | Stator q- and d-axis voltages |
| $i_{sq}$, $i_{sd}$ | Stator q- and d-axis currents |
| $\lambda_{sq}$, $\lambda_{sd}$ | Stator q- and d-axis flux |
| $i_{rq}$, $i_{rd}$ | Rotor q- and d-axis currents |
| $\lambda_{rq}$, $\lambda_{rd}$ | Rotor q- and d-axis flux |
| $v_a$, $v_b$, $v_c$ | Stator voltage phases a, b, c |
| $i_a$, $i_b$, $i_c$ | Stator currents phases a, b, c |
| $R_s$ | Resistance of the stator windings |
| $R_r$ | Resistance of the rotor windings |
| $P$ | Number of pole pairs |
| $T_e$ | Electromagnetic torque |

### Mechanical System

The motor angular velocity is given by:

$$\frac{d}{dt}\omega_m = \frac{1}{J}\left(T_e - T_f - F\omega_m - T_m\right)$$

$$\frac{d\theta_m}{dt} = \omega_m$$

The equations use these variables.

| | |
|---|---|
| $J$ | Combined inertia of motor and load |
| $F$ | Combined viscous friction of motor and load |
| $\theta_m$ | Motor mechanical angular position |
| $T_m$ | Motor shaft torque |

| | |
|---|---|
| $T_e$ | Electromagnetic torque |
| $T_f$ | Motor shaft static friction torque |
| $\omega_m$ | Angular mechanical velocity of the motor |

### Power Accounting

For the power accounting, the block implements these equations.

| Bus Signal | | | Description | Variable | Equations |
|---|---|---|---|---|---|
| `PwrInfo` | `PwrTrnsfrd` — Power transferred between blocks<br>• Positive signals indicate flow into block<br>• Negative signals indicate flow out of block | `PwrMtr` | Mechanical power | $P_{mot}$ | $P_{mot} = \quad -\omega_m T_e$ |
| | | `PwrBus` | Electrical power | $P_{bus}$ | $P_{bus} = \quad v_{an} i_a + \quad v_{bn} i_b + v_{cn} i_c$ |
| | `PwrNotTrnsfrd` — Power crossing the block boundary, but not transferred<br>• Positive signals indicate an input<br>• Negative signals indicate a loss | `PwrElecLoss` | Resistive power loss | $P_{elec}$ | $P_{elec} = \quad -(R_s i_{sd}^2 + R_s i_{sq}^2 + \quad - R_r i_{rd}^2 + R_r i_{rq}^2)$ |
| | | `PwrMechLoss` | Mechanical power loss | $P_{mech}$ | When **Port Configuration** is set to `Torque`:<br>$P_{mech} = \quad -\left(\omega_m^2 F + \quad \lvert\omega_m\rvert T_f\right)$<br>When **Port Configuration** is set to `Speed`:<br>$P_{mech} = \quad 0$ |
| | `PwrStored` — Stored energy rate of change<br>• Positive signals indicate an increase<br>• Negative signals indicate a decrease | `PwrMtrStored` | Stored motor power | $P_{str}$ | $P_{str} = \quad P_{bus} + \quad P_{mot} + \quad P_{elec} \quad + \quad P_{mech}$ |

The equations use these variables.

| | |
|---|---|
| $R_s$ | Stator resistance |
| $R_r$ | Motor resistance |
| $i_a$, $i_b$, $i_c$ | Stator phase a, b, and c current |
| $i_{sq}$, $i_{sd}$ | Stator q- and d-axis currents |
| $v_{an}$, $v_{bn}$, $v_{cn}$ | Stator phase a, b, and c voltage |
| $\omega_m$ | Angular mechanical velocity of the rotor |

| | |
|---|---|
| $F$ | Combined motor and load viscous damping |
| $T_e$ | Electromagnetic torque |
| $T_f$ | Combined motor and load friction torque |

## Ports

### Input

#### `LdTrq` — Rotor shaft torque

`scalar`

Rotor shaft input torque, $T_m$, in N·m.

##### Dependencies

To create this port, select `Torque` for the **Port configuration** parameter.

#### `Spd` — Rotor shaft speed

`scalar`

Angular velocity of the rotor, $\omega_m$, in rad/s.

##### Dependencies

To create this port, select `Speed` for the **Port configuration** parameter.

#### `PhaseVolt` — Stator terminal voltages

1-by-3 array

Stator terminal voltages, $V_a$, $V_b$, and $V_c$, in V.

### Output

#### `Info` — Bus signal

bus

The bus signal contains these block calculations.

| Signal | Description | Variable | Units |
|---|---|---|---|
| `IaStator` | Stator phase current A | $i_a$ | A |
| `IbStator` | Stator phase current B | $i_b$ | A |
| `IcStator` | Stator phase current C | $i_c$ | A |
| `IdSync` | Direct axis current | $i_d$ | A |
| `IqSync` | Quadrature axis current | $i_q$ | A |
| `VdSync` | Direct axis voltage | $v_d$ | V |
| `VqSync` | Quadrature axis voltage | $v_q$ | V |
| `MtrSpd` | Angular mechanical velocity of the rotor | $\omega_m$ | rad/s |
| `MtrMechPos` | Rotor mechanical angular position | $\theta_m$ | rad |

| Signal | | | Description | Variable | Units |
|---|---|---|---|---|---|
| `MtrPos` | | | Rotor electrical angular position | $\theta_e$ | rad |
| `MtrTrq` | | | Electromagnetic torque | $T_e$ | N·m |
| `PwrInfo` | `PwrTrnsfrd` | `PwrMtr` | Mechanical power | $P_{mot}$ | W |
| | | `PwrBus` | Electrical power | $P_{bus}$ | W |
| | `PwrNotTrnsfrd` | `PwrElecLoss` | Resistive power loss | $P_{elec}$ | W |
| | | `PwrMechLoss` | Mechanical power loss | $P_{mech}$ | W |
| | `PwrStored` | `PwrMtrStored` | Stored motor power | $P_{str}$ | W |

**`PhaseCurr` — Phase a, b, c current**
1-by-3 array

Phase a, b, c current, $i_a$, $i_b$, and $i_c$, in A.

**`MtrTrq` — Motor torque**
`scalar`

Motor torque, $T_{mtr}$, in N·m.

**Dependencies**

To create this port, select `Speed` for the **Port configuration** parameter.

**`MtrSpd` — Motor speed**
`scalar`

Angular speed of the motor, $\omega_{mtr}$, in rad/s.

**Dependencies**

To create this port, select `Torque` for the **Port configuration** parameter.

## Parameters

**Block Options**

**`Simulation type` — Select simulation type**
`Continuous` (default) | `Discrete`

By default, the block uses a continuous sample time during simulation. If you want to generate code for single-precision targets, considering setting the parameter to `Discrete`.

**Dependencies**

Setting **Simulation Type** to `Discrete` creates the **Sample Time, Ts** parameter.

**`Sample time, Ts` — Sample time for discrete integration**
`0.001` (default) | `scalar`

Integration sample time for discrete simulation, in s.

#### Dependencies

Setting **Simulation Type** to `Discrete` creates the **Sample Time, Ts** parameter.

### `Port configuration` — Select port configuration

`Torque` (default) | `Speed`

This table summarizes the port configurations.

| Port Configuration | Creates Input Port | Creates Output Port |
|---|---|---|
| `Torque` | `LdTrq` | `MtrSpd` |
| `Speed` | `Spd` | `MtrTrq` |

#### Parameters

### `Number of pole pairs, P` — Pole pairs

`2` (default) | `scalar`

Motor pole pairs, $P$.

### `Stator resistance and leakage inductance, Zs` — Resistance and inductance

`[1.77 0.0139]` (default) | `vector`

Stator resistance, $R_S$, in ohms and leakage inductance, $L_{ls}$, in H.

### `Rotor resistance and leakage inductance, Zr` — Resistance and inductance

`[1.34 0.0121]` (default) | `vector`

Rotor resistance, $R_r$, in ohms and leakage inductance, $L_{lr}$, in H.

### `Magnetizing inductance, Lm` — Inductance

`0.3687` (default) | `scalar`

Magnetizing inductance, $L_m$, in H.

### `Physical inertia, viscous damping, static friction, mechanical` — Inertia, damping, friction

`[0.001 0 0]` (default) | `vector`

Mechanical properties of the rotor:

- Inertia, $J$, in kg·m^2
- Viscous damping, $F$, in N·m/(rad/s)
- Static friction, $T_f$, in N·m

#### Dependencies

To enable this parameter, select `Torque` for the **Port configuration**.

#### Initial Values

### `Initial mechanical position, theta_init` — Angular position

`0` (default) | `scalar`

Initial rotor angular position, $\theta_{m0}$, in rad.

**`Initial mechanical speed, omega_init` — Angular speed**
`0` (default) | `scalar`

Initial angular velocity of the rotor, $\omega_{m0}$, in rad/s.

**Dependencies**

To enable this parameter, select `Torque` for the **Port configuration**.

## References

[1] Mohan, Ned. *Advanced Electric Drives: Analysis, Control and Modeling Using Simulink*. Minneapolis, MN: MNPERE, 2001.

## Extended Capabilities

**C/C++ Code Generation**
Generate C and C++ code using Simulink® Coder™.

## See Also

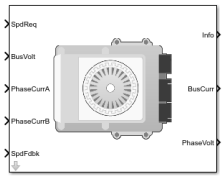
Flux-Based PMSM | IM Controller | Interior PMSM | Mapped Motor | Surface Mount PMSM

**Introduced in R2017a**

# IM Controller

Internal torque-based, field-oriented controller for an induction motor with an optional outer-loop speed controller

**Library:** Powertrain Blockset / Propulsion / Electric Motor Controllers

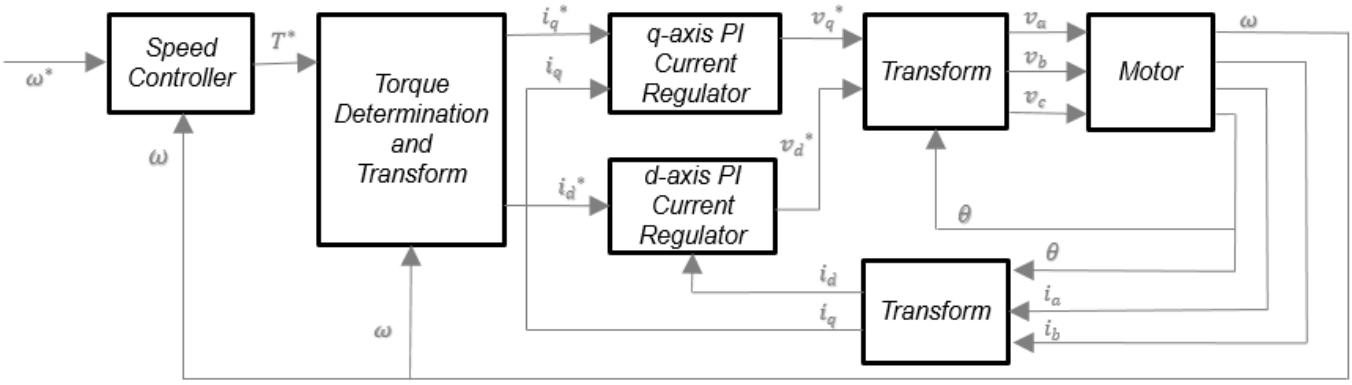



## Description

The IM Controller block implements an internal torque-based, field-oriented controller for an induction motor (IM) with an optional outer-loop speed controller. The torque control implements a strategy to control the motor flux. You can specify either speed or torque control.

The IM Controller implements equations for speed control, torque determination, regulators, transforms, and motors.

The figure illustrates the information flow in the block.




The block implements equations that use these variables.

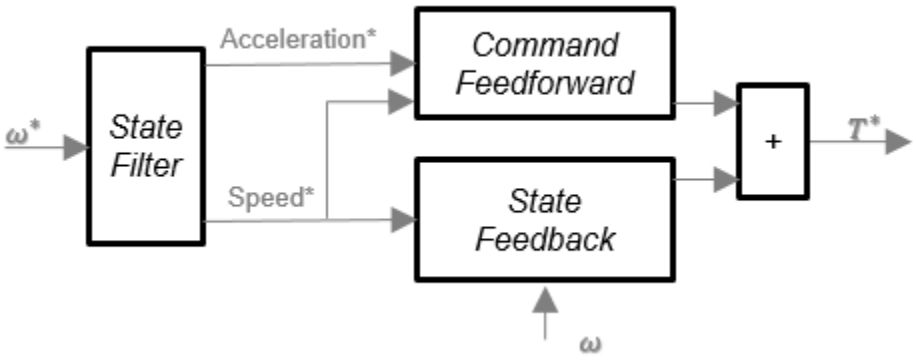
| | |
|---|---|
| $\omega$ | Rotor speed |
| $\omega$* | Rotor speed command |
| $T$* | Torque command |
| $i_d$ | d-axis current |
| $i^*_d$ | d-axis current command |
| $i_q$ | q-axis current |
| $i^*_q$ | q-axis current command |

| | |
|---|---|
| $v_d$, | d-axis voltage |
| $v^*_d$ | d-axis voltage command |
| $v_q$ | q-axis voltage |
| $v^*_q$ | q-axis voltage command |
| $v_a$, $v_b$, $v_c$ | Stator phase a, b, c voltages |
| $i_a$, $i_b$, $i_c$ | Stator phase a, b, c currents |

### Speed Controller

To implement the speed controller, select the **Control Type** parameter `Speed Control`. If you select the **Control Type** parameter `Torque Control`, the block does not implement the speed controller.

The speed controller determines the torque command by implementing a state filter, and calculating the feedforward and feedback commands. If you do not implement the speed controller, input a torque command to the IM Controller block.




#### State Filter

The state filter is a low-pass filter that generates the acceleration command based on the speed command. On the **Speed Controller** tab:

- To make the speed-command lag time negligible, specify a **Bandwidth of the state filter** parameter.
- To calculate a **Speed regulation time constant, Ksf** gain based on the state filter bandwidth, select **Calculate Speed Regulator Gains**.

The discrete form of characteristic equation is given by:

$$z + K_{sf}T_{sm} - 1$$

The filter calculates the gain using this equation.

$$K_{sf} = \frac{1 - \exp\left(-T_{sm}2\pi EV_{sf}\right)}{T_{sm}}$$

The equation uses these variables.

| | |
|---|---|
| $EV_{sf}$ | Bandwidth of the speed command filter |
| $T_{sm}$ | Motion controller sample time |
| $K_{sf}$ | Speed regulator time constant |

#### State Feedback

To generate the state feedback torque, the block uses the filtered speed error signal from the state filter. The feedback torque calculation also requires gains for speed regulator.

On the **Speed Controller** tab, select **Calculate Speed Regulator Gains** to compute:

- **Proportional gain, ba**
- **Angular gain, Ksa**
- **Rotational gain, Kisa**

For the gain calculations, the block uses the inertia from the **Physical inertia, viscous damping, static friction** parameter value on the **Motor Parameter** tab.

The gains for the state feedback are calculated using these equations.

| Calculation | Equations |
|---|---|
| Discrete forms of characteristic equation | $z^3 + \frac{\left(-3J_p + T_s b_a + \quad T_s^2 K_{sa} + T_s^3 K_{isa}\right)}{J_p} z^2 + \frac{\left(3J_p - \quad 2T_s b_a - \quad T_s^2 K_{sa}\right)}{J_p} z + \frac{-J_p + T_s b_a}{J_p}$<br><br>$(z - p_1)(z - p_2)(z - p_3) = z^3 + (p_1 + p_2 + p_3)z^2 + (p_1 p_2 + p_2 p_3 + p_1 3)z^2 - p_1 p_2 p_3$ |
| Speed regulator proportional gain | $b_a = \frac{J_p - J_p p_1 p_2 p_3}{T_{sm}}$ |
| Speed regulator integral gain | $K_{sa} = \frac{J_p(p_1 p_2 + p_2 p_3 + p_3 p_1) - 3J_p + 2b_a T_{sm}}{T_{sm}^2}$ |
| Speed regulator double integral gain | $K_{isa} = \frac{-J_p(p_1 + p_2 + p_3) + 3J_p - b_a T_{sm} - K_{sa} T_{sm}^2}{T_{sm}^3}$ |

The equations use these variables.

| | |
|---|---|
| $P$ | Motor pole pairs |
| $b_a$ | Speed regulator proportional gain |
| $K_{sa}$ | Speed regulator integral gain |
| $K_{isa}$ | Speed regulator double integral gain |
| $J_p$ | Motor inertia |
| $T_{sm}$ | Motion controller sample time |

#### Command Feedforward

To generate the state feedforward torque, the block uses the filtered speed and acceleration from the state filter. Also, the feedforward torque calculation uses the inertia, viscous damping, and static friction. To achieve zero tracking error, the torque command is the sum of the feedforward and feedback torque commands.

Selecting **Calculate Speed Regulator Gains** on the **Speed Controller** tab updates the inertia, viscous damping, and static friction with the **Physical inertia, viscous damping, static friction** parameter values on the **Motor Parameter** tab.

The feedforward torque command uses this equation.

$$T_{cmd_ff} = J_p\dot{\omega}_m + F_v\omega_m + F_s\frac{\omega_m}{|\omega_m|}$$

The equation uses these variables.

| | |
|---|---|
| $J_p$ | Motor inertia |
| $T_{cmd_ff}$ | Torque command feedforward |
| $F_s$ | Static friction torque constant |
| $F_v$ | Viscous friction torque constant |
| $F_s$ | Static friction torque constant |
| $\omega_m$ | Rotor mechanical speed |

### Torque Determination

The block uses a quadrature current to determine the base speed and the current commands. The motor ratings determine the rated electrical speed.

| Calculation | Equations |
|---|---|
| Current commands | $i_{qref} = \frac{T_{cmd}}{i_{sq_0} \cdot P \cdot \left(\frac{L^2_m}{L_r}\right)}$<br><br>`If` $\lvert\omega_e\rvert \le \omega_{rated}$<br>$i_{dref} = i_{sd_0}$<br>`Else`<br>$i_{dref} = \frac{i_{sd_0}}{\lvert\omega_e\rvert}$<br>`End` |
| Inductance | $L_r = L_{lr} + L_m$<br>$L_s = L_{ls} + L_m$ |

The equations use these variables.

| | |
|---|---|
| $i_{dref}$ | d-axis reference current |
| $i_{qref}$ | q-axis reference current |
| $i_{sd_0}$ | d-axis rated current |
| $i_{sq_0}$ | q-axis rated current |
| $\omega_e$ | Rotor electrical speed |
| $\omega_{rated}$ | Rated electrical speed |
| $L_{lr}$ | Rotor leaking inductance |

| | |
|---|---|
| $L_r$ | Rotor winding inductance |
| $L_{ls}$ | Stator leaking inductance |
| $L_s$ | Stator winding inductance |
| $L_m$ | Motor magnetizing inductance |
| $P$ | Motor pole pairs |
| $T_{cmd}$ | Commanded motor maximum torque |

### Current Regulators

The block regulates the current with an anti-windup feature. Classic proportional-integrator (PI) current regulators do not consider the d-axis and q-axis coupling or the back-electromagnetic force (EMF) coupling. As a result, transient performance deteriorates. To account for the coupling, the block implements the complex vector current regulator (CVCR) in the scalar format of the rotor reference frame. The CVCR decouples:

- d-axis and q-axis current cross-coupling
- Back-EMF cross-coupling

The current frequency response is a first-order system, with a bandwidth of $EV_{current}$.

The block implements these equations.

| Calculation | Equations |
|---|---|
| Motor voltage, in the stator reference frame | $\sigma = 1 - \frac{L^2{}_m}{L_s L_r}$<br>$v_{sd} = R_s i_{sd} + \sigma L_s \frac{di_{sd}}{dt} + \frac{L_m}{L_r}\frac{d\lambda_{rd}}{dt} - P\omega_m \sigma L_s i_{sq}$<br>$v_{sq} = R_s i_{sq} + \sigma L_s \frac{di_{sq}}{dt} + \omega_d \frac{L_m}{L_r}\frac{d\lambda_{rd}}{dt} + P\omega_m \sigma L_s i_{sd}$ |
| Current regulator gains | $\omega_b = 2\pi EV_{current}$<br>$K_p = \sigma L_d \omega_b$<br>$K_i = R_s \omega_b$ |
| Transfer functions | $\frac{i_d}{i_{dref}} = \frac{\omega_b}{s + \omega_b}$<br>$\frac{i_q}{i_{qref}} = \frac{\omega_b}{s + \omega_b}$ |

The equations use these variables.

| | |
|---|---|
| $EV_{current}$ | Current regulator bandwidth |
| $i_d$ | d-axis current |
| $i_q$ | q-axis current |
| $i_{sq}$ | Stator q-axis current |
| $i_{sd}$ | Stator d-axis current |
| $v_{sd}$ | Stator d-axis voltage |

| | |
|---|---|
| $v_{sq}$ | Stator q-axis voltage |
| $K_p$ | Current regulator d-axis gain |
| $K_i$ | Current regulator integrator gain |
| $L_s$ | Stator winding inductance |
| $L_m$ | Motor magnetizing inductance |
| $L_r$ | Rotor winding inductance |
| $R_s$ | Stator phase winding resistance |
| $\lambda_{rd}$ | Rotor d-axis magnetic flux |
| $\sigma$ | Leakage factor |
| $p$ | Motor pole pairs |

### Transforms

To calculate the voltages and currents in balanced three-phase ($a$, $b$) quantities, quadrature two-phase ($\alpha$, $\beta$) quantities, and rotating ($d$, $q$) reference frames, the block uses the Clarke and Park Transforms.

In the transform equations.

$$\omega_e = P\omega_m$$

$$\frac{d\theta_e}{dt} = \omega_e$$

| Transform | Description | Equations |
|---|---|---|
| Clarke | Converts balanced three-phase quantities ($a$, $b$) into balanced two-phase quadrature quantities ($\alpha$, $\beta$). | $x_\alpha = \frac{2}{3}x_a - \frac{1}{3}x_b - \frac{1}{3}x_c$<br>$x_\beta = \frac{\sqrt{3}}{2}x_b - \frac{\sqrt{3}}{2}x_c$ |
| Park | Converts balanced two-phase orthogonal stationary quantities ($\alpha$, $\beta$) into an orthogonal rotating reference frame ($d$, $q$). | $x_d = x_\alpha\cos\theta_e + x_\beta\sin\theta_e$<br>$x_q = -x_\alpha\sin\theta_e + x_\beta\cos\theta_e$ |
| Inverse Clarke | Converts balanced two-phase quadrature quantities ($\alpha$, $\beta$) into balanced three-phase quantities ($a$, $b$). | $x_a = x_a$<br>$x_b = -\frac{1}{2}x_\alpha + \frac{\sqrt{3}}{2}x_\beta$<br>$x_c = -\frac{1}{2}x_\alpha - \frac{\sqrt{3}}{2}x_\beta$ |
| Inverse Park | Converts an orthogonal rotating reference frame ($d$, $q$) into balanced two-phase orthogonal stationary quantities ($\alpha$, $\beta$). | $x_\alpha = x_d\cos\theta_e - x_q\sin\theta_e$<br>$x_\beta = x_d\sin\theta_e + x_q\cos\theta_e$ |

The transforms use these variables.

| | |
|---|---|
| $\omega_m$ | Rotor mechanical speed |
| $P$ | Motor pole pairs |
| $\omega_e$ | Rotor electrical speed |

| | |
|---|---|
| $\Theta_e$ | Rotor electrical angle |
| $x$ | Phase current or voltage |

### Motor

The block uses the phase currents and phase voltages to estimate the DC bus current. Positive current indicates battery discharge. Negative current indicates battery charge. The block uses these equations.

| | |
|---|---|
| Load power | $Ld_{Pwr} = v_a \quad i_a + \quad v_b \quad i_b + \quad v_c \quad i_c$ |
| Source power | $Src_{Pwr} = \quad Ld_{Pwr} + Pwr_{Loss}$ |
| DC bus current | $i_{bus} = \quad \frac{Src_{Pwr}}{v_{bus}}$ |
| Estimated rotor torque | $MtrTrq_{est} = P\lambda_{rd} i_{sq} \frac{L_m}{L_r}$ |
| Power loss for single efficiency source to load | $Pwr_{Loss} = \frac{100 - Eff}{Eff} \cdot Ld_{Pwr}$ |
| Power loss for single efficiency load to source | $Pwr_{Loss} = \frac{100 - Eff}{100} \cdot \lvert Ld_{Pwr} \rvert$ |
| Power loss for tabulated efficiency | $Pwr_{Loss} = \quad f(\omega_m, MtrTrq_{est})$ |

The equations use these variables.

| | |
|---|---|
| $v_a$, $v_b$, $v_c$ | Stator phase a, b, c voltages |
| $v_{bus}$ | Estimated DC bus voltage |
| $i_a$, $i_b$, $i_c$ | Stator phase a, b, c currents |
| $i_{bus}$ | Estimated DC bus current |
| $Eff$ | Overall inverter efficiency |
| $\omega_m$ | Rotor mechanical speed |
| $L_r$ | Rotor winding inductance |
| $L_m$ | Motor magnetizing inductance |
| $\lambda_{rd}$ | Rotor d-axis magnetic flux |
| $i_{sq}$ | q-axis current |
| $P$ | Motor pole pairs |

### Electrical Losses

To specify the electrical losses, on the **Electrical Losses** tab, for **Parameterize losses by**, select one of these options.

| Setting | Block Implementation |
|---|---|
| `Single efficiency measurement` | Electrical loss calculated using a constant value for inverter efficiency. |
| `Tabulated loss data` | Electrical loss calculated as a function of motor speeds and load torques. |

| Setting | Block Implementation |
|---|---|
| `Tabulated efficiency data` | Electrical loss calculated using inverter efficiency that is a function of motor speeds and load torques.<br>• Converts the efficiency values you provide into losses and uses the tabulated losses for simulation.<br>• Ignores efficiency values you provide for zero speed or zero torque. Losses are assumed zero when either torque or speed is zero.<br>• Uses linear interpolation to determine losses. Provide tabulated data for low speeds and low torques, as required, to get the desired level of accuracy for lower power conditions.<br>• Does not extrapolate loss values for speed and torque magnitudes that exceed the range of the table. |

For best practice, use `Tabulated loss data` instead of `Tabulated efficiency data`:

- Efficiency becomes ill defined for zero speed or zero torque.
- You can account for fixed losses that are still present for zero speed or torque.

## Ports

### Input

#### SpdReq — Rotor mechanical speed command

`scalar`

Rotor mechanical speed command, $\omega^*_m$, in rad/s.

##### Dependencies

To create this port, select `Speed Control` for the **Control Type** parameter.

#### TrqCmd — Torque command

`scalar`

Torque command, $T^*$, in N·m.

##### Dependencies

To create this port, select `Torque Control` for the **Control Type** parameter.

#### BusVolt — DC bus voltage

`scalar`

DC bus voltage $v_{bus}$, in V.

#### PhaseCurrA — Current

`scalar`

Stator current phase a, $i_a$, in A.

#### PhaseCurrB — Current

`scalar`

Stator current phase b, $i_b$, in A.

**SpdFdbk — Rotor mechanical speed**
`scalar`

Rotor mechanical speed, $\omega_m$, in rad/s.

**Output**

**`Info` — Bus signal**
bus

Bus signal containing these block calculations.

| Signal | Description | Units |
|---|---|---|
| `SrcPwr` | Source power | W |
| `LdPwr` | Load power | W |
| `PwrLoss` | Power loss | W |
| `MtrTrqEst` | Estimated motor torque | N·m |

**`BusCurr` — Bus current**
`scalar`

Estimated DC bus current, $i_{bus}$, in A.

**`PhaseVolt` — Stator terminal voltages**
`array`

Stator terminal voltages, $V_a$, $V_b$, and $V_c$, in V.

## Parameters

**Block Options**

**`Control Type` — Select control**
`Speed Control` (default) | `Torque Control`

If you select `Torque Control`, the block does not implement the speed controller.

This table summarizes the port configurations.

| Port Configuration | Creates Ports |
|---|---|
| `Speed Control` | `SpdReq` |
| `Torque Control` | `TrqCmd` |

**Motor**

**`Stator resistance, Rs` — Resistance**
`1.77` (default) | `scalar`

Stator phase winding resistance, $R_s$, in ohm.

#### Dependencies

This table summarizes the parameter dependencies.

| Parameter | Used to Derive | |
|---|---|---|
| | Parameter | Tab |
| Stator resistance, Rs | D-axis rated current, Isd_0<br>Q-axis rated current, Isq_0<br>Torque at rated current, Tem | Id and Iq Calculation |
| | D and Q axis integral gain, Ki | Current Controller |

### `Stator leakage inductance, Lls` — Inductance

`0.0139` (default) | `scalar`

Stator leakage inductance, $L_{ls}$, in H.

#### Dependencies

This table summarizes the parameter dependencies.

| Parameter | Used to Derive | |
|---|---|---|
| | Parameter | Tab |
| Stator leakage inductance, Lls | D-axis rated current, Isd_0<br>Q-axis rated current, Isq_0<br>Torque at rated current, Tem | Id and Iq Calculation |
| | D and Q axis proportional gain, Kp<br>D and Q axis integral gain, Ki | Current Controller |

### `Rotor resistance, Rr` — Resistance

`1.34` (default) | `scalar`

Rotor resistance, $R_r$, in ohm.

#### Dependencies

This table summarizes the parameter dependencies.

| Parameter | Used to Derive | |
|---|---|---|
| | Parameter | Tab |
| Rotor resistance, Rr | D-axis rated current, Isd_0<br>Q-axis rated current, Isq_0<br>Torque at rated current, Tem | Id and Iq Calculation |

### `Rotor leakage inductance, Llr` — Inductance

`0.0121` (default) | `scalar`

Rotor leakage inductance, $L_{lr}$, in H.

**Dependencies**

This table summarizes the parameter dependencies.

| Parameter | Used to Derive | |
|---|---|---|
| | Parameter | Tab |
| Rotor leakage inductance, Llr | D-axis rated current, Isd_0<br>Q-axis rated current, Isq_0<br>Torque at rated current, Tem | Id and Iq Calculation |
| | D and Q axis proportional gain, Kp | Current Controller |

**`Rotor magnetizing inductance, Lm` — Inductance**
`0.3687` (default) | `scalar`

Rotor magnetizing inductance, $L_m$, in H.

**Dependencies**

This table summarizes the parameter dependencies.

| Parameter | Used to Derive | |
|---|---|---|
| | Parameter | Tab |
| Rotor leakage inductance, Llr | D-axis rated current, Isd_0<br>Q-axis rated current, Isq_0<br>Torque at rated current, Tem | Id and Iq Calculation |
| | D and Q axis proportional gain, Kp | Current Controller |

**`Number of pole pairs, PolePairs` — Poles**
`2` (default) | `scalar`

Motor pole pairs, $P$.

**Dependencies**

This table summarizes the parameter dependencies.

| Parameter | Used to Derive | |
|---|---|---|
| | Parameter | Tab |
| Rotor leakage inductance, Llr | Torque at rated current, Tem | Id and Iq Calculation |

**`Physical inertia, viscous damping, static friction, Mechanical` — Mechanical properties of motor**
`[0.025, 0, 0]` (default) | `vector`

Mechanical properties of the motor:

- Motor inertia, $F_v$, in kgm^2
- Viscous friction torque constant, $F_v$, in N·m/(rad/s)
- Static friction torque constant, $F_s$, in N·m

#### Dependencies

To enable this parameter, set the **Control Type** parameter to `Speed Control`.

For the gain calculations, the block uses the inertia from the **Physical inertia, viscous damping, static friction** parameter value that is on the **Motor Parameters** tab.

| Parameter | Used to Derive | |
|---|---|---|
| | Parameter | Tab |
| **Physical inertia, viscous damping, static friction, Mechanical** | **Proportional gain, ba**<br>**Angular gain, Ksa**<br>**Rotational gain, Kisa**<br>**Inertia compensation, Jcomp**<br>**Viscous damping compensation, Fv**<br>**Static friction, Fs** | **Speed Controller** |

#### Id and Iq Calculation

**`Rated synchronous speed, Frate` — Motor frequency**
`60` (default) | `scalar`

Motor-rated electrical frequency, $F_{rate}$, in Hz.

#### Dependencies

This table summarizes the parameter dependencies.

| Parameter | Used to Derive | |
|---|---|---|
| | Parameter | Tab |
| **Rated synchronous speed, Frate** | **D-axis rated current, Isd_0**<br>**Q-axis rated current, Isq_0**<br>**Torque at rated current, Tem** | **Id and Iq Calculation** |

**`Rated line to line voltage RMS, Vrate` — Motor voltage**
`460` (default) | `scalar`

Motor-rated line-to-line voltage, $V_{rate}$, in V.

#### Dependencies

This table summarizes the parameter dependencies.

| Parameter | Used to Derive | |
|---|---|---|
| | Parameter | Tab |
| Rated synchronous speed, Frate | D-axis rated current, Isd_0<br>Q-axis rated current, Isq_0<br>Torque at rated current, Tem | Id and Iq Calculation |

### `Rated slip, Srate` — Motor slip speed

`0.0172` (default) | `scalar`

Motor-rated slip speed, $S_{rate}$, dimensionless.

#### Dependencies

This table summarizes the parameter dependencies.

| Parameter | Used to Derive | |
|---|---|---|
| | Parameter | Tab |
| Rated slip, Srate | D-axis rated current, Isd_0<br>Q-axis rated current, Isq_0<br>Torque at rated current, Tem | Id and Iq Calculation |

### `Calculate Rated Stator Flux Current` — Derive parameters

button

Click to derive parameters.

#### Dependencies

On the **Id and Iq Calculation** tab, when you select **Calculate Rated Stator Flux Current**, the block calculates derived parameters. The table summarizes the derived parameters that depend on other block parameters.

| Derived Parameter on Id and Iq Calculation tab | Dependency | |
|---|---|---|
| | Parameter | Tab |
| D-axis rated current, Isd_0<br>Q-axis rated current, Isq_0<br>Torque at rated current, Tem | Rated synchronous speed, Frate<br>Rated line to line voltage RMS, Vrate<br>Rated slip, Srate | Id and Iq Calculation |

| Derived Parameter on Id and Iq Calculation tab | Dependency | |
|---|---|---|
| | **Parameter** | **Tab** |
| | **Stator resistance, Rs**<br>**Stator leakage inductance, Lls**<br>**Rotor resistance, Rr**<br>**Rotor leakage inductance, Llr**<br>**Rotor magnetizing inductance, Lm** | **Motor Parameters** |

**`D-axis rated current, Isd_0` — Derived**
`3.1004` (default) | `scalar`

Derived d-axis rated current, in A.

**Dependencies**

On the **Id and Iq Calculation** tab, when you select **Calculate Rated Stator Flux Current**, the block calculates derived parameters. The table summarizes the derived parameters that depend on other block parameters.

| Derived Parameter on Id and Iq Calculation tab | Dependency | |
|---|---|---|
| | **Parameter** | **Tab** |
| **D-axis rated current, Isd_0**<br>**Q-axis rated current, Isq_0**<br>**Torque at rated current, Tem** | **Rated synchronous speed, Frate**<br>**Rated line to line voltage RMS, Vrate**<br>**Rated slip, Srate** | **Id and Iq Calculation** |
| | **Stator resistance, Rs**<br>**Stator leakage inductance, Lls**<br>**Rotor resistance, Rr**<br>**Rotor leakage inductance, Llr**<br>**Rotor magnetizing inductance, Lm** | **Motor Parameters** |

**`Q-axis rated current, Isq_0` — Derived**
`5.7131` (default) | `scalar`

Derived q-axis rated current, in A.

#### Dependencies

On the **Id and Iq Calculation** tab, when you select **Calculate Rated Stator Flux Current**, the block calculates derived parameters. The table summarizes the derived parameters that depend on other block parameters.

| Derived Parameter on Id and Iq Calculation tab | Dependency | |
|---|---|---|
| | Parameter | Tab |
| D-axis rated current, Isd_0<br><br>Q-axis rated current, Isq_0<br><br>Torque at rated current, Tem | Rated synchronous speed, Frate<br><br>Rated line to line voltage RMS, Vrate<br><br>Rated slip, Srate | Id and Iq Calculation |
| | Stator resistance, Rs<br><br>Stator leakage inductance, Lls<br><br>Rotor resistance, Rr<br><br>Rotor leakage inductance, Llr<br><br>Rotor magnetizing inductance, Lm | Motor Parameters |

**`Torque at rated current, Tem` — Derived**
`12.6467` (default) | `scalar`

Torque at rated current, in N·m.

#### Dependencies

On the **Id and Iq Calculation** tab, when you select **Calculate Rated Stator Flux Current**, the block calculates derived parameters. The table summarizes the derived parameters that depend on other block parameters.

| Derived Parameter on Id and Iq Calculation tab | Dependency | |
|---|---|---|
| | Parameter | Tab |
| D-axis rated current, Isd_0<br><br>Q-axis rated current, Isq_0<br><br>Torque at rated current, Tem | Rated synchronous speed, Frate<br><br>Rated line to line voltage RMS, Vrate<br><br>Rated slip, Srate | Id and Iq Calculation |

| Derived Parameter on Id and Iq Calculation tab | Dependency | |
|---|---|---|
| | Parameter | Tab |
| | **Stator resistance, Rs**<br>**Stator leakage inductance, Lls**<br>**Rotor resistance, Rr**<br>**Rotor leakage inductance, Llr**<br>**Rotor magnetizing inductance, Lm** | **Motor Parameters** |

#### Current Controller

**`Bandwidth of the current regulator, EV_current` — Bandwidth**
`200` (default) | `scalar`

Current regulator bandwidth, in Hz.

#### Dependencies

This table summarizes the parameter dependencies.

| Parameter | Used to Derive | |
|---|---|---|
| | Parameter | Tab |
| **Bandwidth of the current regulator, EV_current** | **D and Q axis integral gain, Ki**<br>**D and Q axis proportional gain, Kp** | **Current Controller** |

**`Sample time for the torque control, Tst` — Time**
`5e-5` (default) | `scalar`

Torque control sample time, in s.

#### Dependencies

This table summarizes the parameter dependencies.

| Parameter | Used to Derive | |
|---|---|---|
| | Parameter | Tab |
| **Sample time for the torque control, Tst** | **Speed regulation time constant, Ksf** | **Speed Controller** |

**`Calculate Current Regulator Gains` — Derive parameters**
button

Click to derive parameters.

#### Dependencies

On the **Current Controller** tab, when you select **Calculate Current Regulator Gains**, the block calculates derived parameters. The table summarizes the derived parameters that depend on other block parameters.

| Derived Parameter on Current Controller tab | Dependency | |
|---|---|---|
| | **Parameter** | **Tab** |
| **D and Q axis proportional gain, Kp**<br><br>**D and Q axis integral gain, Ki** | **Bandwidth of the current regulator, EV_current** | **Current Controller** |
| | **Stator resistance, Rs**<br><br>**Stator leakage inductance, Lls**<br><br>**Rotor resistance, Rr**<br><br>**Rotor leakage inductance, Llr**<br><br>**Rotor magnetizing inductance, Lm** | **Motor Parameters** |

### `D and Q axis proportional gain, Kp` — Derived

`32.1894` (default) | `scalar`

Derived proportional gain, in V/A.

#### Dependencies

On the **Current Controller** tab, when you select **Calculate Current Regulator Gains**, the block calculates derived parameters. The table summarizes the derived parameters that depend on other block parameters.

| Derived Parameter on Current Controller tab | Dependency | |
|---|---|---|
| | **Parameter** | **Tab** |
| **D and Q axis proportional gain, Kp**<br><br>**D and Q axis integral gain, Ki** | **Bandwidth of the current regulator, EV_current** | **Current Controller** |
| | **Stator resistance, Rs**<br><br>**Stator leakage inductance, Lls**<br><br>**Rotor resistance, Rr**<br><br>**Rotor leakage inductance, Llr**<br><br>**Rotor magnetizing inductance, Lm** | **Motor Parameters** |

### `D and Q axis integral gain, Ki` — Derived

`2224.2476` (default) | `scalar`

Derived integral gain, in V/A*s.

#### Dependencies

On the **Current Controller** tab, when you select **Calculate Current Regulator Gains**, the block calculates derived parameters. The table summarizes the derived parameters that depend on other block parameters.

| Derived Parameter on Current Controller tab | Dependency | |
|---|---|---|
| | Parameter | Tab |
| D and Q axis proportional gain, Kp<br><br>D and Q axis integral gain, Ki | Bandwidth of the current regulator, EV_current | Current Controller |
| | Stator resistance, Rs<br><br>Stator leakage inductance, Lls<br><br>Rotor resistance, Rr<br><br>Rotor leakage inductance, Llr<br><br>Rotor magnetizing inductance, Lm | Motor Parameters |

#### Speed Controller

**`Bandwidth of the motion controller, EV_motion` — Bandwidth**
`[20, 4, 0.8]` (default) | `vector`

Motion controller bandwidth, in Hz. Set the first element of the vector to the desired cutoff frequency. Set the second and third elements of the vector to the higher-order cut off frequencies. You can set the value of the next element to `1/5` the value of the previous element. For example, if the desired cutoff frequency is `20` Hz, specify `[20 4 0.8]`.

#### Dependencies

The parameter is enabled when the **Control Type** parameter is set to `Speed Control`.

| Parameter | Used to Derive | |
|---|---|---|
| | Parameter | Tab |
| Bandwidth of the motion controller, EV_motion | Proportional gain, ba<br><br>Angular gain, Ksa<br><br>Rotational gain, Kisa | Speed Controller |

**`Bandwidth of the state filter, EV_sf` — Bandwidth**
`200` (default) | `scalar`

State filter bandwidth, in Hz.

#### Dependencies

The parameter is enabled when the **Control Type** parameter is set to `Speed Control`.

| Parameter | Used to Derive | |
|---|---|---|
| | Parameter | Tab |
| **Bandwidth of the state filter, EV_sf** | **Speed regulation time constant, Ksf** | **Speed Controller** |

### `Calculate Speed Regulator Gains` — Derive parameters

button

Click to derive parameters.

#### Dependencies

On the **Speed Controller** tab, when you select **Calculate Speed Regulator Gains**, the block calculates derived parameters. The table summarizes the derived parameters that depend on other block parameters.

| Derived Parameter on Speed Controller tab | | Depends On | |
|---|---|---|---|
| | | Parameter | Tab |
| **Proportional gain, ba** | $b_a = \frac{J_p - J_p p_1 p_2 p_3}{T_{sm}}$ | **Bandwidth of the motion controller, EV_motion**<br><br>**Bandwidth of the state filter, EV_sf** | **Speed Controller** |
| **Angular gain, Ksa** | $K_{sa} = \frac{J_p(p_1p_2 + p_2p_3 + p_3p_1) - 3J_p + 2b_aT_{sm}}{T_{sm}^2}$ | **Sample time for the torque control, Tst** | **Current Controller** |
| **Rotational gain, Kisa** | $K_{isa} = \frac{-J_p(p_1 + p_2 + p_3) + 3J_p - b_aT_{sm} - K_{sa}T_{sm}^2}{T_{sm}^3}$ | **Physical inertia, viscous damping, static friction, Mechanical** | **Motor Parameters** |
| **Speed regulation time constant, Ksf** | $K_{sf} = \frac{1 - \exp\left(-T_{sm}2\pi EV_{sf}\right)}{T_{sm}}$ | | |
| **Inertia compensation, Jcomp** | $J_{comp} = J_p$ | **Physical inertia, viscous damping, static friction, Mechanical** | **Motor Parameters** |
| **Viscous damping compensation, Fv** | $F_v$ | | |
| **Static friction, Fs** | $F_s$ | | |

The equations use these variables.

| | |
|---|---|
| $P$ | Motor pole pairs |
| $b_a$ | Speed regulator proportional gain |
| $K_{sa}$ | Speed regulator integral gain |
| $K_{isa}$ | Speed regulator double integral gain |
| $K_{sf}$ | Speed regulator time constant |
| $J_p$ | Motor inertia |
| $EV_{sf}$ | State filter bandwidth |
| $EV_{motion}$ | Motion controller bandwidth |

**`Proportional gain, ba` — Derived**
`3.7477` (default) | `scalar`

Derived proportional gain, in N·m/(rad/s).

**Dependencies**

This table summarizes the parameter dependencies.

| Parameter | Dependency | |
|---|---|---|
| | **Parameter** | **Tab** |
| **Proportional gain, ba** | **Physical inertia, viscous damping, static friction, Mechanical** | **Motor Parameters** |
| | **Bandwidth of the motion controller, EV_motion** | **Speed Controller** |

**`Angular gain, Ksa` — Derived**
`94.0877` (default) | `scalar`

Derived angular gain, in N·m/rad.

**Dependencies**

This table summarizes the parameter dependencies.

| Parameter | Dependency | |
|---|---|---|
| | **Parameter** | **Tab** |
| **Angular gain, Ksa** | **Physical inertia, viscous damping, static friction, Mechanical** | **Motor Parameters** |
| | **Bandwidth of the motion controller, EV_motion** | **Speed Controller** |

**`Rotational gain, Kisa` — Derived**
`381.7822` (default) | `scalar`

Derived rotational gain, in N·m/(rad*s).

**Dependencies**

This table summarizes the parameter dependencies.

| Parameter | Dependency | |
|---|---|---|
| | Parameter | Tab |
| Rotational gain, Kisa | Physical inertia, viscous damping, static friction, Mechanical | Motor Parameters |
| | Bandwidth of the motion controller, EV_motion | Speed Controller |

**`Speed regulation time constant, Ksf` — Derived**
`1217.9727` (default) | `scalar`

Derived speed regulation time constant, in 1/s.

#### Dependencies

This table summarizes the parameter dependencies.

| Parameter | Dependency | |
|---|---|---|
| | Parameter | Tab |
| Speed regulation time constant, Ksf | Sample time for the torque control, Tst | Current Controller |
| | Bandwidth of the state filter, EV_sf | Speed Controller |

**`Inertia compensation, Jcomp` — Derived**
`0.025` (default) | `scalar`

Derived inertia compensation, in kg·m^2.

#### Dependencies

This table summarizes the parameter dependencies.

| Parameter | Dependency | |
|---|---|---|
| | Parameter | Tab |
| Inertia compensation, Jcomp | Physical inertia, viscous damping, static friction, Mechanical | Motor Parameters |

**`Viscous damping compensation, Fv` — Derived**
`0` (default) | `scalar`

#### Dependencies

This table summarizes the parameter dependencies.

| Parameter | Dependency | |
|---|---|---|
| | Parameter | Tab |
| **Viscous damping compensation, Fv** | **Physical inertia, viscous damping, static friction, Mechanical** | **Motor Parameters** |

### `Static friction, Fs` — Derived

`0` (default) | `scalar`

Derived static friction, in N·m/(rad/s).

#### Dependencies

This table summarizes the parameter dependencies.

| Parameter | Dependency | |
|---|---|---|
| | Parameter | Tab |
| **Static friction, Fs** | **Physical inertia, viscous damping, static friction, Mechanical** | **Motor Parameters** |

#### Electrical Losses

### `Parameterize losses by` — Select type

`Single efficiency measurement` (default) | `Tabulated loss data` | `Tabulated efficiency data`

| Setting | Block Implementation |
|---|---|
| `Single efficiency measurement` | Electrical loss calculated using a constant value for inverter efficiency. |
| `Tabulated loss data` | Electrical loss calculated as a function of motor speeds and load torques. |
| `Tabulated efficiency data` | Electrical loss calculated using inverter efficiency that is a function of motor speeds and load torques.<br>• Converts the efficiency values you provide into losses and uses the tabulated losses for simulation.<br>• Ignores efficiency values you provide for zero speed or zero torque. Losses are assumed zero when either torque or speed is zero.<br>• Uses linear interpolation to determine losses. Provide tabulated data for low speeds and low torques, as required, to get the desired level of accuracy for lower power conditions.<br>• Does not extrapolate loss values for speed and torque magnitudes that exceed the range of the table. |

For best practice, use `Tabulated loss data` instead of `Tabulated efficiency data`:

- Efficiency becomes ill defined for zero speed or zero torque.

- You can account for fixed losses that are still present for zero speed or torque.

**`Overall inverter efficiency, eff` — Constant**
`98` (default) | `scalar`

Overall inverter efficiency, *Eff*, in %.

**Dependencies**

To enable this parameter, for **Parameterize losses by**, select `Tabulated loss data`.

**`Vector of speeds (w) for tabulated loss, w_loss_bp` — Breakpoints**
`[0 200 400 600 800 1000]` (default) | 1-by-`M` vector

Speed breakpoints for lookup table when calculating losses, in rad/s.

**Dependencies**

To enable this parameter, for **Parameterize losses by**, select `Tabulated loss data`.

**`Vector of torques (T) for tabulated loss, T_loss_bp` — Breakpoints**
`[0 25 50 75 100]` (default) | 1-by-`N` vector

Torque breakpoints for lookup table when calculating losses, in N·m.

**Dependencies**

To enable this parameter, for **Parameterize losses by**, select `Tabulated loss data`.

**`Corresponding losses, losses_table` — Table**
`[100 100 100 100 100;100 150 200 250 300;100 200 300 400 500;100 250 400 550 700;100 300 500 700 900;100 350 600 850 1100]` (default) | `M`-by-`N` array

Array of values for electrical losses as a function of `M` speeds and `N` torques, in W. Each value specifies the losses for a specific combination of speed and torque. The matrix size must match the dimensions defined by the speed and torque vectors.

**Dependencies**

To enable this parameter, for **Parameterize losses by**, select `Tabulated loss data`.

**`Vector of speeds (w) for tabulated efficiency, w_eff_bp` — Breakpoints**
`[200 400 600 800 1000]` (default) | 1-by-`M` vector

Speed breakpoints for lookup table when calculating efficiency, in rad/s.

**Dependencies**

To enable this parameter, for **Parameterize losses by**, select `Tabulated efficiency data`.

**`Vector of torques (T) for tabulated efficiency, T_eff_bp` — Breakpoints**
`[25 50 75 100]` (default) | 1-by-`N` vector

Torque breakpoints for lookup table when calculating efficiency, in N·m.

**Dependencies**

To enable this parameter, for **Parameterize losses by**, select `Tabulated efficiency data`.

**Corresponding efficiency, efficiency_table — Table**
`[96.2 98.1 98.7 99;98.1 99 99.4 99.5;98.7 99.4 99.6 99.7;99 99.5 99.7 99.8;99.2 99.6 99.7 99.8]` (default) | `M`-by-`N` array

Array of efficiency as a function of `M` speeds and `N` torque, in %. Each value specifies the efficiency for a specific combination of speed and torque. The matrix size must match the dimensions defined by the speed and torque vectors.

The block ignores efficiency values for zero speed or zero torque. Losses are zero when either torque or speed is zero. The block uses linear interpolation.

To get the desired level of accuracy for lower power conditions, you can provide tabulated data for low speeds and low torques.

**Dependencies**

To enable this parameter, for **Parameterize losses by**, select `Tabulated efficiency data`.

## References

[1] Lorenz, Robert D., Thomas Lipo, and Donald W. Novotny. “Motion control with induction motors.” *Proceedings of the IEEE*, Vol. 82, Issue 8, August 1994, pp. 1215–1240.

[2] Shigeo Morimoto, Masayuka Sanada, Yoji Takeda. “Wide-speed operation of interior permanent magnet synchronous motors with high-performance current regulator.” *IEEE Transactions on Industry Applications*, Vol. 30, Issue 4, July/August 1994, pp. 920–926.

[3] Muyang Li. “Flux-Weakening Control for Permanent-Magnet Synchronous Motors Based on Z-Source Inverters.” Master’s Thesis, Marquette University, e-Publications@Marquette, Fall 2014.

[4] Briz, Fernando, Michael W. Degner, and Robert D. Lorenz. "Analysis and design of current regulators using complex vectors." *IEEE Transactions on Industry Applications*, Vol. 36, Issue 3, May/June 2000, pp. 817–825.

[5] Briz, Fernando, et al. "Current and flux regulation in field-weakening operation [of induction motors]."*IEEE Transactions on Industry Applications*, Vol. 37, Issue 1, Jan/Feb 2001, pp. 42–50.

## Extended Capabilities

**C/C++ Code Generation**
Generate C and C++ code using Simulink® Coder™.

## See Also

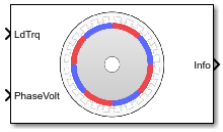
Flux-Based PM Controller | Induction Motor | Interior PM Controller | Surface Mount PM Controller

**Introduced in R2017a**

# Surface Mount PMSM

Three-phase exterior permanent magnet synchronous motor with sinusoidal back electromotive force

**Library:** Powertrain Blockset / Propulsion / Electric Motors and Inverters
Motor Control Blockset / Electrical Systems / Motors




## Description

The Surface Mount PMSM block implements a three-phase exterior permanent magnet synchronous motor (PMSM) with sinusoidal back electromotive force. The block uses the three-phase input voltages to regulate the individual phase currents, allowing control of the motor torque or speed.

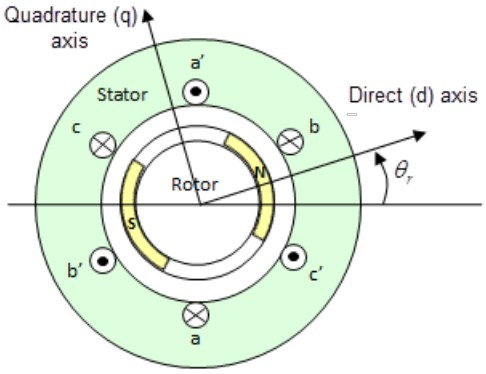
By default, the block sets the **Simulation type** parameter to `Continuous` to use a continuous sample time during simulation. If you want to generate code for fixed-step double- and single-precision targets, considering setting the parameter to `Discrete`. Then specify a **Sample Time, Ts** parameter.

On the **Parameters** tab, if you select `Back-emf` or `Torque constant`, the block implements one of these equations to calculate the permanent flux linkage constant.

| Setting | Equation |
|---|---|
| `Back-emf` | $\lambda_{pm} = \frac{1}{\sqrt{3}} \cdot \frac{K_e}{1000P} \cdot \frac{60}{2\pi}$ |
| `Torque constant` | $\lambda_{pm} = \frac{2}{3} \cdot \frac{K_t}{P}$ |

### Motor Construction

This figure shows the motor construction with a single pole pair on the motor.



The motor magnetic field due to the permanent magnets creates a sinusoidal rate of change of flux with motor angle.

For the axes convention, the *a*-phase and permanent magnet fluxes are aligned when motor angle $\theta_r$ is zero.

### Three-Phase Sinusoidal Model Electrical System

The block implements these equations, expressed in the motor flux reference frame (dq frame). All quantities in the motor reference frame are referred to the stator.

$$\omega_e = P\omega_m$$

$$\frac{d}{dt}i_d = \frac{1}{L_d}v_d - \frac{R}{L_d}i_d + \frac{L_q}{L_d}P\omega_m i_q$$

$$\frac{d}{dt}i_q = \frac{1}{L_q}v_q - \frac{R}{L_q}i_q - \frac{L_d}{L_q}P\omega_m i_d - \frac{\lambda_{pm}P\omega_m}{L_q}$$

$$T_e = 1.5P[\lambda_{pm}i_q + (L_d - L_q)i_d i_q]$$

The $L_q$ and $L_d$ inductances represent the relation between the phase inductance and the motor position due to the saliency of the motor magnets. For the surface mount PMSM, $L_d = L_q$.

The equations use these variables.

| | |
|---|---|
| $L_q$, $L_d$ | q- and d-axis inductances |
| $R$ | Resistance of the stator windings |
| $i_q$, $i_d$ | q- and d-axis currents |
| $v_q$, $v_d$ | q- and d-axis voltages |
| $\omega_m$ | Angular mechanical velocity of the motor |
| $\omega_e$ | Angular electrical velocity of the motor |
| $\lambda_{pm}$ | Permanent magnet flux linkage |
| $K_e$ | Back electromotive force (EMF) |
| $K_t$ | Torque constant |
| $P$ | Number of pole pairs |
| $T_e$ | Electromagnetic torque |
| $\Theta_e$ | Electrical angle |

### Mechanical System

The motor angular velocity is given by:

$$\frac{d}{dt}\omega_m = \frac{1}{J}\left(T_e - T_f - F\omega_m - T_m\right)$$

$$\frac{d\theta_m}{dt} = \omega_m$$

The equations use these variables.

| | |
|---|---|
| $J$ | Combined inertia of motor and load |
| $F$ | Combined viscous friction of motor and load |
| $\theta_m$ | Motor mechanical angular position |
| $T_m$ | Motor shaft torque |
| $T_e$ | Electromagnetic torque |
| $T_f$ | Motor shaft static friction torque |
| $\omega_m$ | Angular mechanical velocity of the motor |

### Power Accounting

For the power accounting, the block implements these equations.

| Bus Signal | | | Description | Variable | Equations |
|---|---|---|---|---|---|
| `PwrInfo` | `PwrTrnsfrd` — Power transferred between blocks<br>• Positive signals indicate flow into block<br>• Negative signals indicate flow out of block | `PwrMtr` | Mechanical power | $P_{mot}$ | $P_{mot} = -\omega_m T_e$ |
| | | `PwrBus` | Electrical power | $P_{bus}$ | $P_{bus} = v_{an} i_a + v_{bn} i_b + v_{cn} i_c$ |
| | `PwrNotTrnsfrd` — Power crossing the block boundary, but not transferred<br>• Positive signals indicate an input<br>• Negative signals indicate a loss | `PwrElecLoss` | Resistive power loss | $P_{elec}$ | $P_{elec} = -\frac{3}{2}(R_s i_{sd}^2 + R_s i_{sq}^2)$ |
| | | `PwrMechLoss` | Mechanical power loss | $P_{mech}$ | When **Port Configuration** is set to `Torque`:<br>$P_{mech} = -\left(\omega_m^2 F + \lvert\omega_m\rvert T_f\right)$<br>When **Port Configuration** is set to `Speed`:<br>$P_{mech} = 0$ |
| | `PwrStored` — Stored energy rate of change<br>• Positive signals indicate an increase<br>• Negative signals indicate a decrease | `PwrMtrStored` | Stored motor power | $P_{str}$ | $P_{str} = P_{bus} + P_{mot} + P_{elec} + P_{mech}$ |

The equations use these variables.

| | |
|---|---|
| $R_s$ | Stator resistance |

| | |
|---|---|
| $i_a$, $i_b$, $i_c$ | Stator phase a, b, and c current |
| $i_{sq}$, $i_{sd}$ | Stator q- and d-axis currents |
| $v_{an}$, $v_{bn}$, $v_{cn}$ | Stator phase a, b, and c voltage |
| $\omega_m$ | Angular mechanical velocity of the motor |
| $F$ | Combined motor and load viscous damping |
| $T_e$ | Electromagnetic torque |
| $T_f$ | Combined motor and load friction torque |

## Ports

### Input

#### `LdTrq` — Motor shaft torque

`scalar`

Motor shaft input torque, $T_m$, in N·m.

##### Dependencies

To create this port, select `Torque` for the **Port Configuration** parameter.

#### Spd — Motor shaft speed

`scalar`

Angular velocity of the motor, $\omega_m$, in rad/s.

##### Dependencies

To create this port, select `Speed` for the **Port Configuration** parameter.

#### `PhaseVolt` — Stator terminal voltages

1-by-3 array

Stator terminal voltages, $V_a$, $V_b$, and $V_c$, in V.

### Output

#### `Info` — Bus signal

bus

The bus signal contains these block calculations.

| Signal | Description | Variable | Units |
|---|---|---|---|
| `IaStator` | Stator phase current A | $i_a$ | A |
| `IbStator` | Stator phase current B | $i_b$ | A |
| `IcStator` | Stator phase current C | $i_c$ | A |
| `IdSync` | Direct axis current | $i_d$ | A |
| `IqSync` | Quadrature axis current | $i_q$ | A |
| `VdSync` | Direct axis voltage | $v_d$ | V |

| Signal | | | Description | Variable | Units |
|---|---|---|---|---|---|
| VqSync | | | Quadrature axis voltage | $v_q$ | V |
| MtrSpd | | | Angular mechanical velocity of the motor | $\omega_m$ | rad/s |
| MtrPos | | | Motor mechanical angular position | $\theta_m$ | rad |
| MtrTrq | | | Electromagnetic torque | $T_e$ | N·m |
| PwrInfo | PwrTrnsfrd | PwrMtr | Mechanical power | $P_{mot}$ | W |
| | | PwrBus | Electrical power | $P_{bus}$ | W |
| | PwrNotTrnsfrd | PwrElecLoss | Resistive power loss | $P_{elec}$ | W |
| | | PwrMechLoss | Mechanical power loss | $P_{mech}$ | W |
| | PwrStored | PwrMtrStored | Stored motor power | $P_{str}$ | W |

**PhaseCurr — Phase a, b, c current**
1-by-3 array

Phase a, b, c current, $i_a$, $i_b$, and $i_c$, in A.

**MtrTrq — Motor torque**
scalar

Motor torque, $T_{mtr}$, in N·m.

**Dependencies**

To create this port, select Speed for the **Mechanical input configuration** parameter.

**MtrSpd — Motor speed**
scalar

Angular speed of the motor, $\omega_{mtr}$, in rad/s.

**Dependencies**

To create this port, select Torque for the **Mechanical input configuration** parameter.

## Parameters

**Block Options**

**Mechanical input configuration — Select port configuration**
Torque (default) | Speed

This table summarizes the port configurations.

| Port Configuration | Creates Input Port | Creates Output Port |
|---|---|---|
| Torque | LdTrq | MtrSpd |

| Port Configuration | Creates Input Port | Creates Output Port |
|---|---|---|
| Speed | Spd | MtrTrq |

**Simulation type — Select simulation type**
Continuous (default) | Discrete

By default, the block uses a continuous sample time during simulation. If you want to generate code for single-precision targets, considering setting the parameter to Discrete.

**Dependencies**

Setting **Simulation type** to Discrete creates the **Sample Time, Ts** parameter.

**Sample Time (Ts) — Sample time for discrete integration**
scalar

Integration sample time for discrete simulation, in s.

**Dependencies**

Setting **Simulation type** to Discrete creates the **Sample Time, Ts** parameter.

**Parameters**

**Number of pole pairs (P) — Pole pairs**
scalar

Motor pole pairs, *P*.

**Stator phase resistance per phase (Rs) — Resistance**
scalar

Stator phase resistance per phase, $R_s$, in ohm.

**Stator d-axis inductance (Ldq_) — Inductance**
scalar

Stator inductance, $L_{dq}$, in H.

**Permanent flux linkage constant (lambda_pm) — Flux**
scalar

Permanent flux linkage constant, $\lambda_{pm}$, in Wb.

**Back-emf constant (Ke) — Back electromotive force**
scalar

Back electromotive force, EMF, $K_e$, in peak Vpk_LL/krpm. Vpk_LL is the peak voltage line-to-line measurement.

To calculate the permanent flux linkage constant, the block implements this equation.

$$\lambda_{pm} = \frac{1}{\sqrt{3}} \cdot \frac{K_e}{1000P} \cdot \frac{60}{2\pi}$$

**`Torque constant (Kt)` — Torque constant**
`scalar`

Torque constant, $K_t$, in N·m/A.

To calculate the permanent flux linkage constant, the block implements this equation.

$$\lambda_{pm} = \frac{2}{3} \cdot \frac{K_t}{P}$$

**`Physical inertia, viscous damping, and static friction (mechanical)` — Inertia, damping, friction**
`vector`

Mechanical properties of the motor:

- Inertia, $J$, in kgm^2
- Viscous damping, $F$, in N·m/(rad/s)
- Static friction, $T_f$, in N·m

#### Dependencies

To enable this parameter, select the `Torque` configuration parameter.

#### Initial Values

**`Initial d-axis and q-axis current (idq0)` — Current**
`vector`

Initial q- and d-axis currents, $i_q$, $i_d$, in A.

**`Initial mechanical position (theta_init)` — Angle**
`scalar`

Initial motor angular position, $\theta_{m0}$, in rad.

**`Initial mechanical speed (omega_init)` — Speed**
`scalar`

Initial angular velocity of the motor, $\omega_{m0}$, in rad/s.

#### Dependencies

To enable this parameter, select the `Torque` configuration parameter.

## References

[1] Kundur, P. *Power System Stability and Control.* New York, NY: McGraw Hill, 1993.

[2] Anderson, P. M. *Analysis of Faulted Power Systems.* Hoboken, NJ: Wiley-IEEE Press, 1995.

## Extended Capabilities

#### C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

## See Also

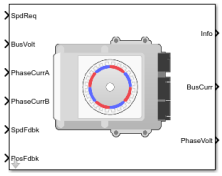
Flux-Based PMSM | Induction Motor | Interior PMSM | Mapped Motor | Surface Mount PM Controller

**Introduced in R2017a**

# Surface Mount PM Controller

Torque-based, field-oriented controller for a surface mount permanent magnet synchronous motor

**Library:** Powertrain Blockset / Propulsion / Electric Motors and Inverters
Motor Control Blockset / Electrical Systems / Motors

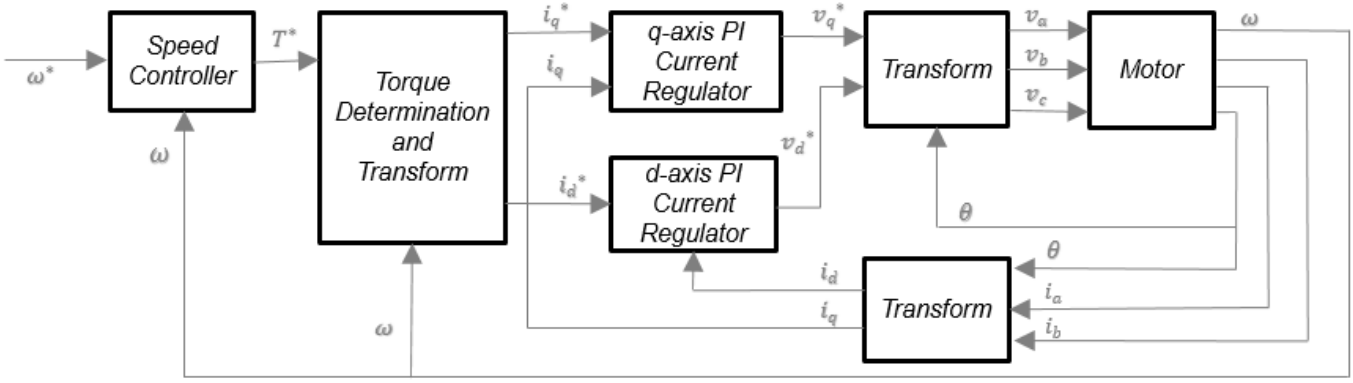



## Description

The Surface Mount PM Controller block implements a torque-based, field-oriented controller for a surface mount permanent magnet synchronous motor (PMSM) with an optional outer-loop speed controller. The torque control utilizes quadrature current and does not weaken the magnetic flux. You can specify either speed or torque control.

The Surface Mount PM Controller implements equations for speed control, torque determination, regulators, transforms, and motors.

The figure illustrates the information flow in the block.




The block implements equations that use these variables.

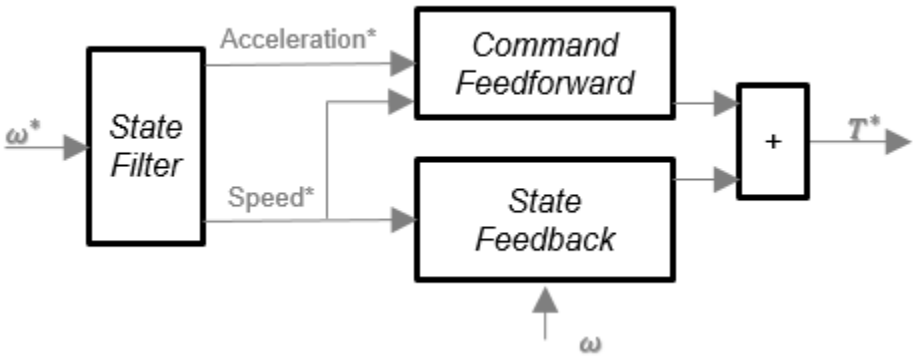
| | |
|---|---|
| $\omega$ | Rotor speed |
| $\omega$* | Rotor speed command |
| $T$* | Torque command |
| $i_d$ | d-axis current |
| $i^*_d$ | d-axis current command |
| $i_q$ | q-axis current |
| $i^*_q$ | q-axis current command |

| | |
|---|---|
| $v_d$, | d-axis voltage |
| $v^*_d$ | d-axis voltage command |
| $v_q$ | q-axis voltage |
| $v^*_q$ | q-axis voltage command |
| $v_a$, $v_b$, $v_c$ | Stator phase a, b, c voltages |
| $i_a$, $i_b$, $i_c$ | Stator phase a, b, c currents |

### Speed Controller

To implement the speed controller, select the **Control Type** parameter `Speed Control`. If you select the **Control Type** parameter `Torque Control`, the block does not implement the speed controller.

The speed controller determines the torque command by implementing a state filter, and calculating the feedforward and feedback commands. If you do not implement the speed controller, input a torque command to the Surface Mount PM Controller block.




#### State Filter

The state filter is a low-pass filter that generates the acceleration command based on the speed command. On the **Speed Controller** tab:

- To make the speed-command lag time negligible, specify a **Bandwidth of the state filter** parameter.
- To calculate a **Speed regulation time constant, Ksf** gain based on the state filter bandwidth, select **Calculate Speed Regulator Gains**.

The discrete form of characteristic equation is given by:

$$z + K_{sf}T_{sm} - 1$$

The filter calculates the gain using this equation.

$$K_{sf} = \frac{1 - \exp\left(-T_{sm}2\pi EV_{sf}\right)}{T_{sm}}$$

The equations use these variables.

| | |
|---|---|
| $EV_{sf}$ | Bandwidth of the speed command filter |
| $T_{sm}$ | Motion controller sample time |
| $K_{sf}$ | Speed regulator time constant |

#### State Feedback

To generate the state feedback torque, the block uses the filtered speed error signal from the state filter. The feedback torque calculation also requires gains for speed regulator.

On the **Speed Controller** tab, select **Calculate Speed Regulator Gains** to calculate:

- **Proportional gain, ba**
- **Angular gain, Ksa**
- **Rotational gain, Kisa**

For the gain calculations, the block uses the inertia from the **Physical inertia, viscous damping, static friction** parameter value on the **Motor Parameters** tab.

The gains for the state feedback are calculated using these equations.

| Calculation | Equations |
|---|---|
| Discrete forms of characteristic equation | $z^3 + \frac{\left(-3J_p + T_s b_a + \quad T_s^2 K_{sa} + T_s^3 K_{isa}\right)}{J_p} z^2 + \frac{\left(3J_p - \quad 2T_s b_a - \quad T_s^2 K_{sa}\right)}{J_p} z + \frac{-J_p + T_s b_a}{J_p}$<br><br>$(z - p_1)(z - p_2)(z - p_3) = z^3 + (p_1 + p_2 + p_3)z^2 + (p_1 p_2 + p_2 p_3 + p_1 3)z^2 - p_1 p_2 p_3$ |
| Speed regulator proportional gain | $b_a = \frac{J_p - J_p p_1 p_2 p_3}{T_{sm}}$ |
| Speed regulator integral gain | $K_{sa} = \frac{J_p(p_1 p_2 + p_2 p_3 + p_3 p_1) - 3J_p + 2b_a T_{sm}}{T_{sm}^2}$ |
| Speed regulator double integral gain | $K_{isa} = \frac{-J_p(p_1 + p_2 + p_3) + 3J_p - b_a T_{sm} - K_{sa} T_{sm}^2}{T_{sm}^3}$ |

The equations use these variables.

| | |
|---|---|
| $P$ | Motor pole pairs |
| $b_a$ | Speed regulator proportional gain |
| $K_{sa}$ | Speed regulator integral gain |
| $K_{isa}$ | Speed regulator double integral gain |
| $J_p$ | Motor inertia |
| $T_{sm}$ | Motion controller sample time |

#### Command Feedforward

To generate the state feedforward torque, the block uses the filtered speed and acceleration from the state filter. Also, the feedforward torque calculation uses the inertia, viscous damping, and static friction. To achieve zero tracking error, the torque command is the sum of the feedforward and feedback torque commands.

Selecting **Calculate Speed Regulator Gains** on the **Speed Controller** tab updates the inertia, viscous damping, and static friction with the **Physical inertia, viscous damping, static friction** parameter values on the **Motor Parameters** tab.

The feedforward torque command uses this equation.

$$T_{cmd_ff} = J_p\dot{\omega}_m + F_v\omega_m + F_s\frac{\omega_m}{|\omega_m|}$$

The equation uses these variables.

| | |
|---|---|
| $J_p$ | Motor inertia |
| $T_{cmd_ff}$ | Torque command feedforward |
| $F_s$ | Static friction torque constant |
| $F_v$ | Viscous friction torque constant |
| $F_s$ | Static friction torque constant |
| $\omega_m$ | Rotor speed |

### Torque Determination

The block uses a quadrature current to determine the base speed and the current commands. The available bus voltage determines the base speed. The direct (d) and quadrature (q) permanent magnet (PM) determines the induced voltage.

| Calculation | Equations |
|---|---|
| Motor maximum torque | $T_{max} = \frac{3}{2}P\left(\lambda_{pm}i_q + \left(L_d - L_q\right)i_d i_q\right)$ |
| Maximum q-axis phase current | $i_{q_max} = \frac{T_{cmd}}{\frac{3}{2}P\lambda_{pm}}$ |
| Electrical base speed | $\omega_{base} = \frac{v_{max}}{\sqrt{\left(L_q i_q\right)^2 + \left(\lambda_{pm}\right)^2}}$ |
| d-axis voltage | $v_d = -\omega_e L_q i_{q_max}$ |
| q-axis voltage | $v_q = \omega_e\lambda_{pm}$ |
| Maximum phase current | $i_{max} = \left\|i_{q_max}\right\|$ |
| Maximum voltage | $v_{max} = \frac{v_{bus}}{\sqrt{3}}$ |

| Calculation | Equations |
|---|---|
| Current command | $i_{dref} = 0$<br>$i_{q_tmp} = \min(i_{q_max}, \frac{T_{cmd}}{\frac{3}{2}P\lambda_{pm}})$<br>If $\lvert\omega_e\rvert \le \omega_{base}$<br>$i_{qref} = i_{q_tmp}$<br>Else<br>$i_{qfw} = sqrt(\min(0, \frac{1}{L_q}((\frac{v_{max}}{\omega_e})^2 - (\lambda_{pm})^2))$<br>If $i_{q_tmp} < i_{qfw}$<br>$i_{qref} = i_{q_tmp}$<br>Else<br>$i_{qref} = i_{qfw}$<br>End<br>End |

The equations use these variables.

| | |
|---|---|
| $i_{max}$ | Maximum phase current |
| $i_d$ | d-axis current |
| $i_q$ | q-axis current |
| $i_{dref}$ | d-axis reference current |
| $i_{qref}$ | q-axis reference current |
| $i_{q_max}$ | Maximum q-axis phase current |
| $\omega_e$ | Rotor electrical speed |
| $\lambda_{pm}$ | Permanent magnet flux linkage |
| $v_d$ | d-axis voltage |
| $v_q$ | q-axis voltage |
| $v_{max}$ | Maximum line to neutral voltage |
| $v_{bus}$ | DC bus voltage |
| $L_d$ | d-axis winding inductance |
| $L_q$ | q-axis winding inductance |
| $P$ | Motor pole pairs |
| $T_{max}$ | Motor maximum torque |
| $T_{cmd}$ | Commanded motor maximum torque |

### Current Regulators

The block regulates the current with an anti-windup feature. Classic proportional-integrator (PI) current regulators do not consider the d-axis and q-axis coupling or the back-electromagnetic force (EMF) coupling. As a result, transient performance deteriorates. To account for the coupling, the block implements the complex vector current regulator (CVCR) in the scalar format of the rotor reference frame. The CVCR decouples:

- d-axis and q-axis current cross-coupling

- back-EMF cross-coupling

The current frequency response is a first-order system, with a bandwidth of $EV_{current}$.

The block implements these equations.

| Calculation | Equations |
|---|---|
| Motor voltage, in the rotor reference frame | $L_d\frac{di_d}{dt} = v_d - R_s i_d + p\omega_m L_q i_q$<br>$L_d\frac{di_q}{dt} = v_q - R_s i_q - p\omega_m L_d i_d - p\omega_m \lambda_{pm}$ |
| Current regulator gains | $\omega_b = 2\pi EV_{current}$<br>$K_{p_d} = L_d\omega_b$<br>$K_{p_q} = L_q\omega_b$<br>$K_i = R_s\omega_b$ |
| Transfer functions | $\frac{i_d}{i_{dref}} = \frac{\omega_b}{s+\omega_b}$<br>$\frac{i_q}{i_{qref}} = \frac{\omega_b}{s+\omega_b}$ |

The equations use these variables.

| | |
|---|---|
| $EV_{current}$ | Current regulator bandwidth |
| $i_d$ | d-axis current |
| $i_q$ | q-axis current |
| $K_{p_d}$ | Current regulator d-axis gain |
| $K_{p_q}$ | Current regulator q-axis gain |
| $K_i$ | Current regulator integrator gain |
| $L_d$ | d-axis winding inductance |
| $L_q$ | q-axis winding inductance |
| $R_s$ | Stator phase winding resistance |
| $\omega_m$ | Rotor speed |
| $v_d$ | d-axis voltage |
| $v_q$ | q-axis voltage |
| $\lambda_{pm}$ | Permanent magnet flux linkage |
| $P$ | Motor pole pairs |

### Transforms

To calculate the voltages and currents in balanced three-phase ($a$, $b$) quantities, quadrature two-phase ($\alpha$, $\beta$) quantities, and rotating ($d$, $q$) reference frames, the block uses the Clarke and Park Transforms.

In the transform equations.

$$\omega_e = P\omega_m$$

$$\frac{d\theta_e}{dt} = \omega_e$$

| Transform | Description | Equations |
|---|---|---|
| Clarke | Converts balanced three-phase quantities ($a$, $b$) into balanced two-phase quadrature quantities ($\alpha$, $\beta$). | $x_\alpha = \frac{2}{3}x_a - \frac{1}{3}x_b - \frac{1}{3}x_c$<br>$x_\beta = \frac{\sqrt{3}}{2}x_b - \frac{\sqrt{3}}{2}x_c$ |
| Park | Converts balanced two-phase orthogonal stationary quantities ($\alpha$, $\beta$) into an orthogonal rotating reference frame ($d$, $q$). | $x_d = x_\alpha\cos\theta_e + x_\beta\sin\theta_e$<br>$x_q = -x_\alpha\sin\theta_e + x_\beta\cos\theta_e$ |
| Inverse Clarke | Converts balanced two-phase quadrature quantities ($\alpha$, $\beta$) into balanced three-phase quantities ($a$, $b$). | $x_a = x_a$<br>$x_b = -\frac{1}{2}x_\alpha + \frac{\sqrt{3}}{2}x_\beta$<br>$x_c = -\frac{1}{2}x_\alpha - \frac{\sqrt{3}}{2}x_\beta$ |
| Inverse Park | Converts an orthogonal rotating reference frame ($d$, $q$) into balanced two-phase orthogonal stationary quantities ($\alpha$, $\beta$). | $x_\alpha = x_d\cos\theta_e - x_q\sin\theta_e$<br>$x_\beta = x_d\sin\theta_e + x_q\cos\theta_e$ |

The transforms use these variables.

| | |
|---|---|
| $\omega_m$ | Rotor speed |
| $P$ | Motor pole pairs |
| $\omega_e$ | Rotor electrical speed |
| $\Theta_e$ | Rotor electrical angle |
| $x$ | Phase current or voltage |

### Motor

The block uses the phase currents and phase voltages to estimate the DC bus current. Positive current indicates battery discharge. Negative current indicates battery charge. The block uses these equations.

| | |
|---|---|
| Load power | $Ld_{Pwr} = v_a\ i_a + v_b\ i_b + v_c\ i_c$ |
| Source power | $Src_{Pwr} = Ld_{Pwr} + Pwr_{Loss}$ |
| DC bus current | $i_{bus} = \frac{Src_{Pwr}}{v_{bus}}$ |
| Estimated rotor torque | $MtrTrq_{est} = 1.5P[\lambda i_q + (L_d - L_q)i_d i_q]$ |
| Power loss for single efficiency source to load | $Pwr_{Loss} = \frac{100 - Eff}{Eff} \cdot Ld_{Pwr}$ |
| Power loss for single efficiency load to source | $Pwr_{Loss} = \frac{100 - Eff}{100} \cdot \lvert Ld_{Pwr}\rvert$ |

| | |
|---|---|
| Power loss for tabulated efficiency | $Pwr_{Loss} = \quad f(\omega_m, MtrTrq_{est})$ |

The equations use these variables.

| | |
|---|---|
| $v_a$, $v_b$, $v_c$ | Stator phase a, b, c voltages |
| $v_{bus}$ | Estimated DC bus voltage |
| $i_a$, $i_b$, $i_c$ | Stator phase a, b, c currents |
| $i_{bus}$ | Estimated DC bus current |
| $Eff$ | Overall inverter efficiency |
| $\omega_m$ | Rotor mechanical speed |
| $L_q$ | q-axis winding inductance |
| $L_d$ | d-axis winding inductance |
| $i_q$ | q-axis current |
| $i_d$ | d-axis current |
| $\lambda$ | Permanent magnet flux linkage |
| $P$ | Motor pole pairs |

### Electrical Losses

To specify the electrical losses, on the **Electrical Losses** tab, for **Parameterize losses by**, select one of these options.

| Setting | Block Implementation |
|---|---|
| `Single efficiency measurement` | Electrical loss calculated using a constant value for inverter efficiency. |
| `Tabulated loss data` | Electrical loss calculated as a function of motor speeds and load torques. |
| `Tabulated efficiency data` | Electrical loss calculated using inverter efficiency that is a function of motor speeds and load torques.<br>• Converts the efficiency values you provide into losses and uses the tabulated losses for simulation.<br>• Ignores efficiency values you provide for zero speed or zero torque. Losses are assumed zero when either torque or speed is zero.<br>• Uses linear interpolation to determine losses. Provide tabulated data for low speeds and low torques, as required, to get the desired level of accuracy for lower power conditions.<br>• Does not extrapolate loss values for speed and torque magnitudes that exceed the range of the table. |

For best practice, use `Tabulated loss data` instead of `Tabulated efficiency data`:

- Efficiency becomes ill defined for zero speed or zero torque.
- You can account for fixed losses that are still present for zero speed or torque.

## Ports

### Input

#### `SpdReq` — Rotor speed command

`scalar`

Rotor speed command, $\omega^*_m$, in rad/s.

##### Dependencies

To create this port, select `Speed Control` for the **Control Type** parameter.

#### `TrqCmd` — Torque command

`scalar`

Torque command, $T^*$, in N·m.

##### Dependencies

To create this port, select `Torque Control` for the **Control Type** parameter.

#### `BusVolt` — DC bus voltage

`scalar`

DC bus voltage $v_{bus}$, in V.

#### `PhaseCurrA` — Current

`scalar`

Stator current phase a, $i_a$, in A.

#### `PhaseCurrB` — Current

`scalar`

Stator current phase b, $i_b$, in A.

#### `SpdFdbk` — Rotor speed

`scalar`

Rotor speed, $\omega_m$, in rad/s.

#### `PosFdbk` — Rotor electrical angle

`scalar`

Rotor electrical angle, $\Theta_m$, in rad.

### Output

#### `Info` — Bus signal

bus

Bus signal containing these block calculations.

| Signal | Description | Units |
|---|---|---|
| SrcPwr | Source power | W |
| LdPwr | Load power | W |
| PwrLoss | Power loss | W |
| MtrTrqEst | Estimated motor torque | N·m |

**BusCurr — Bus current**
scalar

Estimated DC bus current, $i_{bus}$, in A.

**PhaseVolt — Stator terminal voltages**
array

Stator terminal voltages, $V_a$, $V_b$, and $V_c$, in V.

## Parameters

### Configuration

**Control Type — Select control**
Speed Control (default) | Torque Control

If you select Torque Control, the block does not implement the speed controller.

This table summarizes the port configurations.

| Port Configuration | Creates Ports |
|---|---|
| Speed Control | SpdReq |
| Torque Control | TrqCmd |

### Motor Parameters

**Stator resistance, Rs — Resistance**
0.02 (default) | scalar

Stator phase winding resistance, $R_s$, in ohm.

#### Dependencies

This table summarizes the parameter dependencies.

| Parameter | Used to Derive | |
|---|---|---|
| | Parameter | Tab |
| **Stator resistance, Rs** | **D and Q axis integral gain, Ki** | **Current Controller** |

**DQ axis inductance, Ldq — Inductance**
1.7e-3 (default) | scalar

D-axis winding inductance, $L_{dq}$, in H.

#### Dependencies

This table summarizes the parameter dependencies.

| Parameter | Used to Derive | |
|---|---|---|
| | Parameter | Tab |
| **DQ axis inductance, Ldq** | **D-axis proportional gain, Kp_d**<br>**Q-axis proportional gain, Kp_q**<br>**D and Q axis integral gain, Ki** | **Current Controller** |

**`Permanent magnet flux, lambda_pm` — Flux**
`0.2205` (default) | `scalar`

Permanent magnet flux, $\lambda_{pm}$, in Wb.

**`Number of pole pairs, PolePairs` — Poles**
`4` (default) | `scalar`

Motor pole pairs, $P$.

**`Physical inertia, viscous damping, static friction, Mechanical` — Inertia, damping, friction**
`[0.0027, 4.924e-4, 0]` (default) | `vector`

Mechanical properties of the motor:

- Motor inertia, $F_v$, in kgm^2
- Viscous friction torque constant, $F_v$, in N·m/(rad/s)
- Static friction torque constant, $F_s$, in N·m

#### Dependencies

To enable this parameter, set the **Control Type** parameter to `Speed Control`.

For the gain calculations, the block uses the inertia from the **Physical inertia, viscous damping, static friction** parameter value that is on the **Motor Parameters** tab.

This table summarizes the parameter dependencies.

| Parameter | Used to Derive | |
|---|---|---|
| | Parameter | Tab |
| Physical inertia, viscous damping, static friction, Mechanical | Proportional gain, ba<br>Angular gain, Ksa<br>Rotational gain, Kisa<br>Inertia compensation, Jcomp<br>Viscous damping compensation, Fv<br>Static friction, Fs | Speed Controller |

#### Id and Iq Calculation

**`Maximum torque, T_max` — Torque**
`60` (default) | `scalar`

Maximum torque, in N·m.

#### Current Controller

**`Bandwidth of the current regulator, EV_current` — Bandwidth**
`200` (default) | `scalar`

Current regulator bandwidth, in Hz.

#### Dependencies

This table summarizes the parameter dependencies.

| Parameter | Used to Derive | |
|---|---|---|
| | Parameter | Tab |
| Bandwidth of the current regulator, EV_current | D-axis proportional gain, Kp_d<br>Q-axis proportional gain, Kp_q<br>D and q axis proportional gain, Ki | Current Controller |

**`Sample time for the torque control, Tst` — Time**
`5e-5` (default) | `scalar`

Torque control sample time, in s.

#### Dependencies

This table summarizes the parameter dependencies.

| Parameter | Used to Derive | |
|---|---|---|
| | Parameter | Tab |
| Sample time for the torque control, Tst | Speed regulation time constant, Ksf | Speed Controller |

### `Calculate Current Regulator Gains` — Derive parameters

button

Click to derive parameters.

#### Dependencies

On the **Current Controller** tab, when you select **Calculate Current Regulator Gains**, the block calculates derived parameters. The table summarizes the derived parameters that depend on other block parameters.

| Derived Parameter on Current Controller tab | Dependency | |
|---|---|---|
| | Parameter | Tab |
| D-axis proportional gain, Kp_d<br><br>Q-axis proportional gain, Kp_q<br><br>D and Q axis integral gain, Ki | Bandwidth of the current regulator, EV_current | Current Controller |
| | Stator resistance, Rs<br><br>DQ-axis inductance, Ldq | Motor Parameters |

### `D-axis proportional gain, Kp_d` — Derived

`0.47149` (default) | `scalar`

Derived d-axis proportional gain, in V/A.

#### Dependencies

This table summarizes the parameter dependencies.

| Parameter | Dependency | |
|---|---|---|
| | Parameter | Tab |
| D-axis proportional gain, Kp_d | Bandwidth of the current regulator, EV_current | Current Controller |
| | DQ-axis inductance, Ldq | Motor Parameters |

### `Q-axis proportional gain, Kp_q` — Derived

`0.52125` (default) | `scalar`

Derived q-axis proportional gain, in V/A.

#### Dependencies

This table summarizes the parameter dependencies.

| Parameter | Dependency | |
|---|---|---|
| | Parameter | Tab |
| Q-axis proportional gain, Kp_q | Bandwidth of the current regulator, EV_current | Current Controller |
| | DQ-axis inductance, Ldq | Motor Parameters |

**`D and Q axis integral gain, Ki` — Derived**
`251.3274` (default) | `scalar`

Derived axis integral gain, in V/A*s.

#### Dependencies

This table summarizes the parameter dependencies.

| Parameter | Dependency | |
|---|---|---|
| | Parameter | Tab |
| D and Q axis integral gain, Ki | Bandwidth of the current regulator, EV_current | Current Controller |
| | Stator resistance, Rs<br><br>DQ-axis inductance, Ldq | Motor Parameters |

### Speed Controller

**`Bandwidth of the motion controller, EV_motion` — Bandwidth**
`[20, 4, 0.8]` (default) | `vector`

Motion controller bandwidth, in Hz. Set the first element of the vector to the desired cutoff frequency. Set the second and third elements of the vector to the higher-order cut off frequencies. You can set the value of the next element to `1/5` the value of the previous element. For example, if the desired cutoff frequency is `20` Hz, specify `[20 4 0.8]`.

#### Dependencies

The parameter is enabled when the **Control Type** parameter is set to `Speed Control`.

| Parameter | Used to Derive | |
|---|---|---|
| | Parameter | Tab |
| Bandwidth of the motion controller, EV_motion | Proportional gain, ba<br><br>Angular gain, Ksa<br><br>Rotational gain, Kisa | Speed Controller |

**`Bandwidth of the state filter, EV_sf` — Bandwidth**
`200` (default) | `scalar`

State filter bandwidth, in Hz.

#### Dependencies

The parameter is enabled when the **Control Type** parameter is set to `Speed Control`.

| Parameter | Used to Derive | |
|---|---|---|
| | Parameter | Tab |
| **Bandwidth of the state filter, EV_sf** | **Speed regulation time constant, Ksf** | **Speed Controller** |

### `Calculate Speed Regulator Gains` — Derive parameters

button

Click to derive parameters.

#### Dependencies

On the **Speed Controller** tab, when you select **Calculate Speed Regulator Gains**, the block calculates derived parameters. The table summarizes the derived parameters that depend on other block parameters.

| Derived Parameter on Speed Controller tab | | Depends On | |
|---|---|---|---|
| | | Parameter | Tab |
| **Proportional gain, ba** | $b_a = \frac{J_p - J_p p_1 p_2 p_3}{T_{sm}}$ | **Bandwidth of the motion controller, EV_motion**<br><br>**Bandwidth of the state filter, EV_sf** | **Speed Controller** |
| **Angular gain, Ksa** | $K_{sa} = \frac{J_p(p_1p_2 + p_2p_3 + p_3p_1) - 3J_p + 2b_aT_{sm}}{T_{sm}^2}$ | **Sample time for the torque control, Tst** | **Current Controller** |
| **Rotational gain, Kisa** | $K_{isa} = \frac{-J_p(p_1 + p_2 + p_3) + 3J_p - b_aT_{sm} - K_{sa}T_{sm}^2}{T_{sm}^3}$ | **Physical inertia, viscous damping, static friction, Mechanical** | **Motor Parameters** |
| **Speed regulation time constant, Ksf** | $K_{sf} = \frac{1 - \exp\left(-T_{sm}2\pi EV_{sf}\right)}{T_{sm}}$ | | |
| **Inertia compensation, Jcomp** | $J_{comp} = J_p$ | **Physical inertia, viscous damping, static friction, Mechanical** | **Motor Parameters** |
| **Viscous damping compensation, Fv** | $F_v$ | | |
| **Static friction, Fs** | $F_s$ | | |

The equations use these variables.

| | |
|---|---|
| $P$ | Motor pole pairs |
| $b_a$ | Speed regulator proportional gain |
| $K_{sa}$ | Speed regulator integral gain |
| $K_{isa}$ | Speed regulator double integral gain |
| $K_{sf}$ | Speed regulator time constant |
| $J_p$ | Motor inertia |
| $EV_{sf}$ | State filter bandwidth |
| $EV_{motion}$ | Motion controller bandwidth |

**`Proportional gain, ba` — Derived**
`3.7477` (default) | `scalar`

Derived proportional gain, in N·m/(rad/s).

**Dependencies**

This table summarizes the parameter dependencies.

| Parameter | Dependency | |
|---|---|---|
| | Parameter | Tab |
| **Proportional gain, ba** | **Physical inertia, viscous damping, static friction, Mechanical** | **Motor Parameters** |
| | **Bandwidth of the motion controller, EV_motion** | **Speed Controller** |

**`Angular gain, Ksa` — Derived**
`94.0877` (default) | `scalar`

Derived angular gain, in N·m/rad.

**Dependencies**

This table summarizes the parameter dependencies.

| Parameter | Dependency | |
|---|---|---|
| | Parameter | Tab |
| **Angular gain, Ksa** | **Physical inertia, viscous damping, static friction, Mechanical** | **Motor Parameters** |
| | **Bandwidth of the motion controller, EV_motion** | **Speed Controller** |

**`Rotational gain, Kisa` — Derived**
`381.7822` (default) | `scalar`

Derived rotational gain, in N·m/(rad*s).

**Dependencies**

This table summarizes the parameter dependencies.

| Parameter | Dependency | |
|---|---|---|
| | Parameter | Tab |
| Rotational gain, Kisa | Physical inertia, viscous damping, static friction, Mechanical | Motor Parameters |
| | Bandwidth of the motion controller, EV_motion | Speed Controller |

**`Speed regulation time constant, Ksf` — Derived**
`1217.9727` (default) | `scalar`

Derived speed regulation time constant, in 1/s.

**Dependencies**

This table summarizes the parameter dependencies.

| Parameter | Dependency | |
|---|---|---|
| | Parameter | Tab |
| Speed regulation time constant, Ksf | Sample time for the torque control, Tst | Current Controller |
| | Bandwidth of the state filter, EV_sf | Speed Controller |

**`Inertia compensation, Jcomp` — Derived**
`0.025` (default) | `scalar`

Derived inertia compensation, in kg·m^2.

**Dependencies**

This table summarizes the parameter dependencies.

| Parameter | Dependency | |
|---|---|---|
| | Parameter | Tab |
| Inertia compensation, Jcomp | Physical inertia, viscous damping, static friction, Mechanical | Motor Parameters |

**`Viscous damping compensation, Fv` — Derived**
`0` (default) | `scalar`

**Dependencies**

This table summarizes the parameter dependencies.

| Parameter | Dependency | |
|---|---|---|
| | Parameter | Tab |
| **Viscous damping compensation, Fv** | **Physical inertia, viscous damping, static friction, Mechanical** | **Motor Parameters** |

### `Static friction, Fs` — Derived

`0` (default) | `scalar`

Derived static friction, in N·m/(rad/s).

#### Dependencies

This table summarizes the parameter dependencies.

| Parameter | Dependency | |
|---|---|---|
| | Parameter | Tab |
| **Static friction, Fs** | **Physical inertia, viscous damping, static friction, Mechanical** | **Motor Parameters** |

#### Electrical Losses

### `Parameterize losses by` — Select type

`Single efficiency measurement` (default) | `Tabulated loss data` | `Tabulated efficiency data`

| Setting | Block Implementation |
|---|---|
| `Single efficiency measurement` | Electrical loss calculated using a constant value for inverter efficiency. |
| `Tabulated loss data` | Electrical loss calculated as a function of motor speeds and load torques. |
| `Tabulated efficiency data` | Electrical loss calculated using inverter efficiency that is a function of motor speeds and load torques.<br>• Converts the efficiency values you provide into losses and uses the tabulated losses for simulation.<br>• Ignores efficiency values you provide for zero speed or zero torque. Losses are assumed zero when either torque or speed is zero.<br>• Uses linear interpolation to determine losses. Provide tabulated data for low speeds and low torques, as required, to get the desired level of accuracy for lower power conditions.<br>• Does not extrapolate loss values for speed and torque magnitudes that exceed the range of the table. |

For best practice, use `Tabulated loss data` instead of `Tabulated efficiency data`:

- Efficiency becomes ill defined for zero speed or zero torque.

- You can account for fixed losses that are still present for zero speed or torque.

**`Overall inverter efficiency, eff` — Constant**
`98` (default) | `scalar`

Overall inverter efficiency, *Eff*, in %.

**Dependencies**

To enable this parameter, for **Parameterize losses by**, select `Tabulated loss data`.

**`Vector of speeds (w) for tabulated loss, w_loss_bp` — Breakpoints**
`[0 200 400 600 800 1000]` (default) | 1-by-`M` vector

Speed breakpoints for lookup table when calculating losses, in rad/s.

**Dependencies**

To enable this parameter, for **Parameterize losses by**, select `Tabulated loss data`.

**`Vector of torques (T) for tabulated loss, T_loss_bp` — Breakpoints**
`[0 25 50 75 100]` (default) | 1-by-`N` vector

Torque breakpoints for lookup table when calculating losses, in N·m.

**Dependencies**

To enable this parameter, for **Parameterize losses by**, select `Tabulated loss data`.

**`Corresponding losses, losses_table` — Table**
`[100 100 100 100 100;100 150 200 250 300;100 200 300 400 500;100 250 400 550 700;100 300 500 700 900;100 350 600 850 1100]` (default) | `M`-by-`N` array

Array of values for electrical losses as a function of `M` speeds and `N` torques, in W. Each value specifies the losses for a specific combination of speed and torque. The matrix size must match the dimensions defined by the speed and torque vectors.

**Dependencies**

To enable this parameter, for **Parameterize losses by**, select `Tabulated loss data`.

**`Vector of speeds (w) for tabulated efficiency, w_eff_bp` — Breakpoints**
`[200 400 600 800 1000]` (default) | 1-by-`M` vector

Speed breakpoints for lookup table when calculating efficiency, in rad/s.

**Dependencies**

To enable this parameter, for **Parameterize losses by**, select `Tabulated efficiency data`.

**`Vector of torques (T) for tabulated efficiency, T_eff_bp` — Breakpoints**
`[25 50 75 100]` (default) | 1-by-`N` vector

Torque breakpoints for lookup table when calculating efficiency, in N·m.

**Dependencies**

To enable this parameter, for **Parameterize losses by**, select `Tabulated efficiency data`.

**Corresponding efficiency, efficiency_table — Table**
`[96.2 98.1 98.7 99;98.1 99 99.4 99.5;98.7 99.4 99.6 99.7;99 99.5 99.7 99.8;99.2 99.6 99.7 99.8]` (default) | `M`-by-`N` array

Array of efficiency as a function of `M` speeds and `N` torque, in %. Each value specifies the efficiency for a specific combination of speed and torque. The matrix size must match the dimensions defined by the speed and torque vectors.

The block ignores efficiency values for zero speed or zero torque. Losses are zero when either torque or speed is zero. The block uses linear interpolation.

To get the desired level of accuracy for lower power conditions, you can provide tabulated data for low speeds and low torques.

**Dependencies**

To enable this parameter, for **Parameterize losses by**, select `Tabulated efficiency data`.

## References

[1] Lorenz, Robert D., Thomas Lipo, and Donald W. Novotny. “Motion control with induction motors.” *Proceedings of the IEEE*, Vol. 82, Issue 8, August 1994, pp. 1215–1240.

[2] Shigeo Morimoto, Masayuka Sanada, Yoji Takeda. “Wide-speed operation of interior permanent magnet synchronous motors with high-performance current regulator.” *IEEE Transactions on Industry Applications*, Vol. 30, Issue 4, July/August 1994, pp. 920–926.

[3] Muyang Li. “Flux-Weakening Control for Permanent-Magnet Synchronous Motors Based on Z-Source Inverters.” Master’s Thesis, Marquette University, e-Publications@Marquette, Fall 2014.

[4] Briz, Fernando, Michael W. Degner, and Robert D. Lorenz. "Analysis and design of current regulators using complex vectors." *IEEE Transactions on Industry Applications*, Vol. 36, Issue 3, May/June 2000, pp. 817–825.

[5] Briz, Fernando, et al. "Current and flux regulation in field-weakening operation [of induction motors]."*IEEE Transactions on Industry Applications*, Vol. 37, Issue 1, Jan/Feb 2001, pp. 42–50.

## Extended Capabilities

**C/C++ Code Generation**
Generate C and C++ code using Simulink® Coder™.

## See Also

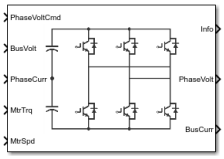
Flux-Based PM Controller | IM Controller | Interior PM Controller | Surface Mount PMSM

**Introduced in R2017a**

# Three-Phase Voltage Source Inverter

Three-phase voltage source inverter

**Library:** Powertrain Blockset / Propulsion / Electric Motors and Inverters



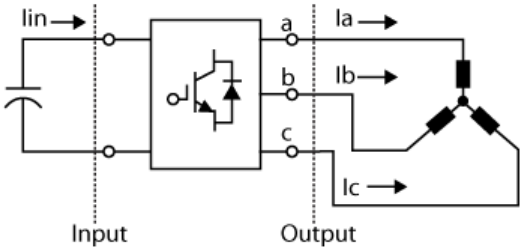

## Description

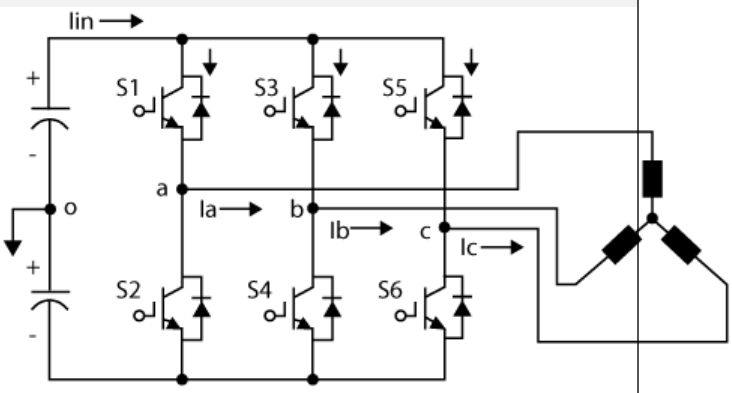
The Three-Phase Voltage Source Inverter block implements a three-phase voltage source inverter that generates neutral voltage commands for a balanced three-phase load. Configure the voltage switching function for continuous vector modulation or inverter switch input signals. You can incorporate the block into a closed-loop model to simulate a power inverter. The block controls the ideal switch states.

To enable power loss calculations suitable for code generation targets that limit memory, select **Enable memory optimized 2D LUT**. Click **Calibrate Maps** to virtually calibrate an inverter power loss lookup table as a function of motor torque and motor speed.

If you select **Input inverter temperature**, click **Calibrate Maps** to virtually calibrate the power loss table as a function of motor torque, motor speed, and inverter temperature. You cannot enable memory optimization for the 3D power loss lookup table.

Use the **Switching voltage function** parameter to set the switching voltage function.

| Setting | Implementation | Illustration |
|---|---|---|
| **Commanded phase voltage** | Phase a, b, c line-to-neutral voltage command input. Suitable for continuous sinusoidal or space vector modulation input signals. |   |

| Setting | Implementation | Illustration |
|---|---|---|
| **Switch inputs** (default) | Inverter switch input command. Suitable for hardware-in-the-loop (HIL) simulation.<br><br>The inverter switches S1, S3, and S5 using complimented control for S2, S4, and S6. |  |

### Virtual Calibration

If you have Model-Based Calibration Toolbox, click **Calibrate Maps** to virtually calibrate the lookup tables using measured data. The dialog box steps through these tasks.

| Task | Description |
|---|---|
| Import Loss Data | Import this loss data from a file. For example, open `<matlabroot>/toolbox/autoblks/autoblksshared/mbctemplates/MappedInverterDataset.xlsx`.<br><br>For more information, see “Using Data” (Model-Based Calibration Toolbox).<br><br>**Input inverter temperature Setting** / **Required Data**<br>`off`: • Motor speed, rad/s • Motor torque, N·m • Power loss, W<br>`on`: • Motor speed, rad/s • Motor torque, N·m • Motor temperature, K • Power loss, W<br><br>Collect inverter data at steady-state operating conditions. Data should cover the inverter speed, torque, and temperature operating range.<br><br>To filter or edit the data, select **Edit in Application**. The Model-Based Calibration Toolbox Data Editor opens. |

<table>
<tr><th>Task</th><th>Description</th></tr>
<tr><td>Generate Response Models</td><td>Model-Based Calibration Toolbox uses test plans to fit data to Gaussian process models (GPMs).<br><br>To assess or adjust the response model fit, select Edit in Application. The Model-Based Calibration Toolbox Model Browser opens. For more information, see “Model Assessment” (Model-Based Calibration Toolbox).</td></tr>
<tr><td>Generate Calibration</td><td>Model-Based Calibration Toolbox calibrates the response models and generates calibrated tables.<br><br>To assess or adjust the calibration, select Edit in Application. The Model-Based Calibration Toolbox CAGE Browser opens. For more information, see “Calibration Lookup Tables” (Model-Based Calibration Toolbox).</td></tr>
<tr><td>Update block parameters</td><td>Update these parameters with the calibration.<br><table><tr><th>Input inverter temperature Setting</th><th>Parameters</th></tr><tr><td>off</td><td>• Vector of speeds (w) for tabulated losses, w_eff_bp<br>• Vector of torques (T) for tabulated losses, T_eff_bp<br>• Corresponding power loss, ploss_table</td></tr><tr><td>on</td><td>• Vector of speeds (w) for tabulated losses, w_eff_bp<br>• Vector of torques (T) for tabulated losses, T_eff_bp<br>• Vector of temperatures for tabulated losses, Temp_eff_bp<br>• Corresponding power loss, ploss_table_3d</td></tr></table></td></tr>
</table>

### Switching Function

For the switch voltage, the block implementation depends on the **Switching voltage function** setting.

| Setting | Calculation | Equations |
|---|---|---|
| `Commanded phase voltage` | Continuous line-to-neutral voltage commands set to phase a, b, c line-to-neutral voltage command input | $v_{an} = v_{a_cmd}$<br>$v_{bn} = v_{b_cmd}$<br>$v_{cn} = v_{c_cmd}$ |
| | Line-to-line voltage | $v_{ab} = v_{an} - v_{bn}$<br>$v_{bc} = v_{bn} - v_{cn}$<br>$v_{ca} = v_{cn} - v_{an}$ |

| Setting | Calculation | Equations |
|---|---|---|
| `Switch inputs` | Switching function | $SF_a = \begin{cases} 1 & \text{S1 on and S2 off} \\ -1 & \text{S1 off and S2 on} \end{cases}$<br>$SF_b = \begin{cases} 1 & \text{S3 on and S4 off} \\ -1 & \text{S3 off and S4 on} \end{cases}$<br>$SF_c = \begin{cases} 1 & \text{S5 on and S6 off} \\ -1 & \text{S5 off and S6 on} \end{cases}$ |
| | Line-to-center point voltage | $v_{ao} = \frac{v_{bus}}{2} SF_a$<br>$v_{bo} = \frac{v_{bus}}{2} SF_b$<br>$v_{co} = \frac{v_{bus}}{2} SF_c$ |
| | Line-to-neutral voltage | $v_{an} = v_{ao} - v_{no}$<br>$v_{bn} = v_{bo} - v_{no}$<br>$v_{cn} = v_{co} - v_{no}$<br>$v_{an} + v_{bn} + v_{cn} = 0$<br>$v_{no} = \frac{1}{3}(v_{ao} + v_{bo} + v_{co})$<br>$v_{an} = v_{ao} - \frac{1}{3}(v_{ao} + v_{bo} + v_{co})$<br>$v_{bn} = v_{bo} - \frac{1}{3}(v_{ao} + v_{bo} + v_{co})$<br>$v_{cn} = v_{co} - \frac{1}{3}(v_{ao} + v_{bo} + v_{co})$ |
| | Line-to-line voltage | $v_{ab} = v_{an} - v_{bn}$<br>$v_{bc} = v_{bn} - v_{cn}$<br>$v_{ca} = v_{cn} - v_{an}$ |

The equations use these variables.

| | |
|---|---|
| $SF_a$, $SF_b$, $SF_c$ | Phase a, b, c line switching functions, respectively |
| $v_{bus}$ | Power source bus voltage |
| $V_{ao}$, $V_{bo}$, $V_{co}$ | Phase a, b, c line-to-center voltage, respectively |
| $V_{an}$, $V_{bn}$, $V_{cn}$ | Phase a, b, c line-to-neutral voltage, respectively |
| $V_{ab}$, $V_{bc}$, $V_{ca}$ | Phase ab, bc, ca line-to-neutral voltage, respectively |
| $V_{a_cmd}$, $V_{b_cmd}$, $V_{c_cmd}$ | Phase a, b, c line-to-neutral voltage commands, respectively |

### Current and Power Loss

For the line-to-center, line-to-neutral, and line-to-line voltage, the block implements these equations.

| Calculation | Equations |
|---|---|
| Motor and bus power | $P_{mtr} = v_{an}i_a + v_{bn}i_b + v_{cn}i_c$<br>$P_{bus} = v_{bus}i_{bus}$ |
| Inverter power loss and bus current | $P_{in} = P_{bus} = v_{bus}i_{bus}$<br>$P_{out} = P_{mtr} = v_{an}i_a + v_{bn}i_b + v_{cn}i_c + P_{LossInv}$<br>$i_{bus} = \frac{v_{an}i_a + v_{bn}i_b + v_{cn}i_c + P_{LossInv}}{v_{bus}}$ |

The equations use these variables.

| | |
|---|---|
| $P_{mtr}$ | Power delivered to the motor |
| $P_{bus}$ | Power from input bus |
| $P_{loss}$ | Power loss |
| $i_{bus}$ | Power source bus current |
| $i_a$, $i_b$, $i_c$ | Phase a, b, c line current, respectively |
| $V_{an}$, $V_{bn}$, $V_{cn}$ | Phase a, b, c line-to-neutral voltage, respectively |
| $v_{bus}$ | Power source bus voltage |

### Power Accounting

For the power accounting, the block implements these equations.

| Bus Signal | | | Description | Variable | Equation |
|---|---|---|---|---|---|
| `PwrInfo` | `PwrTrnsfrd` — Power transferred between blocks<br>• Positive signals indicate flow into block<br>• Negative signals indicate flow out of block | `PwrMtr` | Power delivered to the motor | $P_{TrnsfrdMtr}$ | $P_{TrnsfrdMtr} = -(v_{an}i_a + v_{bn}i_b + v_{cn}i_c)$ |
| | | `PwrBus` | Power from input bus | $P_{TrnsfrdBus}$ | $P_{TrnsfrdBus} = P_{bus}$ |

| Bus Signal | | | Description | Variable | Equation |
|---|---|---|---|---|---|
| | `PwrNotTrnsfrd` — Power crossing the block boundary, but not transferred<br>• Positive signals indicate an input<br>• Negative signals indicate a loss | `PwrLoss` | Power loss<br>Negative value indicates power loss | $P_{NotTrnsfrd}$ | $P_{NotTrnsfrd} = -(P_{TrnsfrdBus} + P_{TrnsfrdMtr})$ |
| | `PwrStored` — Stored energy rate of change<br>• Positive signals indicate an increase<br>• Negative signals indicate a decrease | | *Not used* | | |

### Lookup Table Memory Optimization

The inverter power loss table parameter **Corresponding power loss, ploss_table** data is a function of motor torque and motor speed at different battery voltages. Positive current indicates battery discharge. Negative current indicates battery charge.

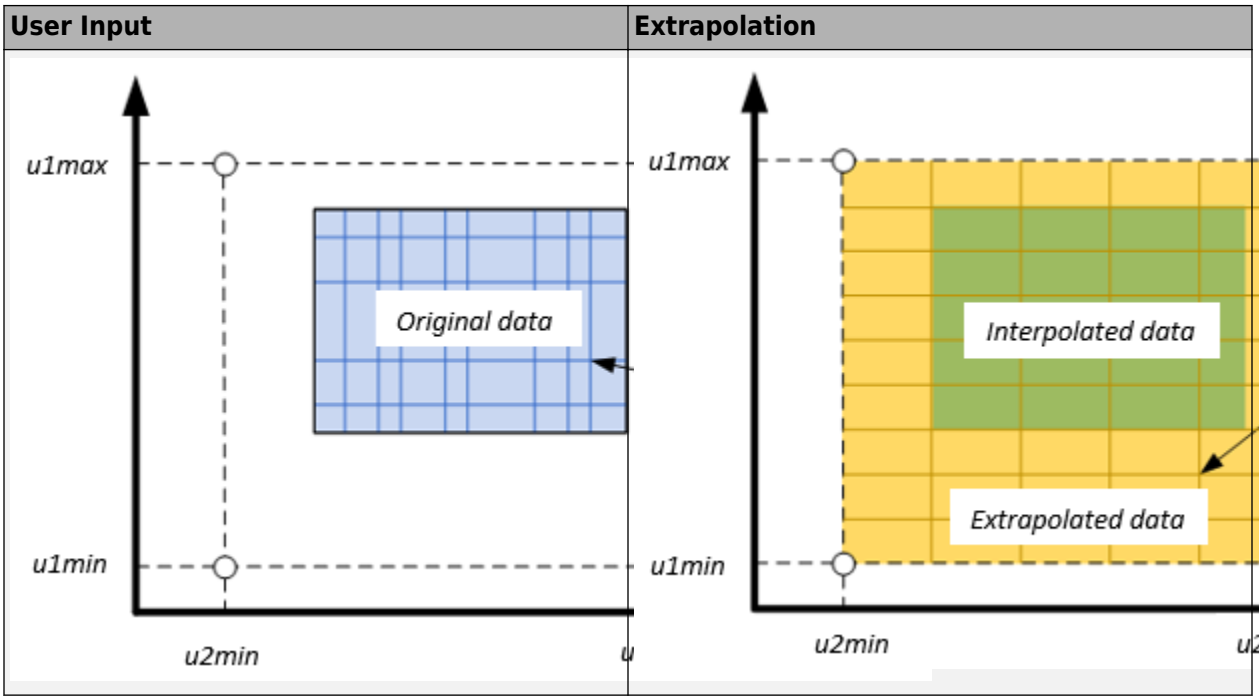
To enable power loss calculations suitable for code generation targets that limit memory, select **Enable memory optimized 2D LUT**. The block uses linear interpolation to optimize the inverter power loss lookup table values for code generation. This table summarizes the optimization implementation.

| Use Case | Implementation |
|---|---|
| Motor speed and torque input align with the lookup table breakpoint values. | Memory-optimized power loss is power loss lookup table value at intersection of motor speed and torque. |
| Motor speed and torque input do not align with the lookup table breakpoint values, but are within range. | Memory-optimized power loss is linear interpolation between corresponding motor speed and torque. |
| Motor speed and torque input do not align with the lookup table breakpoint values, and are out of range. | Cannot compute a memory-optimized power loss. Block uses extrapolated data. |

#### Extrapolation

The lookup tables optimized for code generation do not support extrapolation for data that is out of range. However, you can include pre-calculated extrapolation values in the power loss lookup table by selecting **Specify Extrapolation**.

The block uses the endpoint parameters to resize the table data.




## Ports

### Input

#### `PhaseVoltCmd` — Phase a, b, c line-to-neutral voltage command

1-by-3 array

Phase a, b, c line-to-neutral voltage command, $V_{a_cmd}$, $V_{b_cmd}$, and $V_{c_cmd}$, in V.

**Dependencies**

To create this port, set **Switching voltage function** to `Commanded phase voltage`.

#### `SwitchCmd` — Switch commands

1-by-3 array

Switch commands, $S_a$, $S_b$, and $S_c$, dimensionless.

**Dependencies**

To create this port, set **Switching voltage function** to `Switch inputs`.

#### `BusVolt` — Power source bus voltage

bus

Power source bus voltage, $V_{bus}$, in V.

#### `PhaseCurr` — Phase a, b, c current

1-by-3 array

Phase a, b, c current, $i_a$, $i_b$, and $i_c$, in A.

**`MtrTrq` — Motor torque**
`scalar`

Motor torque, $T_{mtr}$, in N·m.

**`MtrSpd` — Motor speed**
`scalar`

Angular speed of the motor, $\omega_{mtr}$, in rad/s.

**`InvrtrTemp` — Inverter operating temperature**
`scalar`

Inverter operating temperature, $Temp_{Invrtr}$, in K.

#### Dependencies

To create this port, select **Input inverter temperature**.

#### Output

**`Info` — Bus signal**
bus

The bus signal contains these block calculations.

| Signal | | | Description | Variable | Units |
|---|---|---|---|---|---|
| `BusCurr` | | | Power source bus current | $i_{bus}$ | A |
| `PwrLossInv` | | | Inverter power loss | $\varepsilon_{inv}$ | dimensionless |
| `PwrInfo` | `PwrTrnsfrd` | `PwrMtr` | Power delivered to the motor | $P_{TrnsfrdMtr}$ | W |
| | | `PwrBus` | Power from input bus | $P_{TrnsfrdBus}$ | W |
| | `PwrNotTrnsfrd` | `PwrLoss` | Power loss | $P_{NotTrnsfrd}$ | W |
| | `PwrStored` | | *Not used* | | |

**`PhaseVolt` — Phase a, b, c line-to-neutral voltage**
1-by-3 array

Phase a, b, c line-to-neutral voltage, $V_{an}$, $V_{bn}$, and $V_{cn}$, in V.

**`BusCurr` — Power source bus current**
`scalar`

Power source bus current, $i_{bus}$, in A.

## Parameters

#### Block Options

**`Input inverter temperature` — Create input port**
`off` (default) | `on`

Select this parameter to create the `InvrtrTemp` input port.

The block enables you to specify inverter power loss lookup tables that are functions of motor torque, $T_{mtr}$, and motor speed, $\omega_{mtr}$. If you select **Input inverter temperature**, the tables are also a function of the inverter temperature, $Temp_{Invrtr}$.

| Input Inverter Temperature Parameter Setting | Enables Efficiency Table | Function Of |
|---|---|---|
| `off` | **Corresponding power loss, ploss_table** | $f(T_{mtr}, \omega_{mtr})$ |
| `on` | **Corresponding power loss, ploss_table_3d** | $f(T_{mtr}, \omega_{mtr}, Temp_{Invrtr})$ |

#### Dependencies

If you select **Input inverter temperature** to specify a 3D power loss lookup table as a function of motor torque, motor speed, and inverter temperature, you cannot select **Enable memory optimized 2D LUT** to enable a memory optimization.

### `Enable memory optimized 2D LUT` — Selection

`off` (default) | `on`

Enable generation of memory-optimized lookup tables, suitable code generation targets that limit memory.

#### Dependencies

If you select **Enable memory optimized 2D LUT**, you cannot select **Input inverter temperature**.

### `Calibrate Maps` — Calibrate tables with measured data

`selection`

If you have Model-Based Calibration Toolbox, click **Calibrate Maps** to virtually calibrate the lookup tables using measured data. The dialog box steps through these tasks.

<table>
<tr><th>Task</th><th>Description</th></tr>
<tr><td>Import Loss Data</td><td>Import this loss data from a file. For example, open <code><matlabroot>/toolbox/autoblks/autoblksshared/mbctemplates/MappedInverterDataset.xlsx</code>.<br><br>For more information, see “Using Data” (Model-Based Calibration Toolbox).<br><br>
<table>
<tr><th>Input inverter temperature Setting</th><th>Required Data</th></tr>
<tr><td><code>off</code></td><td>• Motor speed, rad/s<br>• Motor torque, N·m<br>• Power loss, W</td></tr>
<tr><td><code>on</code></td><td>• Motor speed, rad/s<br>• Motor torque, N·m<br>• Motor temperature, K<br>• Power loss, W</td></tr>
</table>
Collect inverter data at steady-state operating conditions. Data should cover the inverter speed, torque, and temperature operating range.<br><br>To filter or edit the data, select Edit in Application. The Model-Based Calibration Toolbox Data Editor opens.</td></tr>
<tr><td>Generate Response Models</td><td>Model-Based Calibration Toolbox uses test plans to fit data to Gaussian process models (GPMs).<br><br>To assess or adjust the response model fit, select Edit in Application. The Model-Based Calibration Toolbox Model Browser opens. For more information, see “Model Assessment” (Model-Based Calibration Toolbox).</td></tr>
<tr><td>Generate Calibration</td><td>Model-Based Calibration Toolbox calibrates the response models and generates calibrated tables.<br><br>To assess or adjust the calibration, select Edit in Application. The Model-Based Calibration Toolbox CAGE Browser opens. For more information, see “Calibration Lookup Tables” (Model-Based Calibration Toolbox).</td></tr>
</table>

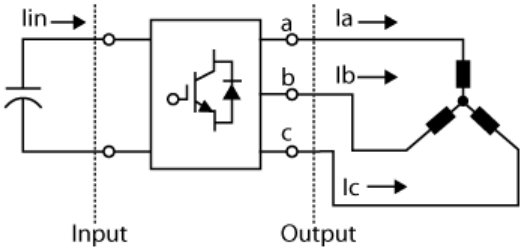
<table>
<tr><th>Task</th><th>Description</th></tr>
<tr><td>Update block parameters</td><td>Update these parameters with the calibration.
<table>
<tr><th>Input inverter temperature Setting</th><th>Parameters</th></tr>
<tr><td><code>off</code></td><td>• Vector of speeds (w) for tabulated losses, w_eff_bp<br>• Vector of torques (T) for tabulated losses, T_eff_bp<br>• Corresponding power loss, ploss_table</td></tr>
<tr><td><code>on</code></td><td>• Vector of speeds (w) for tabulated losses, w_eff_bp<br>• Vector of torques (T) for tabulated losses, T_eff_bp<br>• Vector of temperatures for tabulated losses, Temp_eff_bp<br>• Corresponding power loss, ploss_table_3d</td></tr>
</table></td></tr>
</table>

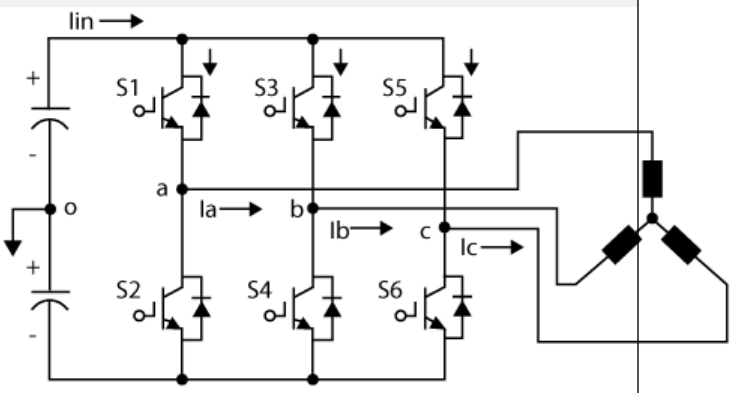
**Electrical Model**

### `Switching voltage function` — Selection

`Commanded phase voltage` (default) | `Switch inputs`

Use the **Switching voltage function** parameter to set the switching voltage function.

| Setting | Implementation | Illustration |
|---|---|---|
| **Commanded phase voltage** | Phase a, b, c line-to-neutral voltage command input. Suitable for continuous sinusoidal or space vector modulation input signals. |  |

| Setting | Implementation | Illustration |
|---|---|---|
| **Switch inputs** (default) | Inverter switch input command. Suitable for hardware-in-the-loop (HIL) simulation.<br><br>The inverter switches S1, S3, and S5 using complimented control for S2, S4, and S6. |  |

**`Vector of speeds (w) for tabulated losses, w_eff_bp` — Speed breakpoints**
`[-1000 -500 0 500 1000]` (default) | 1-by-M vector

Vector of motor speed, $\omega_{mtr}$, breakpoints for power loss, in rad/s. If you set **Enable memory optimized 2D LUT**, the block converts the data to single precision.

**`Resample storage size for w_eff_bp, n1` — Speed bit size**
128 (default) | 2 | 4 | 8 | 16 | 32 | 64 | 256

Speed breakpoint storage size, *n1*, dimensionless. The block resamples the **Corresponding power loss, ploss_table** data based on the storage size.

**Dependencies**

To create this parameter, select **Enable memory optimized 2D LUT**.

**`Vector of torques (T) for tabulated losses, T_eff_bp` — Torque breakpoints**
`[-200 -100 0 100 200]` (default) | 1-by-N vector

Vector of motor torque, $T_{mtr}$, breakpoints for power loss, in N·m. If you set **Enable memory optimized 2D LUT**, the block converts the data to single precision.

**`Resample storage size for T_eff_bp, n2` — Torque bit size**
128 (default) | 2 | 4 | 8 | 16 | 32 | 64 | 256

Torque breakpoint storage size, *n2*, dimensionless. The block resamples the **Corresponding power loss, ploss_table** data based on the storage size.

**Dependencies**

To create this parameter, select **Enable memory optimized 2D LUT**.

**`Vector of temperatures for tabulated losses, Temp_eff_bp` — Temperature breakpoints**
`[213.15 293.15 373.15]` (default) | 1-by-L vector

Vector of inverter temperature, $Temp_{Invrtr}$, breakpoints for power loss, in K.

#### Dependencies

To create this parameter, select **Input inverter temperature**.

### `Corresponding power loss, ploss_table` — 2D lookup table

`[1 0.999 0.989 0.997 0.996;0.995 0.994 0.993 0.992 0.991;0.990 0.989 0.988 0.987 0.986;0.985 0.984 0.983 0.982 0.981;0.980 0.979 0.978 0.977 0.976]` (default) | `M`-by-`N` array

Array of values for power loss as a function of `M` motor speeds, $\omega_{mtr}$, and `N` motor torques, $T_{mtr}$, in W. Each value specifies the power loss for a specific combination of motor speed and motor torque. The array size must match the dimensions defined by the speed and torque vectors.

If you have Model-Based Calibration Toolbox, click **Calibrate Maps** to virtually calibrate the lookup table using measured data.

If you set **Enable memory optimized 2D LUT**, the block converts the data to single precision.

#### Dependencies

To create this parameter, clear **Input inverter temperature**.

### `Corresponding power loss, ploss_table_3d` — 3D lookup table

`M`-by-`N`-by-`L` array

Array of values for power loss as a function of `M` motor speeds, $\omega_{mtr}$, `N` motor torques, $T_{mtr}$, and `L` motor temperatures, $Temp_{Invrtr}$, in W. Each value specifies the power loss for a specific combination of motor speed, motor torque, and temperature. The array size must match the dimensions defined by the speed, torque, and temperature vectors.

If you have Model-Based Calibration Toolbox, click **Calibrate Maps** to virtually calibrate the lookup table using measured data.

#### Dependencies

To create this parameter, select **Input inverter temperature**.

#### Specify Extraction

### `w_eff_bp max endpoint, u1max` — Speed breakpoint

`1000` (default) | `scalar`

Speed breakpoint maximum extrapolation endpoint, *u1max*, in rad/s.

#### Dependencies

To create this parameter, select **Enable memory optimized 2D LUT** and **Specify Extrapolation**.

### `w_eff_bp min endpoint, u1min` — Speed breakpoint

`-1000` (default) | `scalar`

Speed breakpoint minimum extrapolation endpoint, *u1min*, in rad/s.

#### Dependencies

To create this parameter, select **Enable memory optimized 2D LUT** and **Specify Extrapolation**.

**`T_eff_bp max endpoint, u2max` — Torque breakpoint**
`200` (default) | `scalar`

Torque breakpoint maximum extrapolation endpoint, *u2max*, in rad/s.

**Dependencies**

To create this parameter, select **Enable memory optimized 2D LUT** and **Specify Extrapolation**.

**`T_eff_bp min endpoint, u2min` — Torque breakpoint**
`-200` (default) | `scalar`

Torque breakpoint minimum extrapolation endpoint, *u2min*, in rad/s.

**Dependencies**

To create this parameter, select **Enable memory optimized 2D LUT** and **Specify Extrapolation**.

## References

[1] Lee, Byoung-Kuk and Mehrdad Ehsami. "A simplified functional simulation model for three-phase voltage-source inverter using switching function concept." *IEEE Transactions on Industrial Electronics*, Vol. 48, No. 2, pp. 309-321, April 2001.

[2] Ziogas, Phoivas D., Eduardo P. Wiechmann, and Victor R. Stefanovic. "A Computer-Aided Analysis and Design Approach for Static Voltage Source Inverters." *IEEE Transactions on Industrial Electronics. Transactions on Industry Applications*, Vol. IA-21, No. 5, September/October 1985.

## Extended Capabilities

**C/C++ Code Generation**
Generate C and C++ code using Simulink® Coder™.

## See Also

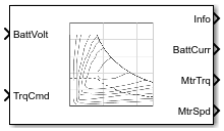
Flux-Based PM Controller | Induction Motor | Interior PMSM | Surface Mount PMSM

**Introduced in R2019a**

# Mapped Motor

Mapped motor and drive electronics operating in torque-control mode

**Library:** Powertrain Blockset / Propulsion / Electric Motors and Inverters
Vehicle Dynamics Blockset / Powertrain / Propulsion




## Description

The Mapped Motor block implements a mapped motor and drive electronics operating in torque-control mode. The output torque tracks the torque reference demand and includes a motor-response and drive-response time constant. Use the block for fast system-level simulations when you do not know detailed motor parameters, for example, for motor power and torque tradeoff studies. The block assumes that the speed fluctuations due to mechanical load do not affect the motor torque tracking.

You can specify:

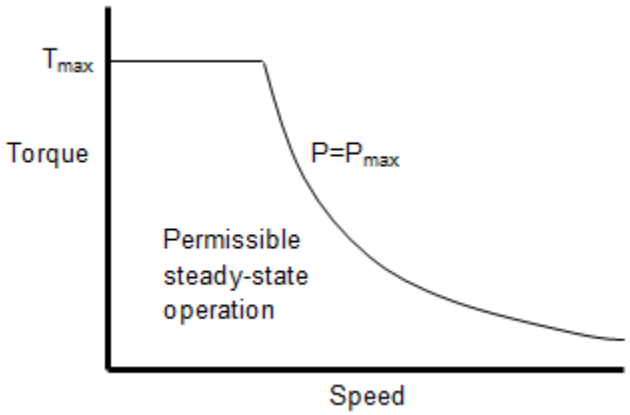
- Port configuration — Input torque or speed.
- Electrical torque range — Torque speed envelope or maximum motor power and torque.
- Electrical loss — Single operating point, measured efficiency, or measured loss. If you have Model-Based Calibration Toolbox, you can virtually calibrate the measured loss tables.

### Electrical Torque

To specify the range of torque and speed that the block allows, on the **Electrical Torque** tab, for **Parametrized by**, select one of these options.

| Setting | Block Implementation |
|---|---|
| `Tabulated torque-speed envelope` | Range specified as a set of speed data points and corresponding maximum torque values. |
| `Maximum torque and power` | Range specified with maximum torque and maximum power. |

For either method, the block implements an envelope similar to this.




### Electrical Losses

To specify the electrical losses, on the **Electrical Losses** tab, for **Parameterize losses by**, select one of these options.

| Setting | Block Implementation |
|---|---|
| `Single efficiency measurement` | Sum of these terms, measured at a single measurement point:<br>• Fixed losses independent of torque and speed, $P_0$. Use $P_0$ to account for fixed converter losses.<br>• A torque-dependent electrical loss $k\tau^2$, where $k$ is a constant and $\tau$ is the torque. Represents ohmic losses in the copper windings.<br>• A speed-dependent electrical loss $k_w\omega^2$, where $k_w$ is a constant and $\omega$ is the speed. Represents iron losses due to eddy currents. |
| `Tabulated loss data` | Loss lookup table that is a function of motor speeds and load torques.<br><br>If you have Model-Based Calibration Toolbox, click **Calibrate Maps** to virtually calibrate the 2D lookup tables using measured data. |
| `Tabulated loss data with temperature` | Loss lookup table that is a function of motor speeds, load torques, and operating temperature.<br><br>If you have Model-Based Calibration Toolbox, click **Calibrate Maps** to virtually calibrate the 3D lookup tables using measured data. |

| Setting | Block Implementation |
|---|---|
| `Tabulated efficiency data` | 2D efficiency lookup table that is a function of motor speeds and load torques:<br>• Converts the efficiency values you provide into losses and uses the tabulated losses for simulation.<br>• Ignores efficiency values you provide for zero speed or zero torque. Losses are assumed zero when either torque or speed is zero.<br>• Uses linear interpolation to determine losses. Provide tabulated data for low speeds and low torques, as required, to get the desired level of accuracy for lower power conditions.<br>• Does not extrapolate loss values for speed and torque magnitudes that exceed the range of the table. |
| `Tabulated efficiency data with temperature` | 3D efficiency lookup table that is a function of motor speeds, load torques, and operating temperature:<br>• Converts the efficiency values you provide into losses and uses the tabulated losses for simulation.<br>• Ignores efficiency values you provide for zero speed or zero torque. Losses are assumed zero when either torque or speed is zero.<br>• Uses linear interpolation to determine losses. Provide tabulated data for low speeds and low torques, as required, to get the desired level of accuracy for lower power conditions.<br>• Does not extrapolate loss values for speed, torque, or temperature magnitudes that exceed the range of the table. |

For best practice, use `Tabulated loss data` instead of `Tabulated efficiency data`:

- Efficiency becomes ill defined for zero speed or zero torque.
- You can account for fixed losses that are still present for zero speed or torque.

**Note** Due to system losses, the motor can draw a current when the motor torque is zero.

### Virtual Calibration

If you have Model-Based Calibration Toolbox, you can virtually calibrate the measured loss lookup tables.

1. On the **Electrical Losses** tab, set **Parameterize losses by** to either:
   - `Tabulated loss data`
   - `Tabulated loss data with temperature`
2. Click **Calibrate Maps**.

The dialog box steps through these tasks.

<table>
<tr><th>Task</th><th>Description</th></tr>
<tr><td>Import Loss Data</td><td>Import this loss data from a file. For example, open <code><matlabroot>/toolbox/autoblks/autoblksshared/mbctemplates/MappedMotorDataset.xlsx</code>.<br><br>For more information, see “Using Data” (Model-Based Calibration Toolbox).<br><br>
<table>
<tr><th>Parameterize losses by</th><th>Required Data</th></tr>
<tr><td><code>Tabulated loss data</code></td><td>• Motor speed, rad/s<br>• Motor torque, N·m<br>• Power loss, W</td></tr>
<tr><td><code>Tabulated loss data with temperature</code></td><td>• Motor speed, rad/s<br>• Motor torque, N·m<br>• Motor temperature, K<br>• Power loss, W</td></tr>
</table>
<br>Collect motor data at steady-state operating conditions. Data should cover the motor speed, torque, and temperature operating range.<br><br>To filter or edit the data, select Edit in Application. The Model-Based Calibration Toolbox Data Editor opens.</td></tr>
<tr><td>Generate Response Models</td><td>Model-Based Calibration Toolbox uses test plans to fit data to Gaussian process models (GPMs).<br><br>To assess or adjust the response model fit, select Edit in Application. The Model-Based Calibration Toolbox Model Browser opens. For more information, see “Model Assessment” (Model-Based Calibration Toolbox).</td></tr>
<tr><td>Generate Calibration</td><td>Model-Based Calibration Toolbox calibrates the response models and generates calibrated tables.<br><br>To assess or adjust the calibration, select Edit in Application. The Model-Based Calibration Toolbox CAGE Browser opens. For more information, see “Calibration Lookup Tables” (Model-Based Calibration Toolbox).</td></tr>
</table>

<table>
<tr><th>Task</th><th>Description</th></tr>
<tr><td>Update block parameters</td><td>Update these parameters with the calibration.
<table>
<tr><th>Parameterize losses by</th><th>Parameters</th></tr>
<tr><td><code>Tabulated loss data</code></td><td>• Vector of speeds(w) for tabulated losses, w_eff_bp<br>• Vector of torques (T) for tabulated losses, T_eff_bp<br>• Corresponding losses, losses_table</td></tr>
<tr><td><code>Tabulated loss data with temperature</code></td><td>• Vector of speeds(w) for tabulated losses, w_eff_bp<br>• Vector of torques (T) for tabulated losses, T_eff_bp<br>• Vector of temperatures for tabulated losses, Temp_eff_bp<br>• Corresponding losses, losses_table_3d</td></tr>
</table></td></tr>
</table>

### Battery Current

The block calculates the battery current using the mechanical power, power loss, and battery voltage. Positive current indicates battery discharge. Negative current indicates battery charge.

$$BattAmp = \frac{MechPwr + PwrLoss}{BattVolt}$$

The equation uses these variables.

| | |
|---|---|
| *BattVolt* | Battery voltage |
| *MechPwr* | Mechanical power |
| *PwrLoss* | Power loss |
| *BattCurr* | Battery current |

### Power Accounting

For the power accounting, the block implements these equations.

| Bus Signal | | | Description | Variable | Equations |
|---|---|---|---|---|---|
| `PwrInfo` | `PwrTrnsfrd`<br>• Positive signals indicate power flow into the block.<br>• Negative signals indicate power flow out of the block. | `PwrMtr` | Mechanical power | $P_{mot}$ | $P_{mot} = \omega_m T_e$ |
| | | `PwrBus` | Electrical power | $P_{bus}$ | $P_{bus} = P_{mot} + P_{loss}$ |
| | `PwrNotTrnsfrd`<br>• Negative signals indicate power loss. | `PwrLoss` | Motor power loss | $P_{loss}$ | $P_{stored} = \omega_m \dot{\omega}_m J$ |

| Bus Signal | | | Description | Variable | Equations |
|---|---|---|---|---|---|
| | `PwrStored`<br>• Positive signals indicate power gain. | `PwrStoredShft` | Motor power stored | $P_{str}$ | $P_{loss} = -(P_{mot} + P_{loss} - P_{stored})$ |

The equations use these variables.

| | |
|---|---|
| $T_e$ | Motor output shaft torque |
| $\omega$ | Motor shaft speed |
| $J$ | Motor inertia |

## Ports

### Input

#### `BattVolt` — Battery voltage

`scalar`

Battery voltage, *BattVolt*, in V.

#### `TrqCmd` — Commanded motor torque

`scalar`

Commanded motor torque, $Trq_{cmd}$, in N·m.

##### Dependencies

To create this input port, for the **Port configuration**, select `Torque`.

#### `MtrSpd` — Motor output shaft speed

`scalar`

Motor shaft speed, $Mtr_{spd}$, in rad/s.

##### Dependencies

To create this input port, for the **Port configuration**, select `Speed`.

### Output

#### `Info` — Bus signal

bus

The bus signal contains these block calculations.

| Signal | | | Description | Units |
|---|---|---|---|---|
| `MechPwr` | | | Mechanical power | rad |
| `PwrLoss` | | | Internal inverter and motor power loss | N·m |
| `PwrInfo` | `PwrTrnsfrd` | `PwrMtr` | Mechanical power | W |
| | | `PwrBus` | Electrical power | W |

| Signal | | | Description | Units |
|---|---|---|---|---|
| | PwrNotTrnsfrd | PwrLoss | Motor power loss | W |
| | PwrStored | PwrStored Shft | Motor power stored | W |

**`BattCurr` — Battery current**
`scalar`

Battery current draw or demand, $I_{batt}$, in A.

**`MtrTrq` — Motor torque**
`scalar`

Motor output shaft torque, $Mtr_{trq}$, in N·m.

**`MtrSpd` — Motor shaft speed**
`scalar`

Motor shaft speed, $Mtr_{spd}$, in rad/s.

**Dependencies**

To create this output port, for the **Port configuration**, select `Torque`.

## Parameters

**Block Options**

**`Port configuration` — Select port configuration**
`Torque` (default) | `Speed`

This table summarizes the port configurations.

| Port Configuration | Creates Ports |
|---|---|
| `Torque` | Outpost `MtrSpd` |
| `Speed` | Input `MtrSpd` |

**`Calibrate Maps` — Calibrate tables with measured data**
`selection`

If you have Model-Based Calibration Toolbox, you can virtually calibrate the measured loss lookup tables.

1. On the **Electrical Losses** tab, set **Parameterize losses by** to either:
   - `Tabulated loss data`
   - `Tabulated loss data with temperature`
2. Click **Calibrate Maps**.

The dialog box steps through these tasks.

<table>
<tr><th>Task</th><th>Description</th></tr>
<tr><td>Import Loss Data</td><td>Import this loss data from a file. For example, open <code><matlabroot>/toolbox/autoblks/autoblksshared/mbctemplates/MappedMotorDataset.xlsx</code>.<br><br>For more information, see “Using Data” (Model-Based Calibration Toolbox).<br><br>
<table>
<tr><th>Parameterize losses by</th><th>Required Data</th></tr>
<tr><td><code>Tabulated loss data</code></td><td>• Motor speed, rad/s<br>• Motor torque, N·m<br>• Power loss, W</td></tr>
<tr><td><code>Tabulated loss data with temperature</code></td><td>• Motor speed, rad/s<br>• Motor torque, N·m<br>• Motor temperature, K<br>• Power loss, W</td></tr>
</table>
<br>Collect motor data at steady-state operating conditions. Data should cover the motor speed, torque, and temperature operating range.<br><br>To filter or edit the data, select Edit in Application. The Model-Based Calibration Toolbox Data Editor opens.</td></tr>
<tr><td>Generate Response Models</td><td>Model-Based Calibration Toolbox uses test plans to fit data to Gaussian process models (GPMs).<br><br>To assess or adjust the response model fit, select Edit in Application. The Model-Based Calibration Toolbox Model Browser opens. For more information, see “Model Assessment” (Model-Based Calibration Toolbox).</td></tr>
<tr><td>Generate Calibration</td><td>Model-Based Calibration Toolbox calibrates the response models and generates calibrated tables.<br><br>To assess or adjust the calibration, select Edit in Application. The Model-Based Calibration Toolbox CAGE Browser opens. For more information, see “Calibration Lookup Tables” (Model-Based Calibration Toolbox).</td></tr>
</table>

<table>
<tr><th>Task</th><th>Description</th></tr>
<tr><td>Update block parameters</td><td>Update these parameters with the calibration.
<table>
<tr><th>Parameterize losses by</th><th>Parameters</th></tr>
<tr><td><code>Tabulated loss data</code></td><td>• Vector of speeds(w) for tabulated losses, w_eff_bp<br>• Vector of torques (T) for tabulated losses, T_eff_bp<br>• Corresponding losses, losses_table</td></tr>
<tr><td><code>Tabulated loss data with temperature</code></td><td>• Vector of speeds(w) for tabulated losses, w_eff_bp<br>• Vector of torques (T) for tabulated losses, T_eff_bp<br>• Vector of temperatures for tabulated losses, Temp_eff_bp<br>• Corresponding losses, losses_table_3d</td></tr>
</table></td></tr>
</table>

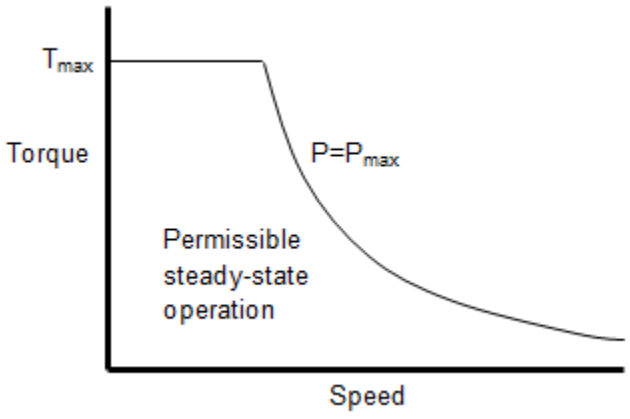
**Electrical Torque**

### `Parameterized by` — Select type

`Tabulated torque-speed envelope` (default) | `Maximum torque and power`

| Setting | Block Implementation |
|---|---|
| `Tabulated torque-speed envelope` | Range specified as a set of speed data points and corresponding maximum torque values. |
| `Maximum torque and power` | Range specified with maximum torque and maximum power. |

For either method, the block implements an envelope similar to this.




### `Vector of rotational speeds, w_t` — Rotational speeds

`[0 375 750 800]` (default) | `vector`

Rotational speeds for permissible steady-state operation, in rad/s. To avoid poor performance due to an infinite slope in the torque-speed curve, specify a vector of rotational speeds that does not contain duplicate consecutive values.

#### Dependencies

To create this parameter, for the **Parameterized by** parameter, select `Tabulated torque-speed envelope`.

### `Vector of maximum torque values, T_t` — Torque

`[0.09 0.08 0.07 0]` (default) | `vector`

Maximum torque values for permissible steady state, in N·m.

#### Dependencies

To create this parameter, for the **Parameterized by** parameter, select `Tabulated torque-speed envelope`.

### `Maximum torque, torque_max` — Torque

`.1` (default) | `scalar`

The maximum permissible motor torque, in N·m.

#### Dependencies

To create this parameter, for the **Parameterized by** parameter, select `Maximum torque and power`.

### `Maximum power, power_max` — Power

`30` (default) | `scalar`

The maximum permissible motor power, in W.

#### Dependencies

To create this parameter, for the **Parameterized by** parameter, select `Maximum torque and power`.

### `Torque control time constant, Tc` — Time constant

`0.02` (default) | `scalar`

Time constant with which the motor driver tracks a torque demand, in s.

#### Electrical Losses

### `Parameterize losses by` — Select type

`Single efficiency measurement` (default) | `Tabulated loss data` | `Tabulated efficiency data`

| Setting | Block Implementation |
| --- | --- |
| `Single efficiency measurement` | Sum of these terms, measured at a single measurement point:<br>• Fixed losses independent of torque and speed, $P_0$. Use $P_0$ to account for fixed converter losses.<br>• A torque-dependent electrical loss $k\tau^2$, where $k$ is a constant and $\tau$ is the torque. Represents ohmic losses in the copper windings.<br>• A speed-dependent electrical loss $k_w\omega^2$, where $k_w$ is a constant and $\omega$ is the speed. Represents iron losses due to eddy currents. |
| `Tabulated loss data` | Loss lookup table that is a function of motor speeds and load torques.<br>If you have Model-Based Calibration Toolbox, click **Calibrate Maps** to virtually calibrate the 2D lookup tables using measured data. |
| `Tabulated loss data with temperature` | Loss lookup table that is a function of motor speeds, load torques, and operating temperature.<br>If you have Model-Based Calibration Toolbox, click **Calibrate Maps** to virtually calibrate the 3D lookup tables using measured data. |
| `Tabulated efficiency data` | 2D efficiency lookup table that is a function of motor speeds and load torques:<br>• Converts the efficiency values you provide into losses and uses the tabulated losses for simulation.<br>• Ignores efficiency values you provide for zero speed or zero torque. Losses are assumed zero when either torque or speed is zero.<br>• Uses linear interpolation to determine losses. Provide tabulated data for low speeds and low torques, as required, to get the desired level of accuracy for lower power conditions.<br>• Does not extrapolate loss values for speed and torque magnitudes that exceed the range of the table. |

| Setting | Block Implementation |
|---|---|
| `Tabulated efficiency data with temperature` | 3D efficiency lookup table that is a function of motor speeds, load torques, and operating temperature:<br>• Converts the efficiency values you provide into losses and uses the tabulated losses for simulation.<br>• Ignores efficiency values you provide for zero speed or zero torque. Losses are assumed zero when either torque or speed is zero.<br>• Uses linear interpolation to determine losses. Provide tabulated data for low speeds and low torques, as required, to get the desired level of accuracy for lower power conditions.<br>• Does not extrapolate loss values for speed, torque, or temperature magnitudes that exceed the range of the table. |

For best practice, use `Tabulated loss data` instead of `Tabulated efficiency data`:

- Efficiency becomes ill defined for zero speed or zero torque.
- You can account for fixed losses that are still present for zero speed or torque.

**Note** Due to system losses, the motor can draw a current when the motor torque is zero.

**`Motor and drive overall efficiency, eff` — Efficiency**
`100` (default) | `scalar`

The block defines overall efficiency as:

$$\eta = 100\frac{\tau_0\omega_0}{\tau_0\omega_0 + P_0 + k\tau_0^2 + k_w\omega_0^2}$$

The equation uses these variables.

| | |
|---|---|
| $\tau_0$ | Torque at which efficiency is measured |
| $\omega_0$ | Speed at which efficiency is measured |
| $P_0$ | Fixed losses independent of torque or speed |
| $k\tau_0^2$ | Torque-dependent electrical losses |
| $k_w\omega^2$ | Speed-dependent iron losses |

At initialization, the block solves the efficiency equation for $k$. The block neglects losses associated with the rotor damping.

**Dependencies**

To create this parameter, for the **Parameterize losses by** parameter, select `Single efficiency measurement`.

**`Speed at which efficiency is measured, w_eff` — Speed**
`375` (default) | `scalar`

Speed at which efficiency is measured, in rad/s.

#### Dependencies

To create this parameter, for the **Parameterize losses by** parameter, select `Single efficiency measurement`.

### `Torque at which efficiency is measured, T_eff` — Torque

`0.08` (default) | `scalar`

Torque at which efficiency is measured, in N·m.

#### Dependencies

To create this parameter, for the **Parameterize losses by** parameter, select `Single efficiency measurement`.

### `Iron losses, Piron` — Power

`0` (default) | `scalar`

Iron losses at the speed and torque at which efficiency is defined, in W.

#### Dependencies

To create this parameter, for the **Parameterize losses by** parameter, select `Single efficiency measurement`.

### `Fixed losses independent of torque and speed, Pbase` — Power

`0` (default) | `scalar`

Fixed electrical loss associated with the driver when the motor current and torque are zero, in W.

#### Dependencies

To create this parameter, for the **Parameterize losses by** parameter, select `Single efficiency measurement`.

### `Vector of speeds (w) for tabulated losses, w_eff_bp` — Breakpoints

`[-8000 -4000 0 4000 8000]` (default) | 1-by-`M` vector

Speed breakpoints for lookup table when calculating losses, in rad/s. Array dimensions are `1` by the number of speed breakpoints, `M`.

#### Dependencies

To create this parameter, for the **Parameterize losses by** parameter, select one of these:

- `Tabulated loss data`
- `Tabulated loss data with temperature`
- `Tabulated efficiency data`
- `Tabulated efficiency data with temperature`

### `Vector of torques (T) for tabulated losses, T_eff_bp` — Breakpoints

`[0 0.03 0.06 0.09]` (default) | 1-by-`N` vector

Torque breakpoints for lookup table when calculating losses, in N·m. Array dimensions are `1` by the number of torque breakpoints, `N`.

**Dependencies**

To create this parameter, for the **Parameterize losses by** parameter, select one of these:

- `Tabulated loss data`
- `Tabulated loss data with temperature`
- `Tabulated efficiency data`
- `Tabulated efficiency data with temperature`

### `Vector of temperatures for tabulated losses, Temp_eff_bp` — Breakpoints

`[233.15 293.15 373.15]` (default) | 1-by-L vector

Temperature breakpoints for lookup table when calculating losses, in K. Array dimensions are `1` by the number of temperature breakpoints, `L`.

**Dependencies**

To create this parameter, for the **Parameterize losses by** parameter, select one of these:

- `Tabulated loss data with temperature`
- `Tabulated efficiency data with temperature`

### `Corresponding losses, losses_table` — 2D lookup table

M-by-N matrix

Array of values for electrical losses as a function of speed and torque, in W. Each value specifies the losses for a specific combination of speed and torque. The array dimensions must match the speed, `M`, and torque, `N`, breakpoint vector dimensions.

**Dependencies**

To create this parameter, for the **Parameterize losses by** parameter, select `Tabulated loss data`.

### `Corresponding losses, losses_table_3d` — 3D lookup table

M-by-N-by-L array

Array of values for electrical losses as a function of speed, torque, and temperature, in W. Each value specifies the losses for a specific combination of speed, torque, and temperature. The array dimensions must match the speed, `M`, torque, `N`, and temperature, `L`, breakpoint vector dimensions.

**Dependencies**

To create this parameter, for the **Parameterize losses by** parameter, select `Tabulated loss data with temperature`.

### `Corresponding efficiency, efficiency_table` — 2D lookup table

M-by-N matrix

Array of efficiency as a function of speed and torque, in %. Each value specifies the losses for a specific combination of speed and torque. The array dimensions must match the speed, `M`, and torque, `N`, breakpoint vector dimensions.

The block ignores efficiency values for zero speed or zero torque. Losses are zero when either torque or speed is zero. The block uses linear interpolation.

To get the desired level of accuracy for lower power conditions, you can provide tabulated data for low speeds and low torques.

**Dependencies**

To create this parameter, for the **Parameterize losses by** parameter, select `Tabulated efficiency data`.

### `Corresponding efficiency, efficiency_table_3d` — 3D lookup table

`M`-by-`N`-by-`L` array

Array of efficiency as a function of speed and torque, in %. Each value specifies the losses for a specific combination of speed and torque. The array dimensions must match the speed, `M`, torque, `N`, and temperature, `L`, breakpoint vector dimensions.

The block ignores efficiency values for zero speed or zero torque. Losses are zero when either torque or speed is zero. The block uses linear interpolation.

To get the desired level of accuracy for lower power conditions, you can provide tabulated data for low speeds and low torques.

**Dependencies**

To create this parameter, for the **Parameterize losses by** parameter, select `Tabulated efficiency data`.

**Mechanical**

### `Rotor inertia, J` — Inertia

`5e-6` (default) | `scalar`

Rotor resistance to change in motor motion, in kg*m$^2$. The value can be zero.

**Dependencies**

To create this parameter, for the **Port configuration** parameter, select `Torque`.

### `Rotor damping, b` — Damping

`1e-5` (default) | `scalar`

Rotor damping, in N·m/(rad/s). The value can be zero.

**Dependencies**

To create this parameter, for the **Port configuration** parameter, select `Torque`.

### `Initial rotor speed, omega_o` — Speed

`0` (default) | `scalar`

Rotor speed at the start of the simulation, in rad/s.

**Dependencies**

To create this parameter, for the **Port configuration** parameter, select `Torque`.

## Extended Capabilities

### C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

## See Also

Flux-Based PMSM | Induction Motor | Interior PMSM | Surface Mount PMSM

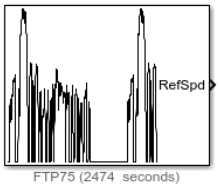
**Introduced in R2017a**

# Scenario Creation Blocks

# Drive Cycle Source

Standard or specified longitudinal drive cycle

**Library:** Powertrain Blockset / Vehicle Scenario Builder
Vehicle Dynamics Blockset / Vehicle Scenarios / Drive Cycle and Maneuvers




## Description

The Drive Cycle Source block generates a standard or user-specified longitudinal drive cycle. The block output is the specified vehicle longitudinal speed, which you can use to:

- Predict the engine torque and fuel consumption that a vehicle requires to achieve desired speed and acceleration for a given gear shift reference.
- Produce realistic velocity and shift references for closed loop acceleration and braking commands for vehicle control and plant models.
- Study, tune, and optimize vehicle control, system performance, and system robustness over multiple drive cycles.
- Identify the faults within tolerances specified by standardized tests, including:
  - EPA dynamometer driving schedules[1]
  - Worldwide Harmonised Light Vehicle Test Procedure (WLTP) laboratory tests[2]

For the drive cycles, you can use:

- Drive cycles from predefined sources. By default, the block includes the `FTP–75` drive cycle. To install additional drive cycles from a support package, see “Install Drive Cycle Data”. The support package has drive cycles that include the gear shift schedules, for example `JC08` and `CUEDC`.
- Workspace variables.
- `.mat`, `.xls`, `.xlsx`, or `.txt` files.
- Wide open throttle (WOT) parameters, including initial and nominal reference speed, deceleration start time, and final reference speed.

To achieve the goals listed in the table, use the specified Drive Cycle Source block parameter options.

| Goal | Action |
|---|---|
| Repeat the drive cycle if the simulation run time exceeds the drive cycle length. | Select **Repeat cyclically**. |

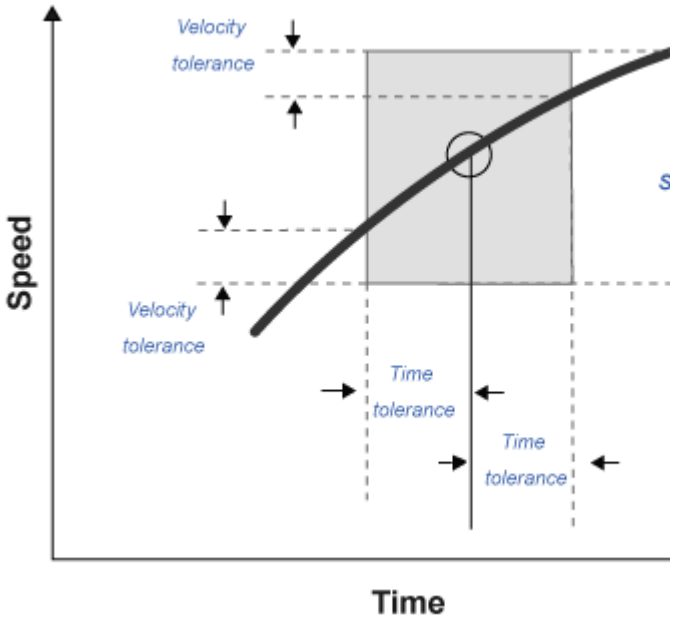
| Goal | Action |
|---|---|
| Output the acceleration, as calculated by Savitzky-Golay differentiation. | Select **Output acceleration**. |
| Specify a sample period for discrete applications. | Specify a **Output sample period (0 for continuous), dt** parameter. |
| Update the simulation run time so that it equals the length of the drive cycle. | Click **Update simulation time**. If a model configuration reference exists, the block does not enable this option. |
| Plot the drive cycle in a MATLAB® figure. | Click **Plot drive cycle**. |
| Specify the drive cycle using a workspace variable. | Click **Specify variable**. The block:<br>• Sets the **Drive cycle source** parameter to `Workspace variable`.<br>• Enables the **From workspace** parameter.<br>Specify the workspace variable so that it contains time, velocity, and, optionally, the gear shift schedule. |
| Specify the drive cycle using a file. | Click **Select file**. The block:<br>• Sets the **Drive cycle source** parameter to `.mat, .xls, .xlsx or .txt file`.<br>• Enables the **Drive cycle source file** parameter.<br>Specify a file that contains time, velocity, and, optionally, the gear shift schedule. |
| Output drive cycle gear. | Specify a drive cycle that contains a gear shift schedule. You can use:<br>• A support package to install standard drive cycles that include the gear shift schedules, for example `JC08` and `CUEDC`.<br>• Workspace variables.<br>• `.mat`, `.xls`, `.xlsx`, or `.txt` files.<br>Click **Output gear shift data**. |
| Install additional drive cycles from a support package. | Click **Install additional drive cycles**. The block enables the parameter if you can install additional drive cycles from a support package. |
| Identify drive cycle faults within tolerances specified by standardized tests. | On the **Fault Tracking** tab, use the parameters to specify the fault tolerances. If the vehicle speed is not within the allowable speed range, the block sets a fault condition. |

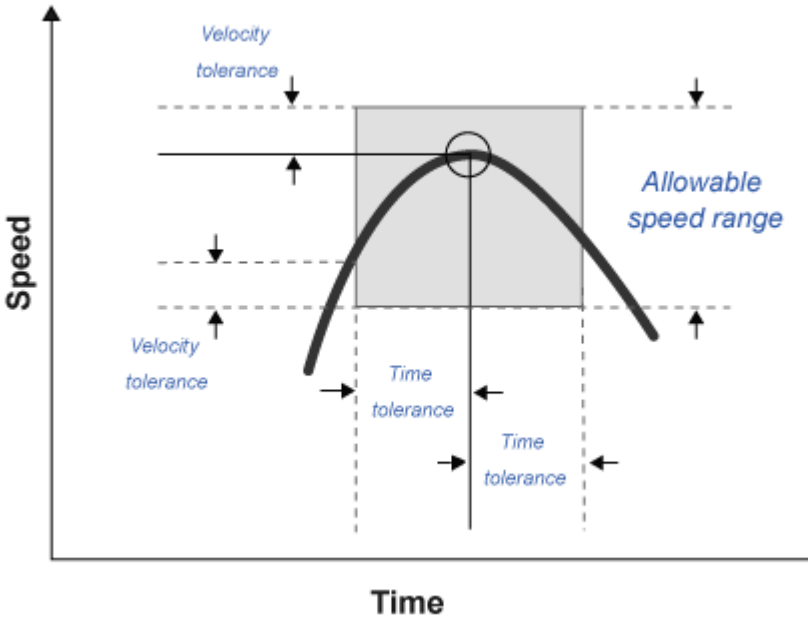
### Fault and Failure Tracking

On the **Fault Tracking** tab, use the parameters to specify the fault tolerances. If the vehicle speed or time is not within the allowable range, the block sets a fault condition.

| Parameter | Description | Setting | |
|---|---|---|---|
| | | EPA Standard[1] | WLTP Tests[2] |
| **Speed tolerance** | Speed tolerance above the highest point and below the lowest point of the drive cycle speed trace within the time tolerance. | 2.0 mph | 2.0 km/h |
| **Time tolerance** | Time that the block uses to determine the speed tolerance. | 1.0 s | 1.0 s |
| **Maximum number of faults** | Maximum number of faults during the drive cycle. | *Not specified* | 10 |
| **Maximum single fault time** | Maximum fault duration. | 2.0 s | 1.0 s |
| **Maximum total fault time** | Maximum accumulated time spent under fault condition. | *Not specified* | *Not specified* |

These figures illustrate how the block uses the velocity and time tolerances to determine the allowable speed range.



## Ports

### Input

#### `VelFdbk` — Vehicle longitudinal speed

`scalar`

Longitudinal vehicle speed.

##### Dependencies

To enable this port, on the **Fault Tracking** tab, select **Enable fault tracking**. Set the **Velocity feedback units, inUnit** parameter to the `VelFdbk` input port signal units.

### Output

#### `Info` — Bus signal

bus

Bus signal containing these block calculations.

| Signal | | Description |
|---|---|---|
| `Reference Speed` | | Vehicle reference speed |
| `Reference Accel` | | Vehicle reference acceleration |
| `Gear` | | Vehicle gear |
| `Fault` | `UpprBnd` | Upper bound of allowable vehicle speed range. |
| | `LowerBnd` | Lower bound of allowable vehicle speed range. |
| | `Fault` | Boolean value indicating fault condition:<br>• 1 — Fault<br>• 0 — No fault<br>If the vehicle speed is not within the allowable speed range, the block sets a fault condition. |
| | `FaultCnt` | Number of faults. |
| | `CumFaultTime` | Cumulative time spent in fault condition. |
| | `SnglFaultTime` | Tim spent in a single fault. |
| | `Fail` | Boolean value indicating fault failure:<br>• 1 — Failure<br>• 0 — No failure<br>If the fault conditions exceed the maximum number of faults, maximum single fault time, or maximum total fault time, the block sets a fault failure. |

##### Dependencies

To enable this port, on the **Fault Tracking** tab, select **Enable fault tracking**.

#### RefSpd — Vehicle reference speed

scalar

Vehicle reference speed, in units that you specify. To specify the units, use the **Output velocity units** parameter.

#### RefAcc — Vehicle reference acceleration

scalar

To calculate the acceleration, the block implements Savitzky-Golay differentiation using a second-order polynomial with a three-sample point filter.

##### Dependencies

To create the output acceleration port, select **Output acceleration**. Selecting **Output acceleration** enables the **Output acceleration units** parameter.

#### Gear — Vehicle gear

scalar

##### Dependencies

To enable this port:

1. Specify a drive cycle that contains a gear shift schedule. You can use:
   - A support package to install standard drive cycles that include the gear shift schedules, for example `JC08` and `CUEDC`.
   - Workspace variables.
   - `.mat`, `.xls`, `.xlsx`, or `.txt` files.
2. Select **Output gear shift** data.

## Parameters

### Cycle Setup

#### Setup

#### Drive cycle source — Select the drive cycle source

`FTP75` (default) | `Wide Open Throttle (WOT)` | `Workspace variable` | `.mat, .xls, .xlsx or .txt file`

- `FTP75` — Load the FTP75 drive cycle from a `.mat` file into a 1-D Lookup Table block. The FTP75 represents a city drive cycle that you can use to determine tailpipe emissions and fuel economy of passenger cars. To install additional drive cycles from a support package, see “Install Drive Cycle Data”.
- `Wide Open Throttle (WOT)` — Use WOT parameters to specify a drive cycle for performance testing.
- `Workspace variable` — Specify time, speed, and, optionally, gear data as a structure, 2-D array, or time series object.
- `.mat, .xls, .xlsx or .txt file` — Specify a file that contains time, speed and, optionally, gear data in column format.

Once you have installed additional cycles, you can use `set_param` to set the drive cycle. For example, to use drive cycle `US06`:

```
set_param([gcs '/Drive Cycle Source'],'cycleVar','US06')
```

#### Dependencies

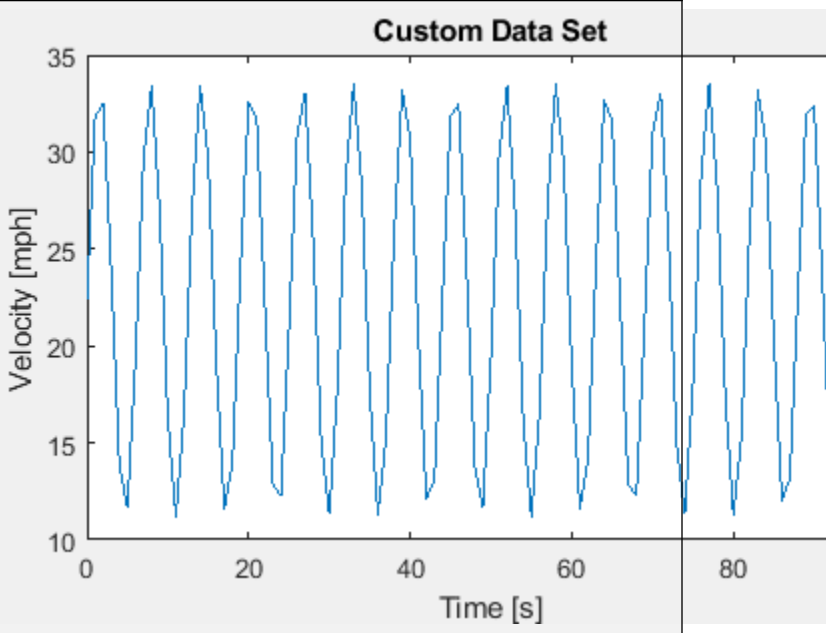
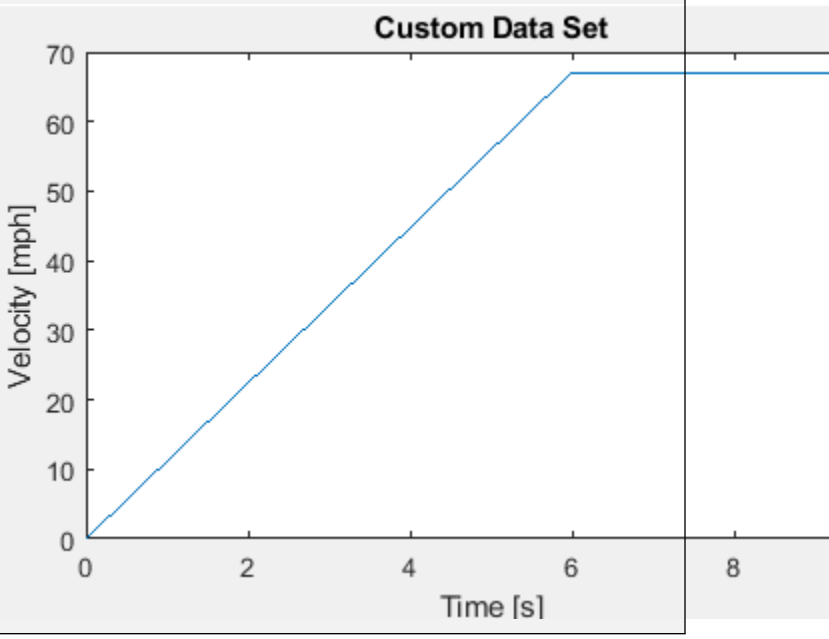
The table summarizes the parameter dependencies.

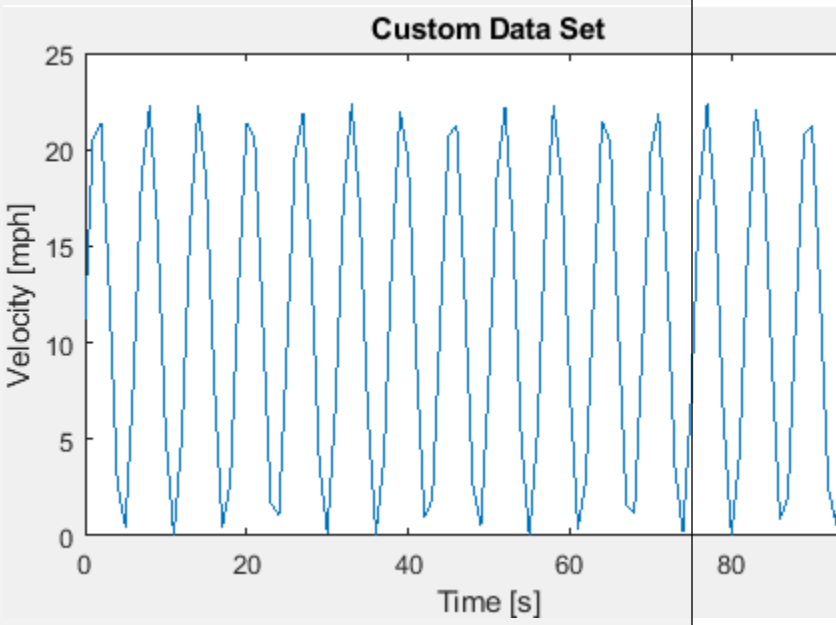
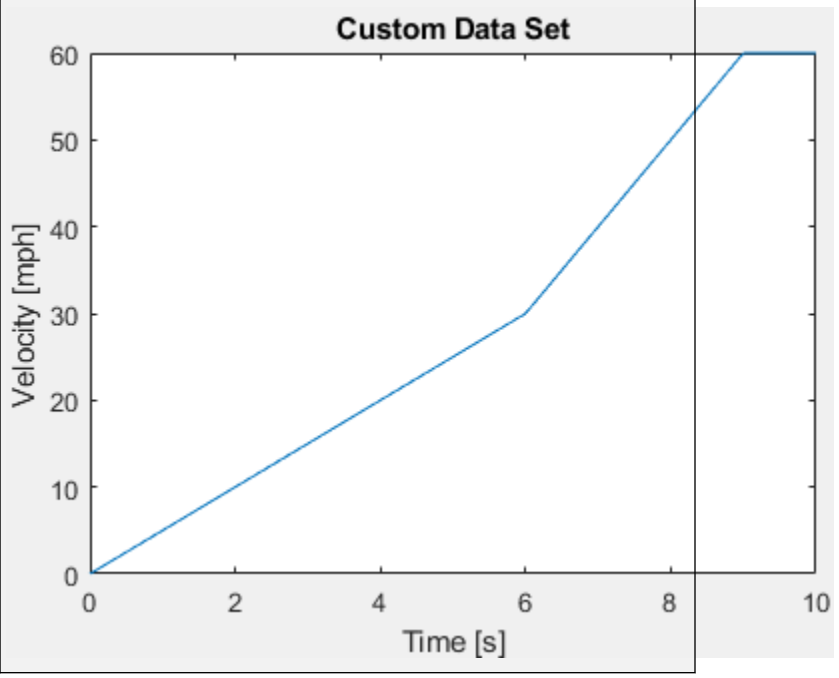
| Drive Cycle Source | Enables Parameter |
|---|---|
| `Wide Open Throttle (WOT)` | **Start time, t_wot1** |
| | **Initial reference speed, xdot_woto** |
| | **Nominal reference speed, xdot_wot1** |
| | **Time to start deceleration, wot2** |
| | **Final reference speed, xdot_wot2** |
| | **WOT simulation time, t_wotend** |
| | **Source velocity units** |
| `Workspace variable` | **From workspace** |
| | **Source velocity units** |
| | **Output gear shift data**, if drive cycle includes gear shift schedule |
| `.mat, .xls, .xlsx or .txt file` | **Drive cycle source file** |
| | **Source velocity units** |
| | **Output gear shift data**, if drive cycle includes gear shift schedule |

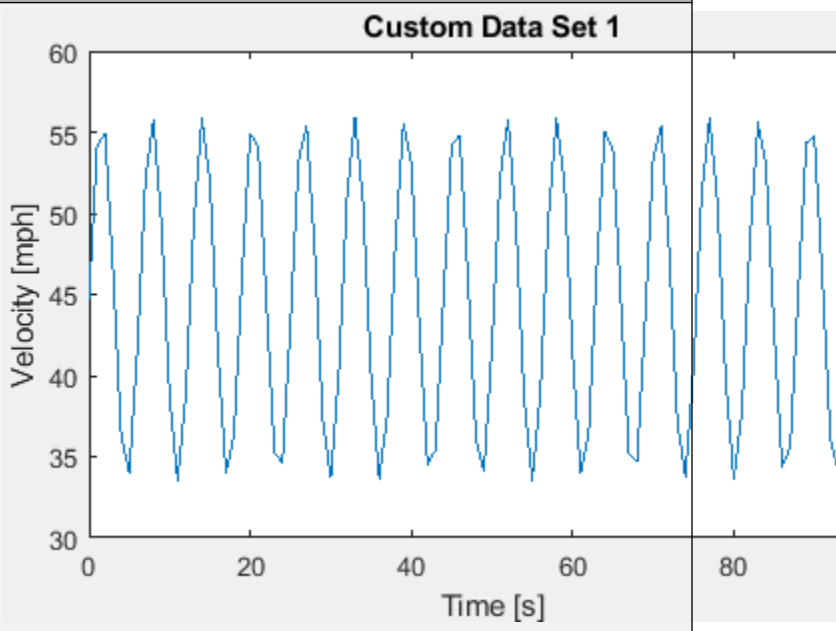
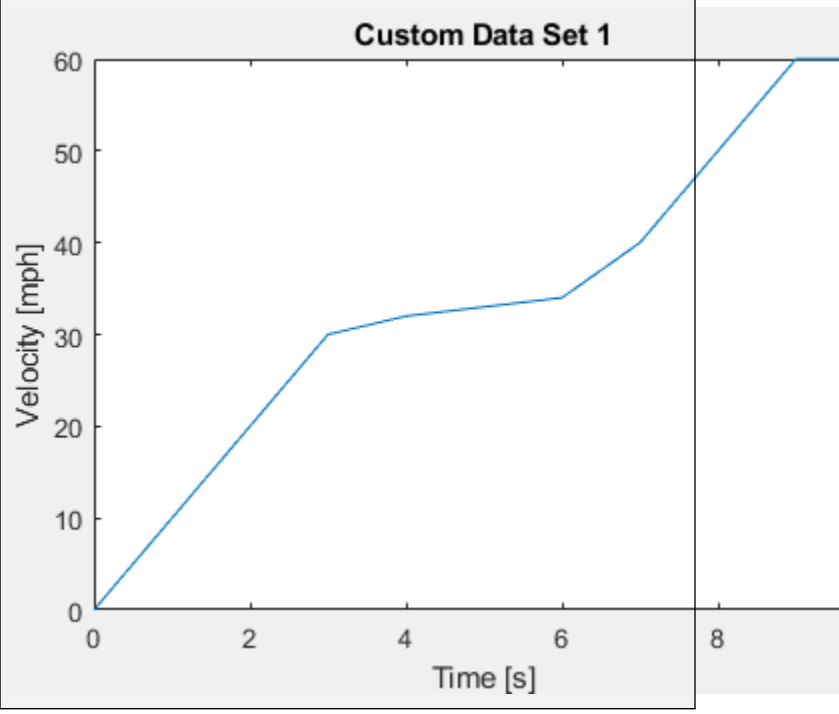
**`From workspace` — Workspace variable**

Monotonically increasing time, velocity, and, optionally, gear data, specified by a structure, 2-D array, or time series object. Enter units for velocity in the **Source velocity units** parameter field.

A valid point must exist for each corresponding time value. You cannot specify `inf`, `empty`, or `NaN`.

| Workspace Variable | Source Velocity Unit | Output Velocity Unit | Drive Cycle Plot |
|---|---|---|---|
| Structure without a gear shift schedule. **From workspace** set to `myCycleS`.<br><br>`t = 0:1:100;`<br>`xdot = 5.*sin(t)+10;`<br>`myCycleS.time = t';`<br>`myCycleS.signals.values = xdot';` | `m/s` | `mph` |  |
| Structure with a gear shift schedule. **From workspace** set to `myCycleS`.<br><br>`gears=[0, 1, 2, 3, 3, 4, 4, 4, 4, 4, 4];`<br>`t=0:1:10;`<br>`xdot=[0,5,10,15,20,25,30,30,30,30,30];`<br>`myCycleS.time=t';`<br>`myCycleS.signals.values=[xdot',gears'];` | `m/s` | `mph` |  |

| Workspace Variable | Source Velocity Unit | Output Velocity Unit | Drive Cycle Plot |
|---|---|---|---|
| 2-D array without a gear shift schedule. **From workspace** set to `myCycleA`.<br><br>`t = 0:1:100;`<br>`xdot = 5.*sin(t)+5;`<br>`myCycleA = [t',xdot'];` | `m/s` | `mph` |  |
| 2-D array with a gear shift schedule. **From workspace** set to `myCycleA`.<br><br>`gears=[0, 1, 2, 3, 4, 4, 4, 5, 5, 5, 5];`<br>`t=0:1:10;`<br>`xdot=[0,5,10,15,20,25,30,40,50,60,60];`<br>`myCycleA=[t',xdot',gears'];` | `mph` | `mph` |  |

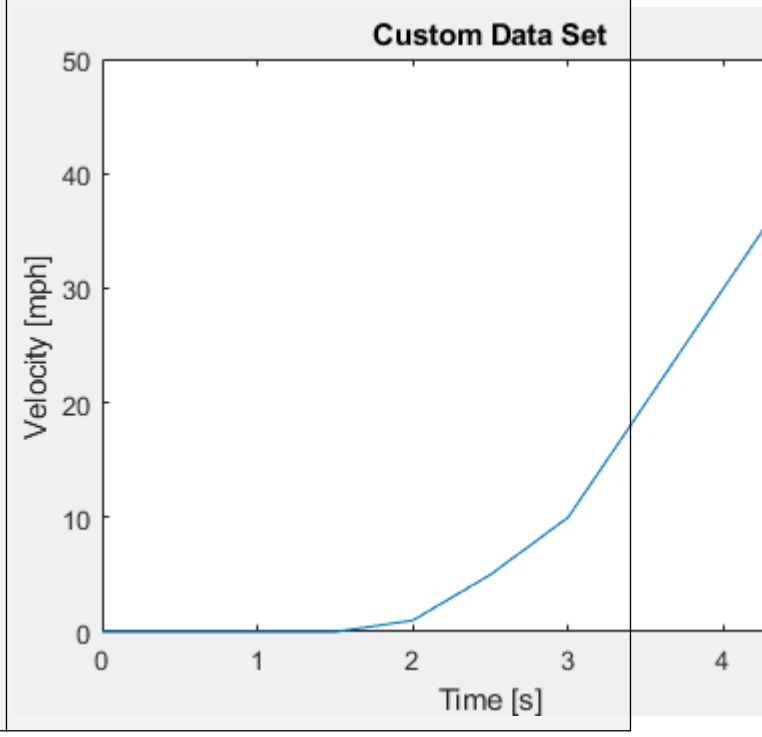
| Workspace Variable | Source Velocity Unit | Output Velocity Unit | Drive Cycle Plot |
|---|---|---|---|
| Time series object without a gear shift schedule. **From workspace** set to `myCycleT`.<br><br>`myCycleT = timeseries;`<br>`t = 0:1:100;`<br>`xdot = 5.*sin(t)+20;`<br>`myCycleT.Data = xdot';`<br>`myCycleT.Time = t;` | `m/s` | `mph` |  |
| Time series object without a gear shift schedule. **From workspace** set to `myCycleT`.<br><br>`myCycleT = timeseries;`<br>`gears=[0, 1, 2, 3, 4, 4, 4, 5, 5, 5, 5];`<br>`t=0:1:10;`<br>`xdot=[0,10,20,30,32,33,34,40,50,60,60];`<br>`myCycleT.Data = [xdot',gears'];`<br>`myCycleT.Time = t';` | `mph` | `mph` |  |

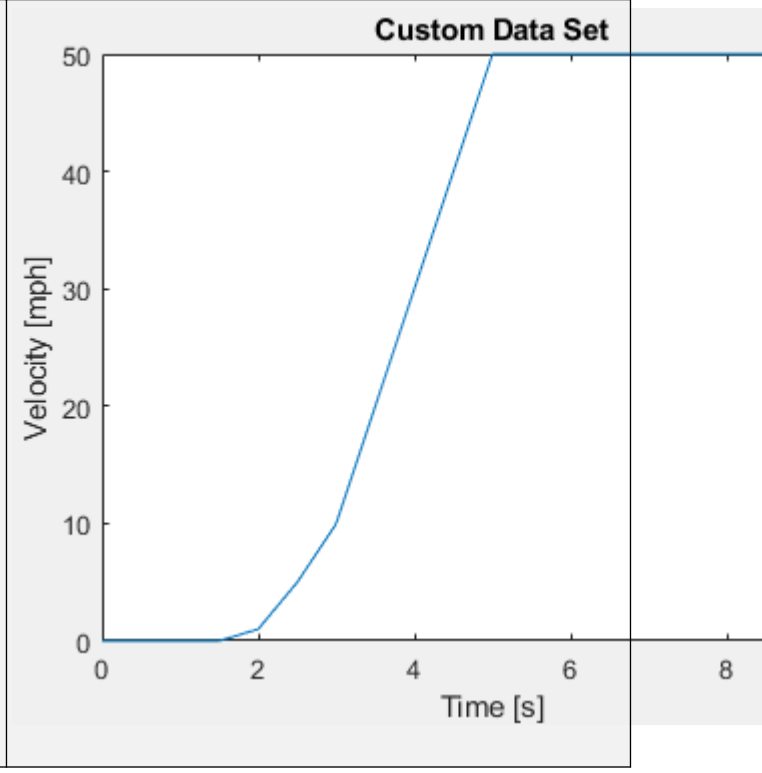
**Dependencies**

To enable this parameter, select `Workspace variable` from **Drive cycle source**.

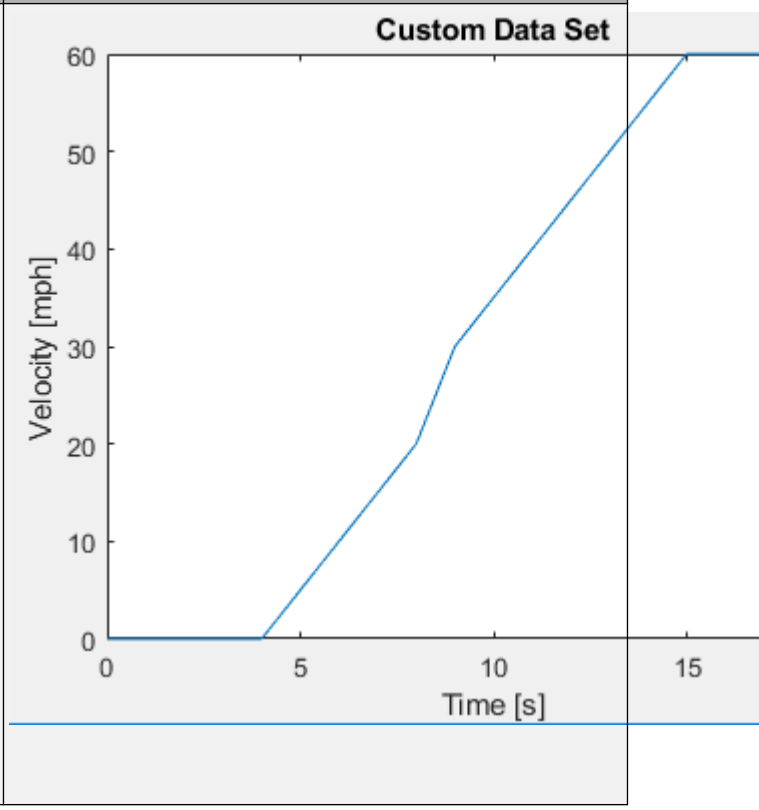
**`Drive cycle source file` — File name**
`.mat, .xls, .xlsx or .txt`

File containing monotonically increasing time, velocity, and, optionally, gear in column or comma-separated format. The block ignores units in the file. Enter units for velocity in the **Source velocity units** parameter field.

| File | Source Velocity Unit | Output Velocity Unit | Drive Cycle Plot |
|---|---|---|---|
| An `.xls` or `.xlsx` file with time in column A and velocity in column B.<br><br>A B<br>1 0 0<br>2 0.5 0<br>3 1 0<br>4 1.5 0<br>5 2 1<br>6 2.5 5<br>7 3 10<br>8 3.5 20<br>9 4 30<br>10 4.5 40<br>11 5 50 | `mph` | `mph` |  |

| File | Source Velocity Unit | Output Velocity Unit | Drive Cycle Plot |
|---|---|---|---|
| An `.xls` or `.xlsx` file with time in column A, velocity in column B, and gear in column C. The block:<br>• Ignores the units in the file.<br>• Converts the gear information to integers:<br>  • N to 0<br>  • D to 2 | `mph` | `mph` |  |

| | A | B | C |
|---|---|---|---|
| 1 | sec | mph | gear |
| 2 | 0 | 0 | N |
| 3 | 0.5 | 0 | N |
| 4 | 1 | 0 | N |
| 5 | 1.5 | 0 | N |
| 6 | 2 | 1 | D |
| 7 | 2.5 | 5 | D |
| 8 | 3 | 10 | D |
| 9 | 3.5 | 20 | D |
| 10 | 4 | 30 | D |
| 11 | 4.5 | 40 | D |
| 12 | 5 | 50 | D |

| File | Source Velocity Unit | Output Velocity Unit | Drive Cycle Plot |
|---|---|---|---|
| A `.txt` with time in column 1 and velocity in column 2. The block ignores the header and units information.<br><br>`Time Speed`<br>`sec mph`<br>`0 0`<br>`1 0`<br>`2 0`<br>`3 0`<br>`4 0`<br>`5 5`<br>`6 10`<br>`7 15`<br>`8 20`<br>`9 30`<br>`10 35`<br>`11 40`<br>`12 45`<br>`13 50`<br>`14 55`<br>`15 60`<br>`16 60`<br>`17 60`<br>`18 60`<br>`19 60`<br>`20 60` | `mph` | `mph` |  |

If you provide the gear schedule using **P**, **R**, **N**, **D**, **L**, **OD**, the block maps the gears to integers.

| Gear | Integer |
|---|---|
| P | `80` |
| R | `-1` |
| N | `0` |
| L | `1` |
| D | `2` |
| OD | Next integer after highest specified gear. |

For example, the block converts the gear schedule `P P N L D 3 4 5 6 5 4 5 6 7 OD 7` to `80 80 0 1 2 3 4 5 6 5 4 5 6 7 8 7`.

#### Dependencies

To enable this parameter, select `.mat, .xls, .xlsx or .txt file` from **Drive cycle source**.

### `Repeat cyclically` — Repeat drive cycle

`off` (default) | `on`

Repeat the drive cycle if the simulation run time exceeds the length of the drive cycle.

### `Output acceleration` — Output the acceleration

`off` (default)

To calculate the acceleration, the block implements Savitzky-Golay differentiation using a second-order polynomial with a three-sample point filter.

#### Dependencies

To create the output acceleration port, select **Output acceleration**. Selecting **Output acceleration** enables the **Output acceleration units** parameter.

### `Output gear shift data` — Output the gear

`off` (default) | `on`

#### Dependencies

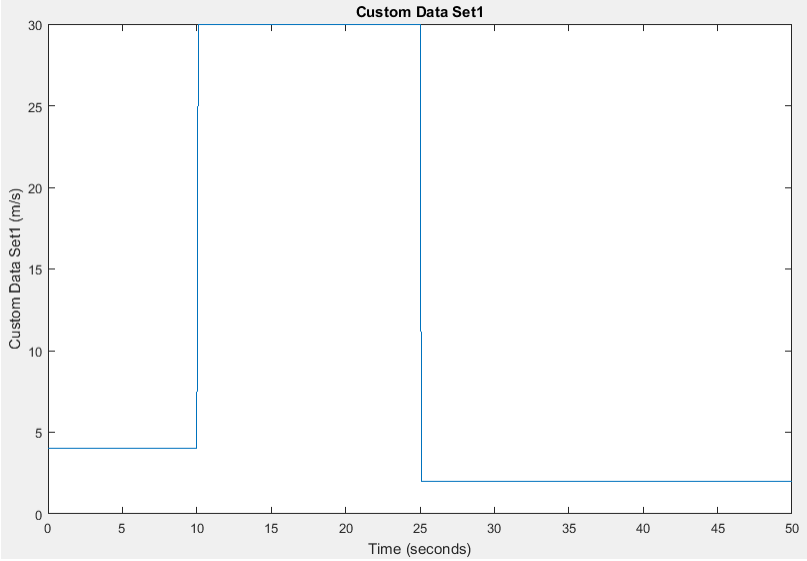
- Specify a drive cycle that contains a gear shift schedule. You can use:
  - A support package to install standard drive cycles that include the gear shift schedules, for example `JC08` and `CUEDC`.
  - Workspace variables.
  - `.mat`, `.xls`, `.xlsx`, or `.txt` files.
- Clicking this parameter creates input port **Gear**.

#### WOT

### `Start time, t_wot1` — Drive cycle start time

`5` (default) | `scalar`

Drive cycle start time, in s. For example, this plot shows a drive cycle with a start time of `10` s.



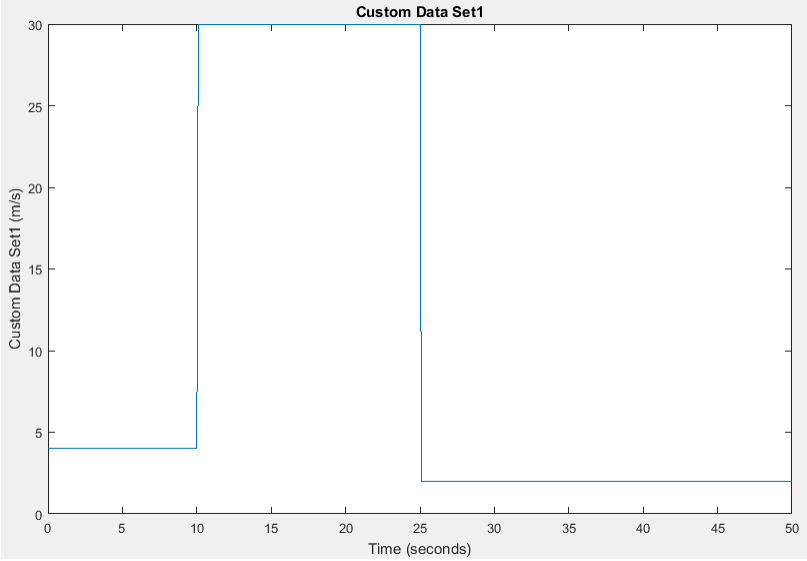

#### Dependencies

To enable this parameter, select the **Drive cycle source** parameter `Wide Open Throttle (WOT)`.

### `Initial reference speed, xdot_woto` — Speed

`0` (default) | `scalar`

Initial reference speed, in units that you specify with the **Source velocity units** parameter. For example, this plot shows a drive cycle with an initial reference speed of 4 m/s.



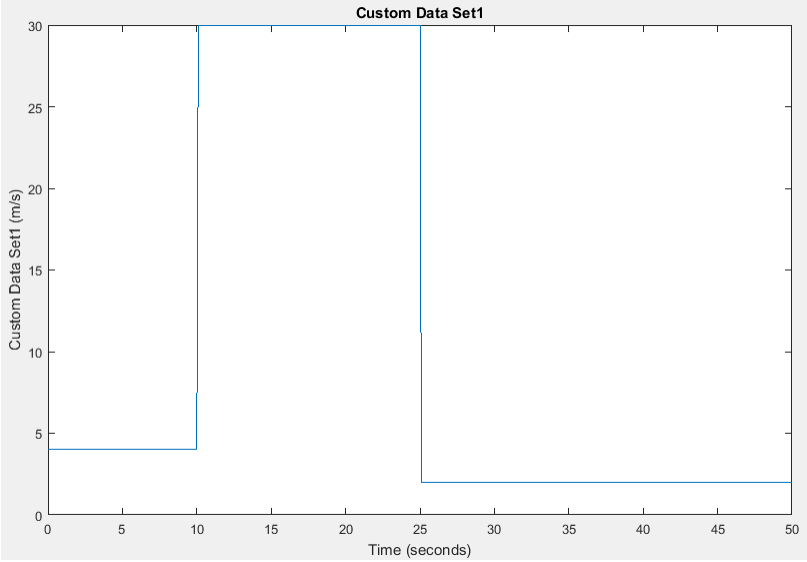

#### Dependencies

To enable this parameter, select the **Drive cycle source** parameter `Wide Open Throttle (WOT)`.

### `Nominal reference speed, xdot_wot1` — Speed

`30` (default) | `scalar`

Nominal reference speed, in units that you specify with the **Source velocity units** parameter. For example, this plot shows a drive cycle with a nominal reference speed of `30` m/s.



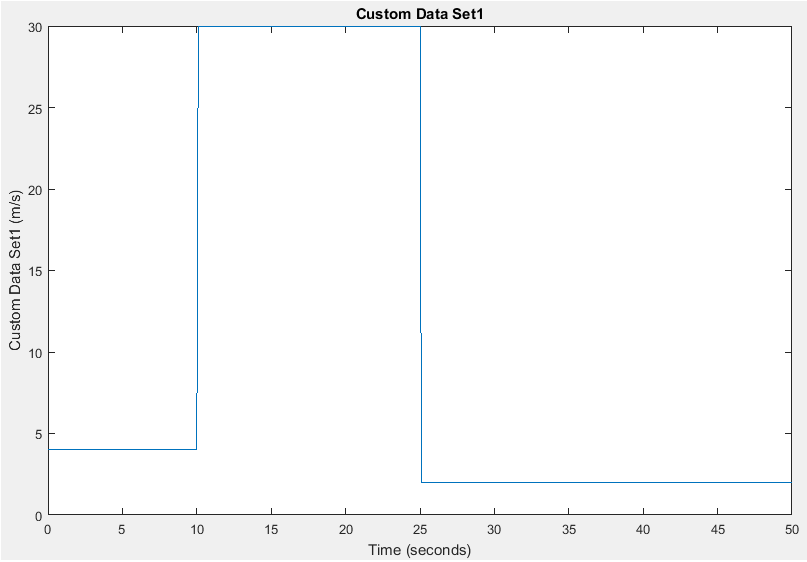
**Dependencies**

To enable this parameter, select the **Drive cycle source** parameter `Wide Open Throttle (WOT)`.

### `Time to start deceleration, wot2` — Time

`20` (default) | `scalar`

Time to start vehicle deceleration, in s. For example, this plot shows a drive cycle with vehicle deceleration starting at 25 s.



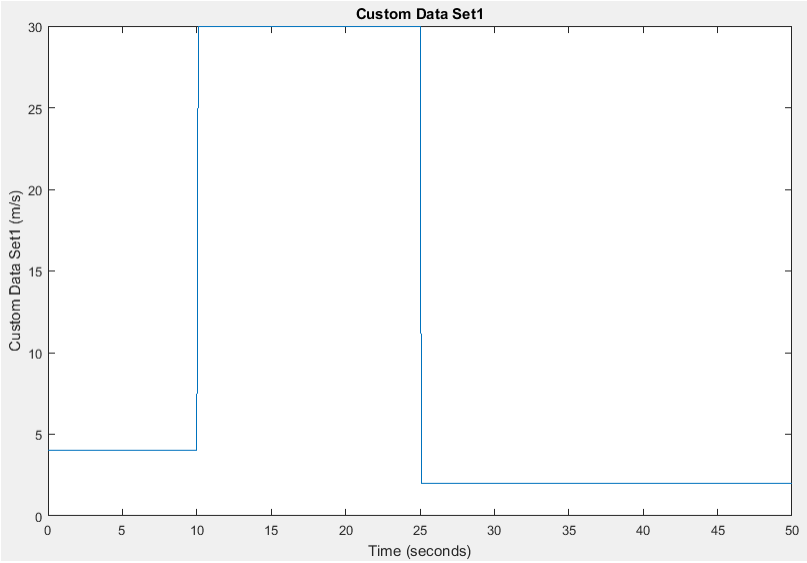

**Dependencies**

To enable this parameter, select the **Drive cycle source** parameter `Wide Open Throttle (WOT)`.

### `Final reference speed, xdot_wot2` — Speed

`0` (default) | `scalar`

Final reference speed, in units that you specify with the **Source velocity units** parameter. For example, this plot shows a drive cycle with a final reference speed of 2 m/s.



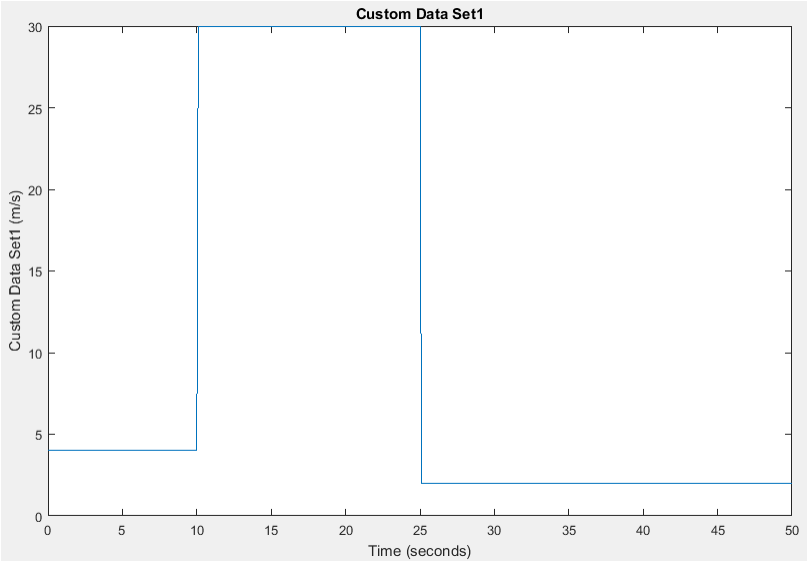

**Dependencies**

To enable this parameter, select the **Drive cycle source** parameter `Wide Open Throttle (WOT)`.

### `WOT simulation time, t_wotend` — Time

`30` (default) | `scalar`

Drive cycle WOT simulation time, in s. For example, this plot shows a drive cycle with a simulation time of `50` s.




**Dependencies**

To enable this parameter, select the **Drive cycle source** parameter `Wide Open Throttle (WOT)`.

#### Units and Sample Period

**`Source velocity units` — Specify velocity units**
`m/s` (default)

Input velocity units.

##### Dependencies

To enable this parameter, select the **Drive cycle source** parameter `Wide Open Throttle (WOT)`, `Workspace variable`, or `.mat, .xls, .xlsx or .txt file`.

**`Output velocity units` — Specify velocity units**
`m/s` (default)

Output velocity units.

**`Output acceleration units` — Specify acceleration units**
`m/s^2` (default)

Specify the output acceleration units.

##### Dependencies

To enable this parameter, select **Output acceleration**.

**`Output sample period (0) for continuous` — Sample rate**
`0` (default) | `scalar`

Sample rate. Set to `0` for continuous sample period. For a discrete period, specify a non-zero rate.

### Fault Tracking

#### Fault Settings

**`Enable fault tracking` — Enable fault tracking**
`off` (default) | `on`

Select this parameter to enable drive cycle fault tracking. Use the parameters to specify the fault tolerances. If the vehicle speed is not within the allowable speed range, the block sets a fault condition.

##### Dependencies

Selecting this parameter enables these parameters:

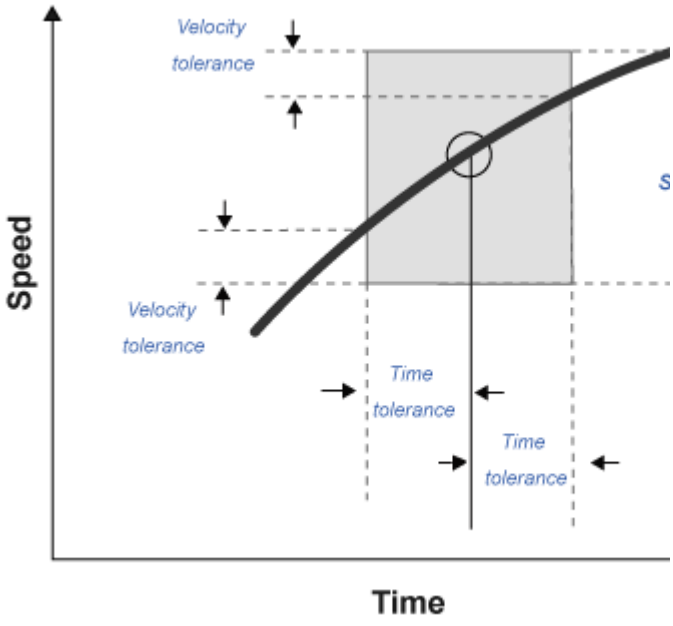
- **Speed tolerance, velBnd**
- **Speed tolerance units, velBndUnit**
- **Velocity feedback units, inUnit**
- **Time tolerance, timeBnd**

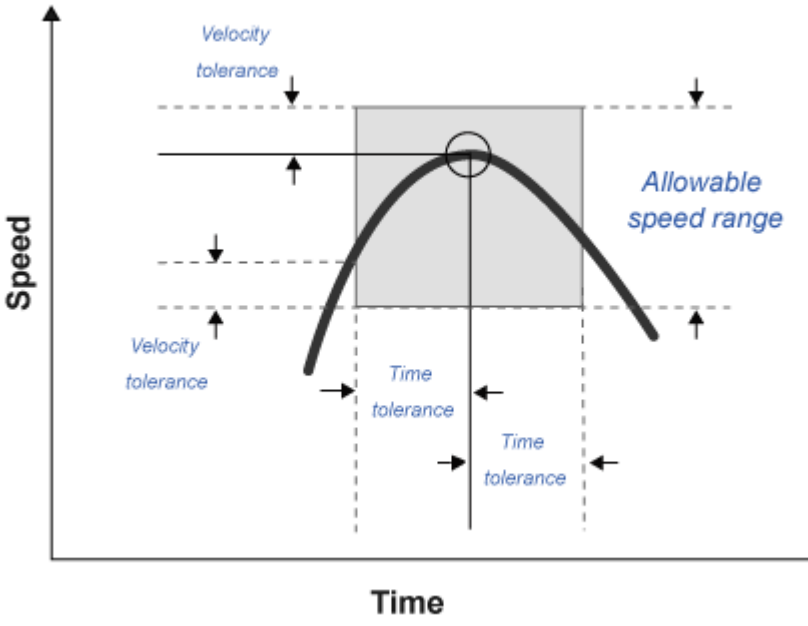
**`Speed tolerance, velBnd` — Drive cycle speed tolerance**
`2.0` (default) | `scalar`

The speed tolerance above the highest point and below the lowest point of the drive cycle speed trace within the time tolerance. If the vehicle speed is not within the allowable speed range, the block sets a fault condition. For the tolerances specified by the standardized tests, use these settings:

- EPA dynamometer driving schedules — `2.0`
- WLTP tests — `2.0`

These figures illustrate how the block uses the velocity and time tolerances to determine the allowable speed range.




**Dependencies**

To enable this parameter, on the **Fault Tracking** tab, select **Enable fault tracking**.

### `Speed tolerance units, velBndUnit` — Set units

`mph` (default)

Speed tolerance units. For the units specified by the standardized tests, use these units:

- EPA dynamometer driving schedules — `m/s`
- WLTP tests — `km/h`

**Dependencies**

To enable this parameter, on the **Fault Tracking** tab, select **Enable fault tracking**.

### `Velocity feedback units, inUnit` — Set velocity feedback units

`m/s` (default)

Velocity feedback units. Set the value to the `VelFdbk` input port signal units.

**Dependencies**

To enable this parameter, on the **Fault Tracking** tab, select **Enable fault tracking**.

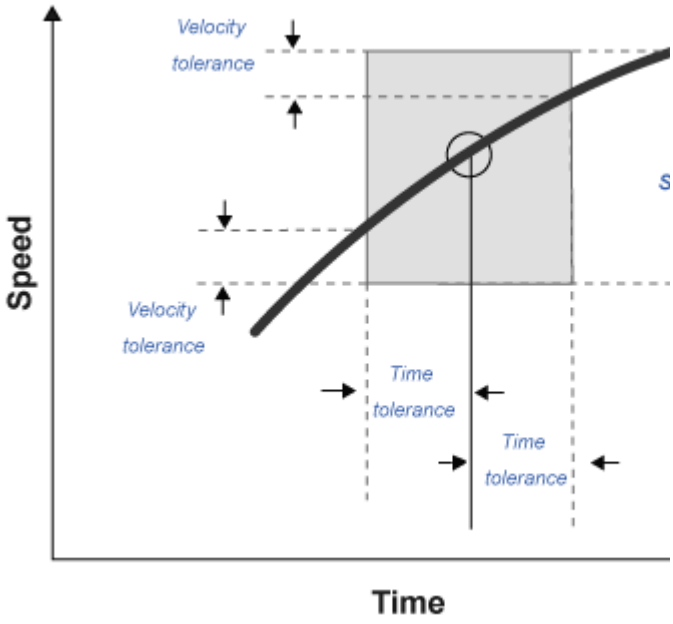
### `Time tolerance, timeBnd` — Time tolerance

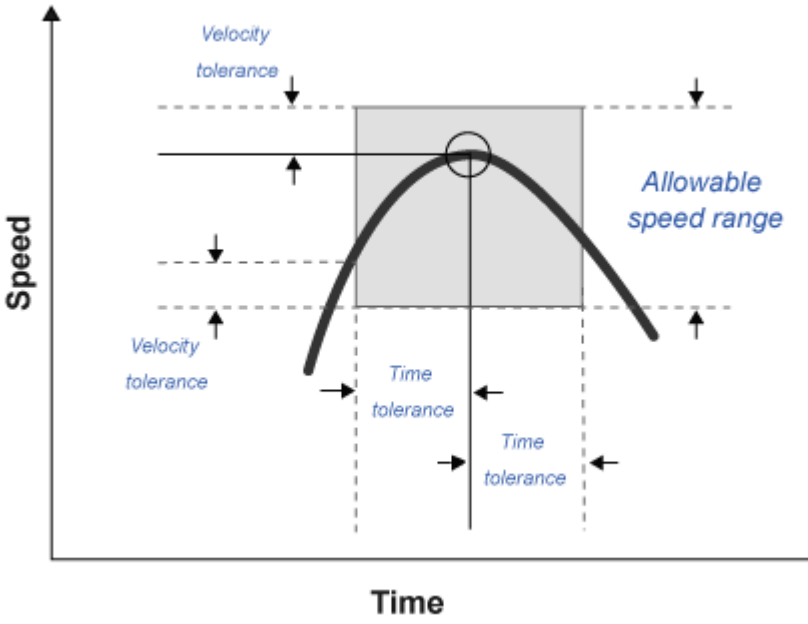
`1.0` (default) | `scalar`

Time that the block uses to determine the speed tolerance. If the vehicle speed is not within the allowable speed range, the block sets a fault condition. For the time tolerances specified by the standardized tests, use these settings:

- EPA dynamometer driving schedules — `1.0`
- WLTP tests — `1.0`

These figures illustrate how the block uses the velocity and time tolerances to determine the allowable speed range.




#### Dependencies

To enable this parameter, on the **Fault Tracking** tab, select **Enable fault tracking**.

#### Failure Settings

### `Enable failure tracking` — Enable failure tracking

`off` (default) | `on`

Select this parameter to enable drive cycle failure tracking.

#### Dependencies

To enable this parameter, select **Enable fault tracking**. Selecting **Enable failure tracking** parameter enables these parameters:

- **Stop simulation when trace fails, stopSim**
- **Maximum number of faults, maxFaultCnt**

- **Maximum single fault time, maxFaultTime**
- **Maximum total fault time, maxTotFaultTime**

**`Maximum number of faults, maxFaultCnt` — Maximum number of faults**
`10` (default) | `scalar`

Maximum number of faults during the drive cycle. For the number specified by the standardized tests, use these settings:

- EPA dynamometer driving schedules — *Not specified*
- WLTP tests — `10`

If the number of faults exceeds the maximum number of faults, the block sets a fault failure.

**Dependencies**

To enable this parameter, on the **Fault Tracking** tab, select **Enable failure tracking**.

**`Maximum single fault time, maxFaultTime` — Maximum duration of single fault**
`2.0` (default) | `scalar`

Maximum duration of single fault, in s. For the time specified by the standardized tests, use these settings:

- EPA dynamometer driving schedules — `2.0`
- WLTP tests — `1.0`

If the fault duration exceeds the maximum single fault time, the block sets a fault failure.

**Dependencies**

To enable this parameter, on the **Fault Tracking** tab, select **Enable failure tracking**.

**`Maximum total fault time, maxTotFaultTime` — Maximum total fault time**
`15.0` (default) | `scalar`

Maximum accumulated time spent under fault condition, in s.

If the accumulated time spent under fault condition exceeds the maximum total fault time, the block sets a fault failure.

**Dependencies**

To enable this parameter, on the **Fault Tracking** tab, select **Enable failure tracking**.

**Simulation Trace**

**`Display simulation trace` — Display velocity trace**
`off` (default) | `on`

Select this parameter to display a velocity trace window. Selecting this parameter can slow the simulation time.

**Dependencies**

Selecting this parameter enables these parameters:

- **Simulation trace update rate, dtTrace**
- **Simulation trace display window, traceWindow**

**`Simulation trace update rate, dtTrace` — Trace update rate**
`1` (default) | `scalar`

Simulation trace update rate, in s. Set to `0` for continuous sample period. For a discrete period, specify a non-zero rate.

**Dependencies**

To enable this parameter, on the **Fault Tracking** tab, select **Display simulation trace**.

**`Simulation trace display window, traceWindow` — Trace window update rate**
`10` (default) | `scalar`

Simulation trace window update rate, in s.

**Dependencies**

To enable this parameter, on the **Fault Tracking** tab, select **Display simulation trace**.

## References

[1] Environmental Protection Agency (EPA). *EPA urban dynamometer driving schedule*. 40 CFR 86.115-78, July 1, 2001.

[2] European Union Commission. "Speed trace tolerances". *European Union Commission Regulation*. 32017R1151, Sec 1.2.6.6, June 1, 2017.

## Extended Capabilities

**C/C++ Code Generation**
Generate C and C++ code using Simulink® Coder™.

## See Also

Longitudinal Driver

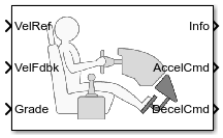
**Topics**
“Install Drive Cycle Data”
“Track Drive Cycle Errors”
“Time Series Objects and Collections”

**Introduced in R2017a**

# Longitudinal Driver

Longitudinal speed-tracking controller

**Library:** Powertrain Blockset / Vehicle Scenario Builder
Vehicle Dynamics Blockset / Vehicle Scenarios / Driver




## Description

The Longitudinal Driver block implements a longitudinal speed-tracking controller. Based on reference and feedback velocities, the block generates normalized acceleration and braking commands that can vary from 0 through 1. You can use the block to model the dynamic response of a driver or to generate the commands necessary to track a longitudinal drive cycle.

### Configurations

#### External Actions

Use the **External Actions** parameters to create input ports for signals that can disable, hold, or override the closed-loop acceleration or deceleration commands. The block uses this priority order for the input commands: disable (highest), hold, override.

This table summarizes the external action parameters.

| Goal | External Action Parameter | Input Ports | Data Type |
|---|---|---|---|
| Override the accelerator command with an input acceleration command. | **Accelerator override** | `EnablAccelOvr` | `Boolean` |
| | | `AccelOvrCmd` | `double` |
| Hold the acceleration command at the current value. | **Accelerator hold** | `AccelHld` | `Boolean` |
| Disable the acceleration command. | **Accelerator disable** | `AccelZero` | `Boolean` |
| Override the decelerator command with an input deceleration command. | **Decelerator override** | `EnablDecelOvr` | `Boolean` |
| | | `DecelOvrCmd` | `double` |
| Hold the decelerator command at current value. | **Decelerator hold** | `DecelHld` | `Boolean` |
| Disable the decelerator command. | **Decelerator disable** | `DecelZero` | `Boolean` |

#### Controller

Use the **Control type, cntrlType** parameter to specify one of these control options.

| Setting | Block Implementation |
|---|---|
| `PI` | Proportional-integral (PI) control with tracking windup and feed-forward gains. |
| `Scheduled PI` | PI control with tracking windup and feed-forward gains that are a function of vehicle velocity. |
| `Predictive` | Optimal single-point preview (look ahead) control model developed by C. C. MacAdam[1, 2, 3]. The model represents driver steering control behavior during path-following and obstacle avoidance maneuvers. Drivers preview (look ahead) to follow a predefined path. To implement the MacAdam model, the block:<br>• Represents the dynamics as a linear single track (bicycle) vehicle<br>• Minimizes the previewed error signal at a single point *T** seconds ahead in time<br>• Accounts for the driver lag deriving from perceptual and neuromuscular mechanisms |

#### Shift

Use the **Shift type, shftType** parameter to specify one of these shift options.

| Setting | Block Implementation |
|---|---|
| `None` | No transmission. Block outputs a constant gear of 1.<br><br>Use this setting to minimize the number of parameters you need to generate acceleration and braking commands to track forward vehicle motion. This setting does not allow reverse vehicle motion. |
| `Reverse, Neutral, Drive` | Block uses a Stateflow® chart to model reverse, neutral, and drive gear shift scheduling.<br><br>Use this setting to generate acceleration and braking commands to track forward and reverse vehicle motion using simple reverse, neutral, and drive gear shift scheduling. Depending on the vehicle state and vehicle velocity feedback, the block uses the initial gear and time required to shift to shift the vehicle up into drive or down into reverse or neutral.<br><br>For neutral gears, the block uses braking commands to control the vehicle speed. For reverse gears, the block uses an acceleration command to generate torque and a brake command to reduce vehicle speed. |

| Setting | Block Implementation |
|---|---|
| `Scheduled` | Block uses a Stateflow chart to model reverse, neutral, park, and N-speed gear shift scheduling.<br><br>Use this setting to generate acceleration and braking commands to track forward and reverse vehicle motion using reverse, neutral, park, and N-speed gear shift scheduling. Depending on the vehicle state and vehicle velocity feedback, the block uses these parameters to determine the:<br><br>• Initial gear<br>• Upshift and downshift accelerator pedal positions<br>• Upshift and downshift velocity<br>• Timing for shifting and engaging forward and reverse from neutral<br><br>For neutral gears, the block uses braking commands to control the vehicle speed. For reverse gears, the block uses an acceleration command to generate torque and a brake command to reduce vehicle speed. |
| `External` | Block uses the input gear, vehicle state, and velocity feedback to generate acceleration and braking commands to track forward and reverse vehicle motion.<br><br>For neutral gears, the block uses braking commands to control the vehicle speed. For reverse gears, the block uses an acceleration command to generate torque and a brake command to reduce vehicle speed. |

#### Gear Signal

Use the **Output gear signal** parameter to create the `GearCmd` output port. The `GearCmd` signal contains the integer value of the commanded vehicle gear.

| Gear | Integer |
|---|---|
| Park | `80` |
| Reverse | `-1` |
| Neutral | `0` |
| Drive | `1` |
| Gear | `Gear number` |

### Controller: PI Speed-Tracking

If you set the control type to `PI` or `Scheduled PI`, the block implements proportional-integral (PI) control with tracking windup and feed-forward gains. For the `Scheduled PI` configuration, the block uses feed forward gains that are a function of vehicle velocity.

To calculate the speed control output, the block uses these equations.

| Setting | Equation |
|---|---|
| `PI` | $$y = \frac{K_{ff}}{v_{nom}} v_{ref} + \frac{K_p e_{ref}}{v_{nom}} + \int \left( \frac{K_i e_{ref}}{v_{nom}} + K_{aw} e_{out} \right) dt + K_g \theta$$ |

| Setting | Equation |
|---|---|
| `Scheduled PI` | $y = \frac{K_{ff}(v)}{v_{nom}} v_{ref} + \frac{K_p(v) e_{ref}}{v_{nom}} + \int \left( \frac{K_i(v) e_{ref}}{v_{nom}} + K_{aw} e_{out} \right) e_{ref} dt + K_g(v)\theta$ |

where:

$e_{ref} = v_{ref} - v$

$e_{out} = y_{sat} - y$

$$y_{sat} = \begin{cases} -1 & y < -1 \\ y & -1 \le y \le 1 \\ 1 & 1 < y \end{cases}$$

The velocity error low-pass filter uses this transfer function.

$$H(s) = \frac{1}{\tau_{err} s + 1} \quad \text{for} \quad \tau_{err} > 0$$

To calculate the acceleration and braking commands, the block uses these equations.

$$y_{acc} = \begin{cases} 0 & y_{sat} < 0 \\ y_{sat} & 0 \le y_{sat} \le 1 \\ 1 & 1 < y_{sat} \end{cases}$$

$$y_{dec} = \begin{cases} 0 & y_{sat} > 0 \\ -y_{sat} & -1 \le y_{sat} \le 0 \\ 1 & y_{sat} < -1 \end{cases}$$

The equations use these variables.

| | |
|---|---|
| $v_{nom}$ | Nominal vehicle speed |
| $K_p$ | Proportional gain |
| $K_i$ | Integral gain |
| $K_{aw}$ | Anti-windup gain |
| $K_{ff}$ | Velocity feed-forward gain |
| $K_g$ | Grade angle feed-forward gain |
| $\theta$ | Grade angle |
| $\tau_{err}$ | Error filter time constant |
| $y$ | Nominal control output magnitude |
| $y_{sat}$ | Saturated control output magnitude |
| $e_{ref}$ | Velocity error |
| $e_{out}$ | Difference between saturated and nominal control outputs |
| $y_{acc}$ | Acceleration signal |
| $y_{dec}$ | Braking signal |
| $v$ | Velocity feedback signal |

| | |
|---|---|
| $v_{ref}$ | Reference velocity signal |

### Controller: Predictive Speed-Tracking

If you set the **Control type, cntrlType** parameter to `Predictive`, the block implements an optimal single-point preview (look ahead) control model developed by C. C. MacAdam[1, 2, 3]. The model represents driver steering control behavior during path-following and obstacle avoidance maneuvers. Drivers preview (look ahead) to follow a predefined path. To implement the MacAdam model, the block:

- Represents the dynamics as a linear single track (bicycle) vehicle
- Minimizes the previewed error signal at a single point T* seconds ahead in time
- Accounts for the driver lag deriving from perceptual and neuromuscular mechanisms

#### Vehicle Dynamics

For longitudinal motion, the block implements these linear dynamics.

$$x_1 = v$$

$$\dot{x}_1 = x_2 = \frac{K_{pt}}{m} - g\sin(\gamma) + F_r x_1$$

In matrix notation:

$$\dot{x} = Fx + g\bar{u}$$

where:

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

$$F = \begin{bmatrix} 0 & 1 \\ \frac{F_r}{m} & 0 \end{bmatrix}$$

$$g = \begin{bmatrix} 0 \\ \frac{K_{pt}}{m} \end{bmatrix}$$

$$\bar{u} = u - \frac{m^2}{K_{pt}} g\sin(\gamma)$$

The block uses this equation for the rolling resistance.

$$F_r = -\left[\tanh(x_1)\left(\frac{a_r}{x_1} + c_r x_1\right) + b_r\right]$$

The single-point model assumes a minimum previewed error signal at a single point $T^*$ seconds ahead in time. $a^*$ is the driver ability to predict the future vehicle response based on the current steering control input. $b^*$ is the driver ability to predict the future vehicle response based on the current vehicle state. The block uses these equations.

$$a* = (T*)m^T\left[I + \sum_{n=1}^{\infty}\frac{F^n(T*)^n}{(n+1)!}\right]ge$$

$$b* = m^T\left[I + \sum_{n=1}^{\infty}\frac{F^n(T*)^n}{n!}\right]$$

where:

$m^T = [1\ \ 1]$

The equations use these variables.

| | |
|---|---|
| $a$, $b$ | Forward and rearward tire location, respectively |
| $m$ | Vehicle mass |
| $I$ | Vehicle rotational inertia |
| $a$*, $\boldsymbol{b}$* | Driver prediction scalar and vector gain, respectively |
| $\boldsymbol{x}$ | Predicted vehicle state vector |
| $v$ | Longitudinal velocity |
| $\boldsymbol{F}$ | System matrix |
| $K_{pt}$ | Tractive force and brake limit |
| $\gamma$ | Grade angle |
| $\boldsymbol{g}$ | Control coefficient vector |
| $g$ | Gravitational constant |
| $T$* | Preview time window |
| $f(t+T*)$ | Previewed path input T* seconds ahead |
| $U$ | Forward vehicle velocity |
| $\boldsymbol{m^T}$ | Constant observer vector; provides vehicle lateral position |
| $F_r$ | Rolling resistance |
| $a_r$ | Static rolling and driveline resistance |
| $b_r$ | Linear rolling and driveline resistance |
| $c_r$ | Aerodynamic rolling and driveline resistance |

#### Optimization

The single-point model implemented by the block finds the steering command that minimizes a local performance index, $J$, over the current preview interval, ($t$, $t$+$T$).

$$J = \frac{1}{T}\int_t^{t+T}[f(\eta) - y(\eta)]^2 d\eta$$

To minimize $J$ with respect to the steering command, this condition must be met.

$$\frac{dJ}{du} = 0$$

You can express the optimal control solution in terms of a current non-optimal and corresponding nonzero preview output error $T$* seconds ahead[1, 2, 3].

$$u^o(t) = u(t) + \frac{e(t + T^*)}{a^*}$$

The block uses the preview distance and vehicle longitudinal velocity to determine the preview time window.

$$T^* = \frac{L}{U}$$

The equations use these variables.

| | |
|---|---|
| $T^*$ | Preview time window |
| $f(t+T^*)$ | Previewed path input $T^*$ sec ahead |
| $y(t+T^*)$ | Previewed plant output $T^*$ sec ahead |
| $e(t+T^*)$ | Previewed error signal $T^*$ sec ahead |
| $u(t)$, $u^o(t)$ | Steer angle and optimal steer angle, respectively |
| $L$ | Preview distance |
| $J$ | Performance index |
| $U$ | Forward (longitudinal) vehicle velocity |

#### Driver Lag

The single-point model implemented by the block introduces a driver lag. The driver lag accounts for the delay when the driver is tracking tasks. Specifically, it is the transport delay deriving from perceptual and neuromuscular mechanisms. To calculate the driver transport delay, the block implements this equation.

$$H(s) = e^{-s\tau}$$

The equations use these variables.

| | |
|---|---|
| $\tau$ | Driver transport delay |
| $y(t+T^*)$ | Previewed plant output $T^*$ sec ahead |
| $e(t+T^*)$ | Previewed error signal $T^*$ sec ahead |
| $u(t)$, $u^o(t)$ | Steer angle and optimal steer angle, respectively |
| $J$ | Performance index |

## Ports

### Input

#### `VelRef` — Reference vehicle velocity

`scalar`

Reference velocity, $v_{ref}$, in m/s.

#### `EnblAccelOvr` — Enable acceleration command override

`scalar`

Enable acceleration command override.

**Dependencies**

To enable this port, select **Acceleration override**.

Data Types: `Boolean`

### `AccelOvrCmd` — Acceleration override command

`scalar`

Acceleration override command, normalized from 0 through 1.

**Dependencies**

To enable this port, select **Acceleration override**.

Data Types: `double`

### `AccelHld` — Acceleration hold

`scalar`

Boolean signal that holds the acceleration command at the current value.

**Dependencies**

To enable this port, select **Acceleration hold**.

Data Types: `Boolean`

### `AccelZero` — Disable acceleration command

`scalar`

Disable acceleration command.

**Dependencies**

To enable this port, select **Acceleration disable**.

Data Types: `Boolean`

### `EnblDecelOvr` — Enable deceleration command override

`scalar`

Enable deceleration command override.

**Dependencies**

To enable this port, select **Deceleration override**.

Data Types: `Boolean`

### `DecelOvrCmd` — Deceleration override command

`scalar`

Deceleration override command, normalized from 0 through 1.

**Dependencies**

To enable this port, select **Deceleration override**.

Data Types: `double`

### `DecelHld` — Deceleration hold

`scalar`

Boolean signal that holds the deceleration command at the current value.

#### Dependencies

To enable this port, select **Deceleration hold**.

Data Types: `Boolean`

### `DecelZero` — Disable deceleration command

`scalar`

Disable deceleration command.

#### Dependencies

To enable this port, select **Deceleration disable**.

Data Types: `Boolean`

### `ExtGear` — Gear

`scalar`

| Gear | Integer |
|---|---|
| Park | `80` |
| Reverse | `-1` |
| Neutral | `0` |
| Drive | `1` |
| Gear | `Gear number` |

#### Dependencies

To enable this port, set **Shift type, shftType** to `External`.

### `VelFdbk` — Longitudinal vehicle velocity

`scalar`

Longitudinal vehicle velocity, $U$, in the vehicle-fixed frame, in m/s.

### `Grade` — Road grade angle

`scalar`

Road grade angle, $\theta$ or $\gamma$, in deg.

## Output

### `Info` — Bus signal

bus

Bus signal containing these block calculations.

| Signal | | Variable | Description |
|---|---|---|---|
| `Accel` | | $y_{acc}$ | Commanded vehicle acceleration, normalized from 0 through 1 |
| `Decel` | | $y_{dec}$ | Commanded vehicle deceleration, normalized from 0 through 1 |
| `Gear` | | | Integer value of commanded gear |
| `Clutch` | | | Clutch command |
| `Err` | | $e_{ref}$ | Difference in reference vehicle speed and vehicle speed |
| `ErrSqrSum` | | $\int_0^t e_{ref}^2 dt$ | Integrated square of error |
| `ErrMax` | | $\max(e_{ref}(t))$ | Maximum error during simulation |
| `ErrMin` | | $\min(e_{ref}(t))$ | Minimum error during simulation |
| `ExtActions` | `EnblAccelOvr` | | Override the accelerator command with an input acceleration command |
| | `AccelOvrCmd` | | Input accelerator override command |
| | `AccelHld` | | Hold the acceleration command at the current value |
| | `AccelZero` | | Disable the acceleration command |
| | `EnblDecelOvr` | | Override the decelerator command with an input deceleration command |
| | `DecelOvrCmd` | | Input deceleration override command |
| | `DecelHld` | | Hold the decelerator command at current value |
| | `DecelZero` | | Disable the decelerator command |

**`AccelCmd` — Commanded vehicle acceleration**
`scalar`

Commanded vehicle acceleration, $y_{acc}$, normalized from 0 through 1.

**`DecelCmd` — Commanded vehicle deceleration**
`scalar`

Commanded vehicle deceleration, $y_{dec}$, normalized from 0 through 1.

**`GearCmd` — Commanded vehicle gear**
`scalar`

Integer value of commanded vehicle gear.

| Gear | Integer |
|---|---|
| Park | `80` |
| Reverse | `-1` |

| Gear | Integer |
|---|---|
| Neutral | `0` |
| Drive | `1` |
| Gear | `Gear number` |

#### Dependencies

To enable this port, select **Output gear signal**.

## Parameters

#### External Actions

#### `Accelerator override` — Override acceleration command

`off` (default) | `on`

Select to override the acceleration command with an input acceleration command.

#### Dependencies

Selecting this parameter creates the `EnblAccelOvr` and `AccelOvrCmd` input ports.

#### `Accelerator hold` — Hold acceleration command

`off` (default) | `on`

Select to hold the acceleration command.

#### Dependencies

Selecting this parameter creates the `AccelHld` input port.

#### `Accelerator disable` — Disable acceleration command

`off` (default) | `on`

Select to disable the acceleration command.

#### Dependencies

Selecting this parameter creates the `AccelZero` input port.

#### `Decelerator override` — Override deceleration command

`off` (default) | `on`

Select to override the deceleration command with an input deceleration command.

#### Dependencies

Selecting this parameter creates the `EnblDecelOvr` and `DecelOvrCmd` input ports.

#### `Decelerator hold` — Hold deceleration command

`off` (default) | `on`

Select to hold the deceleration command.

#### Dependencies

Selecting this parameter creates the `DecelHld` input port.

### `Decelerator disable` — Disable deceleration command

`off` (default) | `on`

Select to disable the deceleration command.

#### Dependencies

Selecting this parameter creates the `DecelZero` input port.

#### Configuration

### `Control type, cntrlType` — Longitudinal control

`PI` (default) | `Scheduled PI` | `Predictive`

Type of longitudinal control.

| Setting | Block Implementation |
|---|---|
| `PI` | Proportional-integral (PI) control with tracking windup and feed-forward gains. |
| `Scheduled PI` | PI control with tracking windup and feed-forward gains that are a function of vehicle velocity. |
| `Predictive` | Optimal single-point preview (look ahead) control model developed by C. C. MacAdam[1, 2, 3]. The model represents driver steering control behavior during path-following and obstacle avoidance maneuvers. Drivers preview (look ahead) to follow a predefined path. To implement the MacAdam model, the block:<br>• Represents the dynamics as a linear single track (bicycle) vehicle<br>• Minimizes the previewed error signal at a single point *T** seconds ahead in time<br>• Accounts for the driver lag deriving from perceptual and neuromuscular mechanisms |

### `Shift type, shftType` — Shift type

`None` (default) | `Reverse, Neutral, Drive` | `Scheduled` | `External`

Shift type.

| Setting | Block Implementation |
|---|---|
| `None` | No transmission. Block outputs a constant gear of 1.<br><br>Use this setting to minimize the number of parameters you need to generate acceleration and braking commands to track forward vehicle motion. This setting does not allow reverse vehicle motion. |

| Setting | Block Implementation |
|---|---|
| `Reverse, Neutral, Drive` | Block uses a Stateflow chart to model reverse, neutral, and drive gear shift scheduling.<br><br>Use this setting to generate acceleration and braking commands to track forward and reverse vehicle motion using simple reverse, neutral, and drive gear shift scheduling. Depending on the vehicle state and vehicle velocity feedback, the block uses the initial gear and time required to shift to shift the vehicle up into drive or down into reverse or neutral.<br><br>For neutral gears, the block uses braking commands to control the vehicle speed. For reverse gears, the block uses an acceleration command to generate torque and a brake command to reduce vehicle speed. |
| `Scheduled` | Block uses a Stateflow chart to model reverse, neutral, park, and N-speed gear shift scheduling.<br><br>Use this setting to generate acceleration and braking commands to track forward and reverse vehicle motion using reverse, neutral, park, and N-speed gear shift scheduling. Depending on the vehicle state and vehicle velocity feedback, the block uses these parameters to determine the:<br><br>• Initial gear<br>• Upshift and downshift accelerator pedal positions<br>• Upshift and downshift velocity<br>• Timing for shifting and engaging forward and reverse from neutral<br><br>For neutral gears, the block uses braking commands to control the vehicle speed. For reverse gears, the block uses an acceleration command to generate torque and a brake command to reduce vehicle speed. |
| `External` | Block uses the input gear, vehicle state, and velocity feedback to generate acceleration and braking commands to track forward and reverse vehicle motion.<br><br>For neutral gears, the block uses braking commands to control the vehicle speed. For reverse gears, the block uses an acceleration command to generate torque and a brake command to reduce vehicle speed. |

**`Reference and feedback units, velUnits` — Velocity units**
m/s (default)

Vehicle velocity reference and feedback units.

#### Dependencies

If you set **Control type, cntrlType** control type to `Scheduled` or `Scheduled PI`, the block uses the **Reference and feedback units, velUnits** for the **Nominal speed, vnom** parameter dimension.

If you set **Shift Type, shftType** to `Scheduled`, the block uses the **Longitudinal velocity units, velUnits** for these parameter dimensions:

- **Upshift velocity data table, upShftTbl**

- **Downshift velocity data table, dwnShftTbl**

**`Output gear signal` — Create `GearCmd` output port**
`off` (default) | `on`

Specify to create output port `GearCmd`.

**Control**

**Longitudinal**

**`Proportional gain, Kp` — Gain**
`10` (default) | `scalar`

Proportional gain, $K_p$, dimensionless.

**Dependencies**

To create this parameter, set **Control type** to `PI`.

**`Integral gain, Ki` — Gain**
`5` (default) | `scalar`

Proportional gain, $K_i$, dimensionless.

**Dependencies**

To create this parameter, set **Control type** to `PI`.

**`Velocity feed-forward, Kff` — Gain**
`.1` (default) | `scalar`

Velocity feed-forward gain, $K_{ff}$, dimensionless.

**Dependencies**

To create this parameter, set **Control type** to `PI`.

**`Grade angle feed-forward, Kg` — Gain**
`0` (default) | `scalar`

Grade angle feed-forward gain, $K_g$, in 1/deg.

**Dependencies**

To create this parameter, set **Control type** to `PI`.

**`Velocity gain breakpoints, VehVelVec` — Breakpoints**
`[0 100]` (default) | `vector`

Velocity gain breakpoints, *VehVelVec*, dimensionless.

**Dependencies**

To create this parameter, set **Control type** to `Scheduled PI`.

**`Velocity feed-forward gain values, KffVec` — Gain**
`[.1 .1]` (default) | `vector`

Velocity feed-forward gain values, *KffVec*, as a function of vehicle velocity, dimensionless.

**Dependencies**

To create this parameter, set **Control type** to `Scheduled PI`.

**`Proportional gain values, KpVec` — Gain**
`[10 10]` (default) | `vector`

Proportional gain values, *KpVec*, as a function of vehicle velocity, dimensionless.

**Dependencies**

To create this parameter, set **Control type** to `Scheduled PI`.

**`Integral gain values, KiVec` — Gain**
`[5 5]` (default) | `vector`

Integral gain values, *KiVec*, as a function of vehicle velocity, dimensionless.

**Dependencies**

To create this parameter, set **Control type** to `Scheduled PI`.

**`Grade angle feed-forward values, KgVec` — Grade gain**
`[0 0]` (default) | `vector`

Grade angle feed-forward values, *KgVec*, as a function of vehicle velocity, in 1/deg.

**Dependencies**

To create this parameter, set **Control type** to `Scheduled PI`.

**`Nominal speed, vnom` — Nominal vehicle speed**
`5` (default) | `scalar`

Nominal vehicle speed, $v_{nom}$, in units specified by the **Reference and feedback units, velUnits** parameter. The block uses the nominal speed to normalize the controller gains.

**Dependencies**

To create this parameter, set **Control type** to `PI` or `Scheduled PI`.

**`Anti-windup, Kaw` — Gain**
`1` (default) | `scalar`

Anti-windup gain, $K_{aw}$, dimensionless.

**Dependencies**

To create this parameter, set **Control type** to `PI` or `Scheduled PI`.

**`Error filter time constant, tauerr` — Filter**
`.01` (default) | `scalar`

Error filter time constant, $\tau_{err}$, in s. To disable the filter, enter 0.

**Dependencies**

To create this parameter, set **Control type** to `PI` or `Scheduled PI`.

**Predictive**

**`Vehicle mass, m` — Mass**
`1500` (default) | `scalar`

Vehicle mass, $m$, in kg.

**Dependencies**

To create this parameter, set **Longitudinal control type, cntrlType** to `Predictive`.

**`Effective vehicle total tractive force, Kp` — Tractive force**
`3000` (default) | `scalar`

Effective vehicle total tractive force, $K_p$, in N.

**Dependencies**

To create this parameter, set **Longitudinal control type, cntrlType** to `Predictive`.

**`Driver response time, tau` — Tau**
`.1` (default) | `scalar`

Driver response time, $\tau$, in s.

**Dependencies**

To create this parameter, set **Longitudinal control type, cntrlType** to `Predictive`.

**`Preview distance, L` — Distance**
`2` (default) | `scalar`

Driver preview distance, $L$, in m.

**Dependencies**

To create this parameter, set **Longitudinal control type, cntrlType** to `Predictive`.

**`Rolling resistance coefficient, aR` — Resistance**
`200` (default) | `scalar`

Static rolling and driveline resistance coefficient, $a_R$, in N. Block uses the parameter to estimate the constant acceleration or braking effort.

**Dependencies**

To create this parameter, set **Longitudinal control type, cntrlType** to `Predictive`.

**`Rolling and driveline resistance coefficient, bR` — Resistance**
`2.5` (default) | `scalar`

Rolling and driveline resistance coefficient, $b_R$, in N·s/m. Block uses the parameter to estimate the linear velocity-dependent acceleration or braking effort.

#### Dependencies

To create this parameter, set **Longitudinal control type, cntrlType** to `Predictive`.

### `Aerodynamic drag coefficient, cR` — Drag

`.5` (default) | `scalar`

Aerodynamic drag coefficient, $c_R$, in N·s^2/m^2. Block uses the parameter to estimate the quadratic velocity-dependent acceleration or braking effort.

#### Dependencies

To create this parameter, set **Longitudinal control type, cntrlType** to `Predictive`.

### `Gravitational constant, g` — Gravitational constant

`9.81` (default) | `scalar`

Gravitational constant, g, in m/s^2.

#### Dependencies

To create this parameter, set **Longitudinal control type, cntrlType** to `Predictive`.

## Shift

#### Reverse, Neutral, Drive

### `Initial gear, GearInit` — Initial gear

`0` (default) | `scalar`

Integer value of the initial gear. The block uses the initial gear to generate acceleration and braking commands to track forward and reverse vehicle motion.

| Gear | Integer |
|---|---|
| Park | `80` |
| Reverse | `-1` |
| Neutral | `0` |
| Drive | `1` |
| Gear | `Gear number` |

#### Dependencies

To create this parameter, set **Shift type, shftType** to `Reverse, Neutral, Drive` or `Scheduled`. If you specify `Reverse, Neutral, Drive`, the **Initial Gear, GearInit** parameter value can be only `-1`, `0`, or `1`.

### `Time required to shift, tShift` — Time

`.1` (default) | `scalar`

Time required to shift, *tShift*, in s. The block uses the time required to shift to generate acceleration and braking commands to track forward and reverse vehicle motion using reverse, neutral, and drive gear shift scheduling.

#### Dependencies

To create this parameter, set **Shift type, shftType** to `Reverse, Neutral, Drive`.

#### Scheduled

### `Initial gear, GearInit` — Initial gear

`0` (default) | `scalar`

Integer value of the initial gear. The block uses the initial gear to generate acceleration and braking commands to track forward and reverse vehicle motion.

| Gear | Integer |
|---|---|
| Park | `80` |
| Reverse | `-1` |
| Neutral | `0` |
| Drive | `1` |
| Gear | `Gear number` |

#### Dependencies

To create this parameter, set **Shift type, shftType** to `Reverse, Neutral, Drive` or `Scheduled`. If you specify `Reverse, Neutral, Drive`, the **Initial Gear, GearInit** parameter value can be only `-1`, `0`, or `1`.

### `Up and down shift accelerator pedal positions, pdlVec` — Pedal position breakpoints

`[0.1 0.4 0.5 0.9]` (default) | `[1-by-m] vector`

Pedal position breakpoints for lookup tables when calculating upshift and downshift velocities, dimensionless. Vector dimensions are 1 by the number of pedal position breakpoints, `m`.

#### Dependencies

To create this parameter, set **Shift type, shftType** to `Scheduled`.

### `Upshift velocity data table, upShftTbl` — Table

`[m-by-n] array`

Upshift velocity data as a function of pedal position and gear, in units specified by the **Reference and feedback units, velUnits** parameter. Upshift velocities indicate the vehicle velocity at which the gear should increase by 1.

The array dimensions are `m` pedal positions by `n` gears. The first column of data, when `n` equals 1, is the upshift velocity for the neutral gear.

#### Dependencies

To create this parameter, set **Shift type, shftType** to `Scheduled`.

### `Downshift velocity data table, dwnShftTbl` — Table

`[m-by-n] array`

Downshift velocity data as a function of pedal position and gear, in units specified by the **Reference and feedback units, velUnits** parameter. Downshift velocities indicate the vehicle velocity at which the gear should decrease by 1.

The array dimensions are `m` pedal positions by `n` gears. The first column of data, when `n` equals 1, is the downshift velocity for the neutral gear.

**Dependencies**

To create this parameter, set **Shift type, shftType** to `Scheduled`.

**`Time required to shift, tClutch` — Time**
`.5` (default) | `scalar`

Time required to shift, $t_{Clutch}$, in s.

**Dependencies**

To create this parameter, set **Shift type, shftType** to `Scheduled`.

**`Time required to engage reverse from neutral, tRev` — Time**
`.5` (default) | `scalar`

Time required to engage reverse from neutral, $t_{Rev}$, in s.

**Dependencies**

To create this parameter, set **Shift type, shftType** to `Scheduled`.

**`Time required to engage park from neutral, tPark` — Time**
`120` (default) | `scalar`

Time required to engage park from neutral, $t_{Park}$, in s.

**Dependencies**

To create this parameter, set **Shift type, shftType** to `Scheduled`.

## References

[1] MacAdam, C. C. "An Optimal Preview Control for Linear Systems". *Journal of Dynamic Systems, Measurement, and Control*. Vol. 102, Number 3, Sept. 1980.

[2] MacAdam, C. C. "Application of an Optimal Preview Control for Simulation of Closed-Loop Automobile Driving ". *IEEE Transactions on Systems, Man, and Cybernetics*. Vol. 11, Issue 6, June 1981.

[3] MacAdam, C. C. *Development of Driver/Vehicle Steering Interaction Models for Dynamic Analysis*. Final Technical Report UMTRI-88-53. Ann Arbor, Michigan: The University of Michigan Transportation Research Institute, Dec. 1988.

## Extended Capabilities

**C/C++ Code Generation**
Generate C and C++ code using Simulink® Coder™.

## See Also

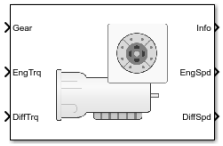
Drive Cycle Source | Vehicle Body Total Road Load

**Introduced in R2017a**

# Transmission Blocks

# Automated Manual Transmission

Ideal automated manual transmission
**Library:** Powertrain Blockset / Transmission / Transmission Systems




## Description

The Automated Manual Transmission block implements an ideal automated transmission (AMT). An AMT is a manual transmission with additional actuators and an electronic control unit (ECU) to regulate clutch and gear selection based on commands from a controller. The number of gears is specified via an integer vector with corresponding gear ratios, inertias, viscous damping, and efficiency factors. The clutch and synchronization engagement rates are linear and adjustable.

Use the block for:

- Power and torque capacity sizing
- Determining gear ratio impact on fuel economy and performance

To determine the rotational drive shaft speed and reaction torque, the Automated Manual Transmission block calculates:

- Clutch lock-up and clutch friction
- Locked rotational dynamics
- Unlocked rotational dynamics

To specify the block efficiency calculation, for **Efficiency factors**, select either of these options.

| Setting | Block Implementation |
|---|---|
| `Gear only` | Efficiency determined from a 1D lookup table that is a function of the gear. |
| `Gear, input torque, input speed, and temperature` | Efficiency determined from a 4D lookup table that is a function of:<br>• Gear<br>• Input torque<br>• Input speed<br>• Oil temperature |

### Clutch Control

The AMT delivers drive shaft torque continuously by controlling the pressure signals from the clutch. If you select **Control type** parameter `Ideal integrated controller`, the block generates idealized clutch pressure signals. To use your own clutch control signals, select **Control type** parameter `External control`.

### Clutch Lock-Up and Clutch Friction

Based on the clutch lock-up condition, the block implements one of these friction models.

| If | Clutch Condition | Friction Model |
|---|---|---|
| $\omega_i \neq N\omega_d$<br>or<br>$T_S < \left\|T_f - Nw_ib_i\right\|$ | Unlocked | $T_f = T_k$<br>where,<br>$T_k = F_cR_{eff}\mu_k\tanh\left[4\left(\frac{w_i}{N} - w_d\right)\right]$<br>$T_s = F_cR_{eff}\mu_s$<br>$R_{eff} = \frac{2(R_o{}^3 - R_i{}^3)}{3(R_o{}^2 - R_i{}^2)}$ |
| $\omega_i = N\omega_t$<br>and<br>$T_S \geq \left\|T_f - Nb_i\omega_i\right\|$ | Locked | $T_f = T_s$ |

The equations use these variables.

| | |
|---|---|
| $\omega_t$ | Output drive shaft speed |
| $\omega_i$ | Input drive shaft speed |
| $\omega_d$ | Drive shaft speed |
| $b_i$ | Viscous damping |
| $F_c$ | Applied clutch force |
| $N$ | Engaged gear |
| $T_f$ | Frictional torque |
| $T_k$ | Kinetic frictional torque |
| $T_s$ | Static frictional torque |
| $R_{eff}$ | Effective clutch radius |
| $R_o$ | Annular disk outer radius |
| $R_i$ | Annular disk inner radius |
| $\mu_s$ | Coefficient of static friction |
| $\mu_k$ | Coefficient of kinetic friction |

### Locked Rotational Dynamics

To model the rotational dynamics when the clutch is locked, the block implements these equations.

$$\dot{\omega}_d J_N = \eta_N T_d - \frac{\omega_i}{N} b_N + NT_i$$

$$\omega_i = N\omega_d$$

The block determines the input torque, $T_i$, through differentiation.

The equations use these variables.

| | |
|---|---|
| $\omega_i$ | Input drive shaft speed |
| $\omega_d$ | Drive shaft speed |
| $N$ | Engaged gear |
| $b_N$ | Engaged gear viscous damping |
| $J_N$ | Engaged gear inertia |
| $\eta_N$ | Engaged gear efficiency |
| $T_d$ | Drive shaft torque |
| $T_i$ | Applied input torque |

### Unlocked Rotational Dynamics

To model the rotational dynamics when the clutch is unlocked, the block implements this equation.

$$\dot{\omega}_d J_N = N T_f - \omega_d b_N + T_d$$

where:

| | |
|---|---|
| $\omega_d$ | Drive shaft speed |
| $N$ | Engaged gear |
| $b_N$ | Engaged gear viscous damping |
| $J_N$ | Engaged gear inertia |
| $T_d$ | Drive shaft torque |
| $T_i$ | Applied input torque |

### Power Accounting

For the power accounting, the block implements these equations.

| Bus Signal | | | Description | Variable | Equations |
|---|---|---|---|---|---|
| `PwrInfo` | `PwrTrnsfrd` — Power transferred between blocks<br>• Positive signals indicate flow into block<br>• Negative signals indicate flow out of block | `PwrEng` | Engine power | $P_{eng}$ | $\omega_i T_i$ |
| | | `PwrDiffrntl` | Differential power | $P_{diff}$ | $\omega_d T_d$ |
| | `PwrNotTrnsfrd` — Power crossing the block boundary, but not transferred<br>• Positive signals indicate an input<br>• Negative signals indicate a loss | `PwrEffLoss` | Mechanical power loss | $P_{effloss}$ | $\omega_d T_d(\eta_N - 1)$ |
| | | `PwrDampLoss` | Mechanical damping loss | $P_{damploss}$ | $-b_N\omega_d^2 - b_{in}\omega_i^2$ |
| | | `PwrCltchLoss` | Clutch power loss | $P_{mech}$ | When locked: 0<br>When unlocked: $-T_k(\omega_i - N\omega_d)$ |

| Bus Signal | | | Description | Variable | Equations |
|---|---|---|---|---|---|
| | `PwrStored` — Stored energy rate of change<br>• Positive signals indicate an increase<br>• Negative signals indicate a decrease | `PwrStoredTrans` | Rate change in rotational kinetic energy | $P_{str}$ | When locked:<br>$\dot{\omega}_i\omega_i(J_{in} + \frac{J_N}{N^2})$<br>When unlocked:<br>$J_{in}\dot{\omega}_i\omega_i + J_N\dot{\omega}_d\omega_d$ |

The equations use these variables.

| | |
|---|---|
| $b_N$ | Engaged gear viscous damping |
| $J_N$ | Engaged gear rotational inertia |
| $J_{in}$ | Flywheel rotational inertia |
| $\eta_N$ | Engaged gear efficiency |
| $N$ | Engaged gear ratio |
| $T_i$ | Applied input torque, typically from the engine crankshaft or dual mass flywheel damper |
| $T_d$ | Applied load torque, typically from the differential or drive shaft |
| $\omega_d$ | Initial input drive shaft rotational velocity |
| $\omega_i$, $\acute{\omega}_i$ | Applied drive shaft angular speed and acceleration |

## Ports

### Input

#### `Gear` — Gear number to engage

`scalar`

Integer value of gear number to engage.

#### `CltchCmd` — Clutch command

`scalar`

Clutch pressure command.

##### Dependencies

To create this port, select **Control type** parameter `External control`.

#### `EngTrq` — Applied input torque

`scalar`

Applied input torque, $T_i$, typically from the engine crankshaft or dual mass flywheel damper, in N·m.

#### `DiffTrq` — Applied load torque

`scalar`

Applied load torque, $T_d$, typically from the differential or driveshaft, in N·m.

### `Temp` — Oil temperature

`scalar`

Oil temperature, in K. To determine the efficiency, the block uses a 4D lookup table that is a function of:

- Gear
- Input torque
- Input speed
- Oil temperature

#### Dependencies

To create this port, set **Efficiency factors** to `Gear, input torque, input speed, and temperature`.

### Output

### `Info` — Bus signal

bus

Bus signal contains these block calculations.

| Signal | | | Description | Variable | Units |
|---|---|---|---|---|---|
| `Eng` | `EngTrq` | | Input applied torque | $T_i$ | N·m |
| | `EngSpd` | | Input drive shaft speed | $\omega_i$ | rad/s |
| `Diff` | `DiffTrq` | | Output drive shaft torque | $T_t$ | N·m |
| | `DiffSpd` | | Output drive shaft speed | $\omega_t$ | rad/s |
| `Cltch` | `CltchForce` | | Applied clutch force | $F_c$ | N |
| | `CltchLocked` | | Clutch lock status, Boolean:<br>• Locked — `0`<br>• Unlocked — `1` | N/A | N/A |
| `Trans` | `TransSpdRatio` | | Speed ratio at time $t$ | $\phi(t)$ | N/A |
| | `TransEta` | | Ratio of output power to input power | $\eta$ | N/A |
| | `TransGearCmd` | | Commanded gear | $N_{cmd}$ | N/A |
| | `TransGear` | | Engaged gear | $N$ | N/A |
| `PwrInfo` | `PwrTrnsfrd` | `PwrEng` | Engine power | $P_{eng}$ | W |
| | | `PwrDiffrntl` | Differential power | $P_{diff}$ | W |
| | `PwrNotTrnsfrd` | `PwrEffLoss` | Mechanical power loss | $P_{effloss}$ | W |

| Signal | | | Description | Variable | Units |
|---|---|---|---|---|---|
| | | PwrDampLoss | Mechanical damping loss | $P_{damploss}$ | W |
| | | PwrCltchLoss | Clutch power loss | $P_{mech}$ | W |
| | PwrStored | PwrStoredTrans | Rate change in rotational kinetic energy | $P_{str}$ | W |

**EngSpd — Angular speed**
`scalar`

Applied drive shaft angular speed input, $\omega_i$, in rad/s.

**DiffSpd — Angular speed**
`scalar`

Drive shaft angular speed output, $\omega_d$, in rad/s.

## Parameters

**`Control type` — Specify control type**
`Ideal integrated controller` (default) | `External control`

The AMT delivers drive shaft torque continuously by controlling the pressure signals from the clutch. If you select **Control type** parameter `Ideal integrated controller`, the block generates idealized clutch pressure signals. To use your own clutch control signals, select **Control type** parameter `External control`.

**Dependencies**

This table summarizes the port configurations.

| Control Mode | Creates Ports |
|---|---|
| `External control` | `CltchCmd` |

**`Efficiency factors` — Specify efficiency calculation**
`Gear only` (default) | `Gear, input torque, input speed, and temperature`

To specify the block efficiency calculation, for **Efficiency factors**, select either of these options.

| Setting | Block Implementation |
|---|---|
| `Gear only` | Efficiency determined from a 1D lookup table that is a function of the gear. |
| `Gear, input torque, input speed, and temperature` | Efficiency determined from a 4D lookup table that is a function of:<br>• Gear<br>• Input torque<br>• Input speed<br>• Oil temperature |

#### Dependencies

| Setting Parameter To | Enables |
|---|---|
| `Gear only` | **Efficiency vector, eta** |
| `Gear, input torque, input speed, and temperature` | **Efficiency torque breakpoints, Trq_bpts**<br>**Efficiency speed breakpoints, omega_bpts**<br>**Efficiency temperature breakpoints, Temp_bpts**<br>**Efficiency lookup table, eta_tbl** |

#### Transmission

**`Input shaft inertia, Jin` — Inertia**
`.01` (default) | `scalar`

Input shaft inertia, in kg·m^2.

**`Input shaft damping, bin` — Damping**
`.001` (default) | `scalar`

Input shaft damping, in N·m·s/rad.

**`Initial input velocity, omegain_o` — Angular velocity**
`0` (default) | `scalar`

Angular velocity, in rad/s.

**`Gear number vector, G` — Specify number of transmission speeds**
`[-1, 0, 1, 2, 3, 4, 5]` (default) | `vector`

Vector of integer gear commands used to specify the number of transmission speeds. Neutral gear is `0`. For example, you can set these parameter values.

| To Specify | Set Gear number, G To |
|---|---|
| Four transmission speeds, including neutral | `[0,1,2,3,4]` |
| Three transmission speeds, including neutral and reverse | `[-1,0,1,2,3]` |
| Five transmission speeds, including neutral and reverse | `[-1,0,1,2,3,4,5]` |

Vector dimensions for the **Gear number vector**, **Gear ratio vector**, **Transmission inertia vector**, **Transmission damping vector**, and **Efficiency vector** parameters must be equal.

**`Efficiency torque breakpoints, Trq_bpts` — Breakpoints**
`[25, 50, 75, 100, 150, 200, 250]` (default) | `vector`

Torque breakpoints for efficiency table, in N·m.

#### Dependencies

To enable this parameter, set **Efficiency factors** to `Gear, input torque, input speed, and temperature`.

### `Efficiency speed breakpoints, omega_bpts` — Breakpoints

`[52.4 78.5 105 131 157 183 209 262 314 419 524]` (default) | `vector`

Speed breakpoints for efficiency table, rad/s.

#### Dependencies

To enable this parameter, set **Efficiency factors** to `Gear, input torque, input speed, and temperature`.

### `Efficiency temperature breakpoints, Temp_bpts` — Breakpoints

`[313 358]` (default) | `vector`

Temperature breakpoints for efficiency table, in K.

#### Dependencies

To enable this parameter, set **Efficiency factors** to `Gear, input torque, input speed, and temperature`.

### `Gear ratio vector, N` — Ratio of input speed to output speed

`[-4.47, 1, 4.47, 2.47, 1.47, 1, 0.8]` (default) | `vector`

Vector of gear ratios (that is, input speed to output speed) with indices corresponding to the ratios specified in **Gear number, G**. For neutral, set the gear ratio to 1. For example, you can set these parameter values.

| To Specify Gear Ratios For | Set Gear number, G To | Set Gear ratio, N To |
|---|---|---|
| Four transmission speeds, including neutral | `[0,1,2,3,4]` | `[1,4.47,2.47,1.47,1]` |
| Five transmission speeds, including neutral and reverse | `[-1,0,1,2,3,4,5]` | `[-4.47,1,4.47,2.47,1.47,1,0.8]` |

Vector dimensions for the **Gear number vector**, **Gear ratio vector**, **Transmission inertia vector**, **Transmission damping vector**, and **Efficiency vector** parameters must be equal.

### `Transmission inertia vector, Jout` — Gear rotational inertia

`[0.128 0.01 0.128 0.1 0.062 0.028 0.01]` (default) | `vector`

Vector of gear rotational inertias, with indices corresponding to the inertias specified in **Gear number, G**, in kg·m^2. For example, you can set these parameter values.

| To Specify Inertia For | Set Gear number, G To | Set Inertia, J To |
|---|---|---|
| Four gears, including neutral | `[0,1,2,3,4]` | `[0.01,2.28,2.04,0.32,0.028]` |
| Inertia for five gears, including reverse and neutral | `[-1,0,1,2,3,4,5]` | `[2.28,0.01,2.28,2.04,0.32,0.028,0.01]` |

Vector dimensions for the **Gear number vector**, **Gear ratio vector**, **Transmission inertia vector**, **Transmission damping vector**, and **Efficiency vector** parameters must be equal.

**`Transmission damping vector, bout` — Gear viscous damping coefficient**
`[.003 .001 .003 .0025 .002 .001 .001]` (default) | `vector`

Vector of gear viscous damping coefficients, with indices corresponding to the coefficients specified in **Gear number, G**, in N·m·s/rad. For example, you can set these parameter values.

| To Specify Damping For | Set Gear number, G To | Set Damping, b To |
|---|---|---|
| Four gears, including neutral | `[0,1,2,3,4]` | `[0.001,0.003,0.0025, 0.002,0.001]` |
| Five gears, including reverse and neutral | `[-1,0,1,2,3,4,5]` | `[0.003,0.001,0.003, 0.0025,0.002,0.001,0.001]` |

Vector dimensions for the **Gear number vector**, **Gear ratio vector**, **Transmission inertia vector**, **Transmission damping vector**, and **Efficiency vector** parameters must be equal.

**`Efficiency vector, eta` — Gear efficiency**
`[0.9, 0.9, 0.9, 0.9, 0.9, 0.95, 0.95]` (default) | `vector`

Vector of gear mechanical efficiency, with indices corresponding to the efficiencies specified in **Gear number, G**. For example, you can set these parameter values.

| To Specify Efficiency For | Set Gear number, G To | Set Efficiency, eta To |
|---|---|---|
| Four gears, including neutral | `[0,1,2,3,4]` | `[0.9,0.9,0.9,0.9,0.95]` |
| Five gears, including reverse and neutral | `[-1,0,1,2,3,4,5]` | `[0.9,0.9,0.9, 0.9,0.9,0.95,0.95]` |

Vector dimensions for the **Gear number vector**, **Gear ratio vector**, **Transmission inertia vector**, **Transmission damping vector**, and **Efficiency vector** parameters must be equal.

**Dependencies**

To enable this parameter, set **Efficiency factors** to `Gear only`.

**`Efficiency lookup table, eta_tbl` — Gear efficiency**
`array`

Table of gear mechanical efficiency, $\eta_N$ as a function of gear, input torque, input speed, and temperature.

**Dependencies**

To enable this parameter, set **Efficiency factors** to `Gear, input torque, input speed, and temperature`.

**`Initial output velocity, omegaout_o` — Transmission**
`0` (default) | `scalar`

Transmission initial output rotational velocity, $\omega_{to}$, in rad/s. If you select **Clutch initially locked**, the block ignores the **Initial output velocity, omega_o** parameter value.

**`Initial gear, G_o` — Engaged gear**
`0` (default) | `scalar`

Initial gear to engage, $G_o$.

#### Clutch and Synchronizer

**`Clutch pressure time constant, tauc` — Time**
`.02` (default) | `scalar`

Pressure input filter time constant, $\tau_c$, in s.

**`Synchronization time, ts` — Time**
`.25` (default) | `scalar`

Time required for gear selection and synchronization, $t_s$, in s.

**`Clutch time, tc` — Time**
`.5` (default) | `scalar`

Time required to engage and disengage the clutch during shift events, $t_c$, in s.

#### Dependencies

To create this parameter, select **Control type** parameter `Ideal integrated controller`.

**`Effective clutch radius, R` — Radius**
`.2` (default) | `scalar`

The effective radius, $R_{eff}$, used with the applied clutch friction force to determine the friction force, in m. The effective radius is defined as:

$$R_{eff} = \frac{2(R_O{}^3 - R_i{}^3)}{3(R_O{}^2 - R_i{}^2)}$$

The equation uses these variables.

| | |
|---|---|
| $R_O$ | Annular disk outer radius |
| $R_i$ | Annular disk inner radius |

**`Clutch force gain, K_c` — Force**
`5e3` (default) | `scalar`

Open loop lock-up clutch gain, $K_c$, in N.

**`Clutch static friction coefficient, mus` — Coefficient**
`0.6` (default) | `scalar`

Dimensionless clutch disc coefficient of static friction, $\mu_s$.

**`Clutch kinematic friction coefficient, muk` — Coefficient**
`0.4` (default) | `scalar`

Dimensionless clutch disc coefficient of kinetic friction, $\mu_k$.

**`Clutch initially locked` — Select to initially lock clutch**
`off` (default) | `on`

Select to lock clutch initially.

#### Dependencies

To create this parameter, select **Control type** parameter `Ideal integrated controller`.

**`Synchronizer initially locked` — Select to initially lock synchronizer**
`off` (default) | `on`

Select to initially lock synchronizer.

## Extended Capabilities

**C/C++ Code Generation**
Generate C and C++ code using Simulink® Coder™.

## See Also

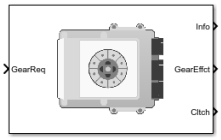
AMT Controller | Continuously Variable Transmission | Dual Clutch Transmission | Ideal Fixed Gear Transmission

#### Introduced in R2017a

# AMT Controller

Automated manual transmission controller with clutch open, close, and synchronization timing

**Library:** Powertrain Blockset / Transmission / Transmission Controllers




## Description

The AMT Controller block implements an automated manual transmission (AMT) controller. You can specify the clutch open, close, and synchronization timing parameters. The block determines the clutch commands using integrator-based timers and latching logic that is based on the specified timing parameters and gear request.

## Ports

### Inputs

#### `GearReq` — Gear number to engage

`scalar`

Gear number request, $G_{req}$.

### Output

#### `Info` — Bus signal

bus

Bus signal containing these block calculations.

| Signal | Description | Variable |
|---|---|---|
| `GearReq` | Gear number request | $G_{req}$ |
| `GearEngd` | Nominal gear commanded by the controller | $G_o$ |
| `Cltch` | Clutch pressure command for gears, between `0` and `1` | NA |

#### `GearEffct` — Effective gear for shifting

`scalar`

Effective gear for shifting. The block uses this signal for the smooth application of inertial, efficiency, gear ratio, and damping parameters.

#### `Cltch` — Command for clutch pressure

`scalar`

Clutch pressure command, between `0` and `1`.

## Parameters

**`Initial gear, G_o` — Engaged gear**
`0` (default) | `scalar`

Initial gear to engage, $G_o$.

**`Clutch actuation time, tc` — Time**
`.1` (default) | `scalar`

Time required to engage and disengage the clutch during shift events, $t_c$, in s.

**`Synchronizer time, ts` — Time**
`.01` (default) | `scalar`

Time required for gear selection and synchronization, $t_s$, in s.

**`Sample period, dt` — Time**
`-1` (default) | `scalar`

Sample period, $dt$, in s.

**`Clutch initially locked` — Select to initially lock clutch**
`off` (default) | `on`

Selecting this parameter initially locks the clutch.

**`Synchronizer initially locked` — Select to initially lock synchronizer**
`off` (default) | `on`

Selecting this parameter initially locks the synchronizer.

## Extended Capabilities

**C/C++ Code Generation**
Generate C and C++ code using Simulink® Coder™.

## See Also

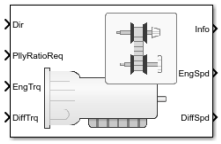
Automated Manual Transmission

**Introduced in R2017a**

# Continuously Variable Transmission

Push belt continuously variable transmission with independent radii control

**Library:** Powertrain Blockset / Transmission / Transmission Systems




## Description

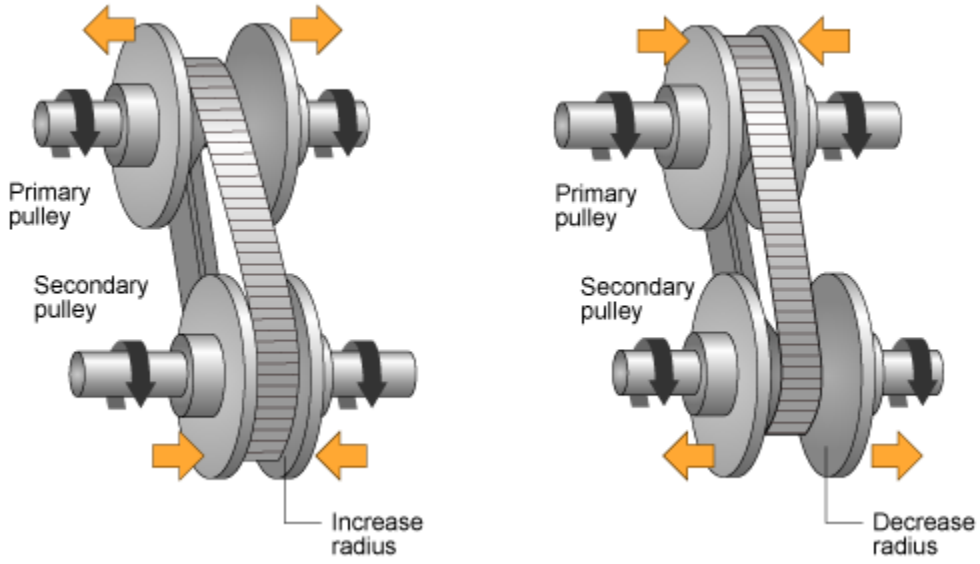
The Continuously Variable Transmission block implements a push belt continuously variable transmission (CVT) with independent radii control. Use the block for control system design, powertrain matching, and fuel economy studies. You can configure the block for internal or external control:

- Internal — Input direction and pulley ratio requests
- External — Input direction and pulley displacement requests

The table summarizes the pulley kinematic, speed reduction, and dynamic calculations made by the Continuously Variable Transmission block.

| Calculation | Pulley Kinematics | Reverse and Final Speed Reduction | Dynamics |
|---|---|---|---|
| Final angular speed ratio | ✓ | ✓ | ✓ |
| Belt torque applied to the secondary and primary pulleys | | | ✓ |
| Torque applied to the secondary and primary pulleys | | ✓ | |
| Angular velocity of secondary and primary pulleys | ✓ | ✓ | ✓ |
| Belt and pulley geometry | ✓ | | |
| Belt linear speed | | | ✓ |
| Wrap angle on secondary and primary pulley | ✓ | | |
| Primary and secondary pulley radii | ✓ | | |

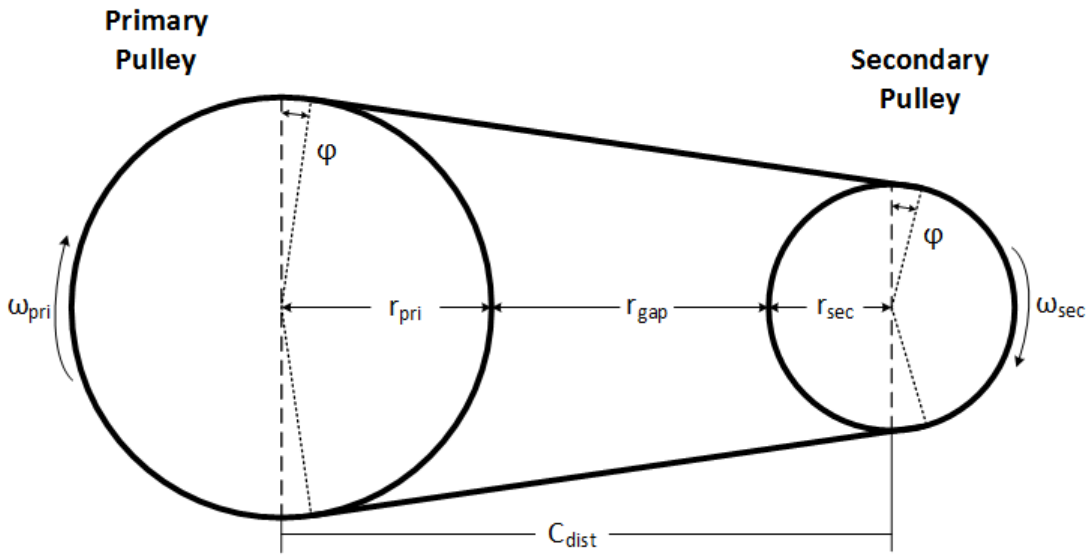
The figure shows the CVT variator with two configurations. In the first configuration, which illustrates speed reduction, the variator is set to decrease the primary pulley radius and increase the secondary pulley radius. In the second configuration, which illustrates overdrive, the variator is set to increase the primary pulley radius and decrease the secondary pulley radius.




### Pulley Kinematics

Using the physical dimensions of the system, the block calculates the primary and secondary variator positions that meet the pulley ratio request.

The figure and equations summarize the geometric dependencies.



$$C_{dist} = rp_{max} + r_{gap} + r_{sec_max}$$
$$L_0 = f(rp_{max}, rs_{max}, rp_{min}, rs_{min}, C_{dist})$$
$$ratio_{command} = f\left(ratio_{request}, ratio_{max}, ratio_{min}\right)$$
$$r_{pri} = f(r_0, ratio_{command}, C_{dist})$$
$$r_{\mathrm{sec}} = f(r_0, ratio_{command}, C_{dist})$$
$$x_{pri} = f\left(r_0, r_{pri}, \theta_{wedge}\right)$$
$$x_{\mathrm{sec}} = f\left(r_0, r_{\mathrm{sec}}, \theta_{wedge}\right)$$

The equations use these variables.

| | |
|---|---|
| $ratio_{request}$ | Pulley gear ratio request |
| $ratio_{command}$ | Pulley gear ratio command, based on request and physical limitations |
| $r_{gap}$ | Gap distance between variator pulleys |
| $C_{dist}$ | Distance between variator pulley centers |
| $rp_{max}$ | Maximum variator primary pulley radius |
| $rs_{max}$ | Maximum variator secondary pulley radius |
| $rp_{min}$ | Minimum variator primary pulley radius |
| $rs_{min}$ | Minimum variator secondary pulley radius |
| $r_o$ | Initial pulley radii with gear ratio of 1 |
| $L_o$ | Initial belt length, resulting from variator specification |
| $x_{pri}$ | Variator primary pulley displacement, resulting from controller request |
| $x_{sec}$ | Variator secondary pulley displacement, resulting from controller request |
| $r_{pri}$ | Variator primary pulley radius, resulting from controller request |
| $r_{sec}$ | Variator secondary pulley radius, resulting from controller request |
| $\Theta_{wedge}$ | Variator wedge angle |
| $\Phi$ | Angle of belt to pulley contact point |
| $L$ | Belt length, resulting from variator position |

### Reverse and Final Speed Reduction

The CVT input shaft connects to a planetary gear set that drives the primary pulley. The shift direction determines the input gear inertia, efficiency, and gear ratio. The shift direction is the filtered commanded direction:

$$\frac{Dir_{shift}}{Dir}(s) = \frac{1}{\tau_s s + 1}$$

For forward motion ($Dir_{shift} = 1$):

$$N_i = 1$$
$$\eta_i = \eta_{fwd}$$
$$J_i = J_{fwd}$$

For reverse motion ($Dir_{shift} = -1$):

$$N_i = -N_{rev}$$
$$\eta_i = \eta_{rev}$$
$$J_i = J_{rev}$$

The gear ratio and efficiency determine the input drive shaft speed and torque applied to the primary pulley:

$$T_{app_pri} = \eta_i N_i T_i$$

The block reduces the secondary pulley speed and applied torque using a fixed gear ratio.

$$T_{app_sec} = \frac{T_o}{\eta_o N_o}$$

$$\omega_o = \frac{\omega_{sec}}{N_o}$$

The final gear ratio, without slip, is given by:

$$N_{final} = \frac{\omega_i}{\omega_o} = N_i N_o \frac{r_{sec}}{r_{pri}}$$

The equations use these variables.

| | |
|---|---|
| $N_i$ | Input planetary gear ratio |
| $Dir$ | CVT direction command |
| $Dir_{shift}$ | Direction used to determine planetary inertia, efficiency, and ratio |
| $\tau_s$ | Direction shift time constant |
| $\eta_{fwd}$, $\eta_{rev}$ | Forward and reverse gear efficiency, respectively |
| $J_{fwd}$, $J_{rev}$ | Forward and reverse gear inertia, respectively |
| $N_{rev}$ | Reverse gear ratio |
| $T_{app_pri}$, $T_{app_sec}$ | Torque applied to primary and secondary pulleys, respectively |
| $T_i$ | Input drive shaft torque |
| $\omega_i$, $\omega_o$ | Input and output drive shaft speed, respectively |
| $\omega_{pri}$, $\omega_{sec}$ | Primary and secondary pulley speed, respectively |
| $N_{final}$ | Total no-slip gear ratio |

### Dynamics

The maximum torque that the CVT can transmit depends on the friction between the pulleys and belt. According to *Prediction of Friction Drive Limit of Metal V-Belt*, the torque friction is defined as:

$$T_{fric}(r_p, \mu) = \frac{2\mu F_{ax} r_p}{\cos(\vartheta_{wedge})}$$

Without macro slip, the tangential acceleration of the pulley is assumed to be equal to the belt acceleration. Once the torque reaches the static friction limit, the belt begins to slip, and the pulley and belt acceleration are independent. During slip, the torque transmitted by the belt is a function of the kinetic friction factor. During the transition from slip to non-slip conditions, the belt and tangential pulley velocities are equal.

The block implements these equations for four different slip conditions.

| Condition | Equations |
|---|---|
| Belt slips on both secondary and primary pulleys | $(J_{pri} + J_i)\dot{\omega}_{pri} = T_{app_pri} - T_{BoP_pri} - b_{pri}\omega_{pri}$<br>$J_{sec}\dot{\omega}_{sec} = T_{app_sec} - T_{BoP_sec} - b_{sec}\omega_{sec}$<br>$m_b\dot{v}_b = \frac{T_{BoP_pri}}{r_{pri}} + \frac{T_{BoP_sec}}{r_{sec}} - b_b v_b$<br>$r_{pri}\omega_{pri} \neq v_b$<br>$r_{sec}\omega_{sec} \neq v_b$ |
| Belt slips on only the primary pulley | $(J_{pri} + J_i)\dot{\omega}_{pri} = T_{app_pri} - T_{BoP_pri} - b_{pri}\omega_{pri}$<br>$\left(m_b + \frac{J_{sec}}{r^2_{sec}}\right)\dot{v}_b = \frac{T_{BoP_pri}}{r_{pri}} + \frac{T_{BoP_sec}}{r_{sec}} - \left(b_b + \frac{b_{sec}}{r^2_{sec}}\right)v_b$<br>$\omega_{sec} = \frac{v_b}{r_{sec}}$<br>$r_{pri}\omega_{pri} \neq v_b$<br>$T_{BoP_pri} = \text{sgn}(r_{pri}\omega_{pri} - v_b)T_{fric}(r_{pri}, \mu_{kin})$<br>$\left\|T_{BoP_sec}\right\| < T_{fric}(r_{sec}, \mu_{static})$ |
| Belt slips on only the secondary pulley | $(m_b + \frac{J_{pri} + J_i}{r^2_{pri}})\dot{v}_b = \frac{T_{app_pri}}{r_{pri}} + \frac{T_{BoP_sec}}{r_{sec}} - \left(b_b + \frac{b_{pri}}{r^2_{pri}}\right)v_b$<br>$J_{sec}\dot{\omega}_b = T_{app_sec} + T_{BoP_sec} - b_{sec}\omega_{sec}$<br>$\omega_{pri} = \frac{v_b}{r_{pri}}$<br>$r_{sec}\omega_{sec} \neq v_b$<br>$T_{BoP_sec} = \text{sgn}(r_{sec}\omega_{sec} - v_b)T_{fric}(r_{sec}, \mu_{kin})$<br>$\left\|T_{BoP_pri}\right\| < T_{fric}(r_{pri}, \mu_{static})$ |
| Belt does not slip | $\left(m_b + \frac{J_{sec}}{r^2_{sec}} + \frac{J_{pri} + J_i}{r^2_{pri}}\right)\dot{v}_b = \frac{T_{app_pri}}{r_{pri}} + \frac{T_{app_sec}}{r_{sec}} - \left(b_b + \frac{b_{sec}}{r^2_{sec}} + \frac{b_{pri}}{r^2_{pri}}\right)v_b$<br>$\omega_{pri} = \frac{v_b}{r_{pri}}$<br>$\omega_{sec} = \frac{v_b}{r_{sec}}$<br>$\left\|T_{BoP_pri}\right\| < T_{fric}(r_{pri}, \mu_{static})$<br>$\left\|T_{BoP_sec}\right\| < T_{fric}(r_{sec}, \mu_{static})$ |

| Condition | Equations |
|---|---|
| Slip direction | $PriSlipDir = \begin{cases} 0 & r_{pri}\omega_{pri} = v_b \\ 1 & r_{pri}\omega_{pri} > v_b \\ -1 & r_{pri}\omega_{pri} < v_b \end{cases}$<br>$SecSlipDir = \begin{cases} 0 & r_{sec}\omega_{sec} = v_b \\ 1 & r_{sec}\omega_{sec} > v_b \\ -1 & r_{sec}\omega_{sec} < v_b \end{cases}$ |

The equations use these variables.

| | |
|---|---|
| $T_{BoP_pri}$, $T_{BoP_sec}$ | Belt torque acting on the primary and secondary pulleys, respectively |
| $T_{app_pri}$, $T_{app_sec}$ | Torque applied to primary and secondary pulleys, respectively |
| $J_{pri}$, $J_{sec}$ | Primary and secondary pulley rotational inertias, respectively |
| $b_{pri}$, $b_{sec}$ | Primary and secondary pulley rotational viscous damping, respectively |
| $F_{ax}$ | Pulley clamp force |
| $\mu$ | Coefficient of friction |
| $\mu_{kin}$, $\mu_{static}$ | Coefficient of kinetic and static friction |
| $v_b$, $a_b$ | Linear speed and acceleration of the belt, respectively |
| $m_b$ | Total belt mass |
| $r_{pri}$, $r_{sec}$ | Radii of the primary and secondary pulleys, respectively |
| $\Phi_{wrap}$ | Wrap angle of belt to pulley contact point |
| $\Phi_{wrap_pri}$, $\Phi_{wrap_sec}$ | Primary and secondary pulley wrap angles, respectively |

### Power Accounting

For the power accounting, the block implements these equations.

| Bus Signal | | | Description | Variable | Equations |
|---|---|---|---|---|---|
| `PwrInfo` | `PwrTrnsfrd` — Power transferred between blocks<br>• Positive signals indicate flow into block<br>• Negative signals indicate flow out of block | `PwrEng` | Engine power | $P_{eng}$ | $\omega_i T_i$ |
| | | `PwrDiffrntl` | Differential power | $P_{diff}$ | $\omega_o T_o$ |
| | `PwrNotTrnsfrd` — Power crossing the block boundary, but not transferred<br>• Positive signals indicate an input<br>• Negative signals indicate a loss | `PwrBltLoss` | Belt slip power loss | $P_{bltloss}$ | $(J_{in} + J_{pri})\dot{\omega}_{pri}\omega_{pri} + J_{sec}\dot{\omega}_{sec}\omega_{sec} + m_b\dot{v}_b v_b + b_{pri}\omega_{pri}^2 + b_{sec}\omega_{sec}^2 + b_b v_b^2 - T_{app_{pri}}\omega_{pri} - T_{app_{sec}}\omega_{sec}$ |

| Bus Signal | | | Description | Variable | Equations |
|---|---|---|---|---|---|
| | | PwrGearInLoss | Input planetary gear mechanical power loss | $P_{grinloss}$ | $-\left\lvert \omega_i T_i - T_{app_pri}\omega_{pri} \right\rvert$ |
| | | PwrGearOutLoss | Output gear reduction mechanical power loss | $P_{groutloss}$ | $-\left\lvert \omega_o T_o - T_{app_sec}\omega_{sec} \right\rvert$ |
| | | PwrDampLoss | Mechanical damping loss | $P_{damploss}$ | $-b_{pri}\omega_{pri}^2 - b_{sec}\omega_{sec}^2 - b_b v_b^2$ |
| | PwrStored — Stored energy rate of change<br>• Positive signals indicate an increase<br>• Negative signals indicate a decrease | PwrStoredTrans | Rate change in rotational kinetic energy | $P_{str}$ | $(J_{in} + J_{pri})\dot{\omega}_{pri}\omega_{pri} + J_{sec}\dot{\omega}_{sec}\omega_{sec} + m_b\dot{v}_b v_b$ |

The equations use these variables.

| | |
|---|---|
| $T_{app_pri}$, $T_{app_sec}$ | Torque applied to primary and secondary pulleys, respectively |
| $T_i$, $T_o$ | Input and output drive shaft torque, respectively |
| $J_{pri}$, $J_{sec}$ | Primary and secondary pulley rotational inertias, respectively |
| $b_{pri}$, $b_{sec}$ | Primary and secondary pulley rotational viscous damping, respectively |
| $\omega_{pri}$, $\omega_{sec}$ | Primary and secondary pulley speed, respectively |
| $\omega_i$, $\omega_o$ | Input and output drive shaft speed, respectively |
| $v_b$, $a_b$ | Linear speed and acceleration of the belt, respectively |
| $r_{pri}$, $r_{sec}$ | Radii of the primary and secondary pulleys, respectively |

## Ports

### Inputs

#### `Dir` — Direction request

`scalar`

Direction request, $Dir_{req}$, controlling the direction. The block filters the request to determine the direction, forward or reverse. *Dir* equals `1` for forward motion. *Dir* equals `-1` for reverse.

$$Dir = \begin{cases} 1 & \text{when } Dir_{req} \geq 0 \\ -1 & \text{when } Dir_{req} < 0 \end{cases}$$

#### `PllyRatioReq` — Pulley ratio request

`scalar`

CVT pulley ratio request, $ratio_{request}$.

#### Dependencies

To create this port, for the **Control mode** parameter, select `Ideal integrated controller`.

### `PriDisp` — Primary pulley displacement

`scalar`

Variator primary pulley displacement, $x_{pri}$, in m.

#### Dependencies

To create this port, for the **Control mode** parameter, select `External control`.

### `SecDisp` — Secondary pulley displacement

`scalar`

Variator secondary pulley displacement, $x_{sec}$, in m.

#### Dependencies

To create this port, for the **Control mode** parameter, select `External control`.

### `EngTrq` — Input drive shaft torque

`scalar`

External torque applied to the input drive shaft, $T_i$, in N·m.

### `DiffTrq` — Output drive shaft torque

`scalar`

External torque applied to the output drive shaft, $T_o$, in N·m.

## Output

### `Info` — Bus signal

bus

Bus signal containing these block calculations.

| Signal | Description | Variable | Units |
|---|---|---|---|
| `EngTrq` | Input shaft torque | $T_i$ | N·m |
| `DiffTrq` | Output shaft torque | $T_o$ | N·m |
| `EngSpd` | Input shaft speed | $\omega_i$ | rad/s |
| `DiffSpd` | Output shaft speed | $\omega_o$ | rad/s |
| `PriRadius` | Primary pulley radius | $r_{pri}$ | m |
| `PriPhi` | Primary pulley wrap angle | $\Phi_{pri}$ | rad |
| `SecRadius` | Secondary pulley radius | $r_{sec}$ | m |
| `SecPhi` | Secondary pulley wrap angle | $\Phi_{sec}$ | rad |
| `BltLngthDelta` | Change in belt length | $\Delta L$ | m |
| `BltLngth` | Belt length | $L$ | m |
| `BltLngthInit` | Initial belt length | $L_o$ | m |

| Signal | | | Description | Variable | Units |
|---|---|---|---|---|---|
| `BltOnPriTrq` | | | Belt torque acting on the primary pulley | $T_{BoP_pri}$ | N·m |
| `BltOnSecTrq` | | | Belt torque acting on the secondary pulley | $T_{BoP_sec}$ | N·m |
| `BltVel` | | | Linear speed of the belt | $v_b$ | m/s |
| `PriAngVel` | | | Primary pulley speed | $\omega_{pri}$ | rad/s |
| `SecAngVel` | | | Secondary pulley speed | $\omega_{sec}$ | rad/s |
| `PriSlipDir` | | | Primary pulley slip direction indicator | *PriSlipDir* | N/A |
| `SecSlipDir` | | | Secondary pulley slip direction indicator | *SecSlipDir* | N/A |
| `TransSpdRatio` | | | Total no-slip gear ratio | $N_{final}$ | N/A |
| `PwrInfo` | `PwrTrnsfrd` | `PwrEng` | Engine power | $P_{eng}$ | W |
| | | `PwrDiffrntl` | Differential power | $P_{diff}$ | W |
| | `PwrNotTrnsfrd` | `PwrBltLoss` | Belt slip power loss | $P_{bltloss}$ | W |
| | | `PwrGearInLoss` | Input planetary gear mechanical power loss | $P_{grinloss}$ | W |
| | | `PwrGearOutLoss` | Output gear reduction mechanical power loss | $P_{groutloss}$ | W |
| | | `PwrDampLoss` | Mechanical damping loss | $P_{damploss}$ | W |
| | `PwrStored` | `PwrStoredTrans` | Rate change in rotational kinetic energy | $P_{str}$ | W |

**`EngSpd` — Input drive shaft speed**
`scalar`

Input drive shaft angular speed, $\omega_i$, in rad/sec.

**`DiffSpd` — Output drive shaft speed**
`scalar`

Output drive shaft angular speed, $\omega_o$, in rad/sec.

## Parameters

**`Control mode` — External or internal**
`Ideal integrated controller` (default) | `External control`

Specify the control method, either internal or external.

**Dependencies**

This table summarizes the port and input model configurations.

| Control Mode | Creates Ports |
|---|---|
| `Ideal integrated controller` | `PllyRatioReq` |
| `External control` | `PriDisp`<br>`SecDisp` |

**Kinematics**

**`Maximum variator primary pulley radius, rp_max` — Radius**
`.08` (default) | `scalar`

Maximum variator primary pulley radius, $rp_{max}$, in m.

**`Maximum variator secondary pulley radius, rs_max` — Radius**
`.07` (default) | `scalar`

Maximum variator secondary pulley radius, $rs_{max}$, in m.

**`Minimum variator primary pulley radius, rp_min` — Radius**
`.03` (default) | `scalar`

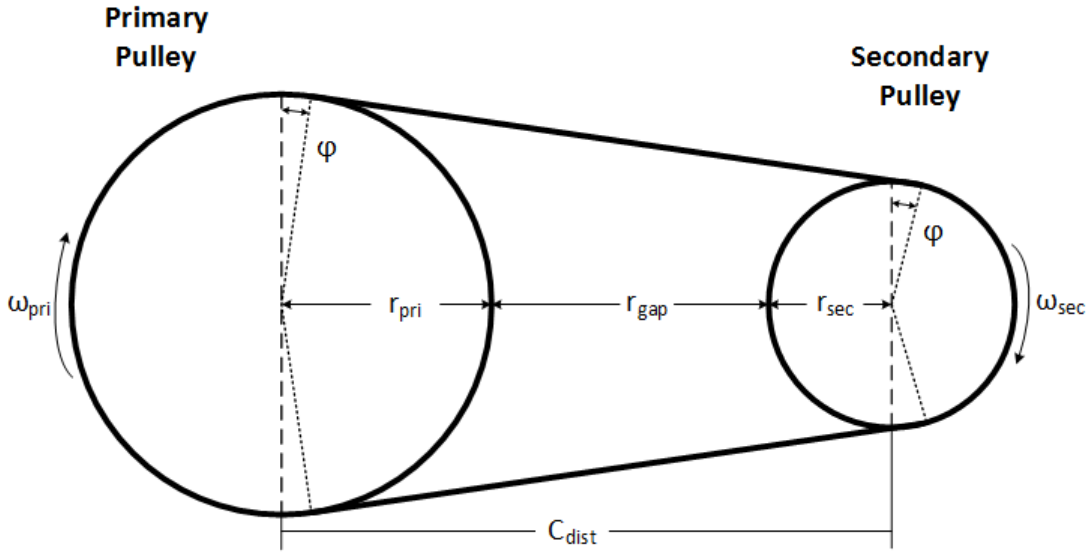
Minimum variator primary pulley radius, $rp_{min}$, in m.

**`Minimum variator secondary pulley radius, rs_min` — Radius**
`.03` (default) | `scalar`

Minimum variator secondary pulley radius, $rs_{min}$, in m.

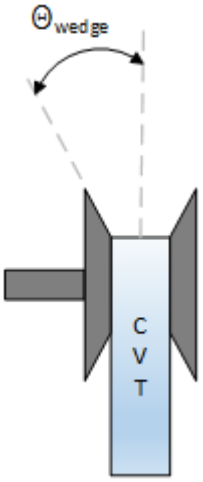
**`Gap distance between variator pulleys, rgap` — Specify crown wheel connection**
`.025` (default) | `scalar`

The gap between the secondary and primary pulleys, $r_{gap}$, in m. The figure shows the pulley geometry.




**`Variator wedge angle, thetawedge` — Specify crown wheel connection**
11 (default) | `scalar`

Variator wedge angle, $\Theta_{wedge}$, in deg.




**Dynamics**

**`Primary pulley inertia, J_pri` — Inertia**
`0.1` (default) | `scalar`

Primary pulley inertia, $J_{pri}$, in kg·m^2.

**`Secondary pulley inertia, J_sec` — Inertia**
`0.1` (default) | `scalar`

Secondary pulley inertia, $J_{sec}$, in kg·m^2.

**`Primary pulley damping coefficient, b_pri` — Damping**
`0.001` (default) | `scalar`

Primary pulley damping coefficient, $b_{pri}$, in N·m·s/rad.

**`Secondary pulley damping coefficient, b_sec` — Damping**
`0.001` (default) | `scalar`

Secondary pulley damping coefficient, $b_{sec}$, in N·m·s/rad.

**`Belt damping coefficient, b_b` — Damping**
`0.0025` (default) | `scalar`

Belt damping coefficient, $b_b$, in kg/s.

**`Static friction coefficient, mu_static` — Friction**
`0.3` (default) | `scalar`

Static friction coefficient between the belt and primary pulley, $\mu_{static}$, dimensionless.

**`Kinetic friction coefficient, mu_kin` — Friction**
`0.2` (default) | `scalar`

Kinetic friction coefficient between the belt and primary pulley, $\mu_{kin}$, dimensionless.

**Belt mass, m_b — Mass**
3 (default) | scalar

Belt mass, $m_b$, in kg.

**Pulley clamp force, F_ax — Pulley clamp force**
5000 (default) | scalar

Pulley clamp force, $F_{ax}$, in N.

#### Reverse and Output Ratio

**Forward inertia, J_fwd — Inertia**
0.1 (default) | scalar

Forward inertia, $J_{fwd}$, in kg·m^2.

**Reverse inertia, J_rev — Inertia**
0.1 (default) | scalar

Reverse inertia, $J_{rev}$, in kg·m^2.

**Forward efficiency, eta_fwd — Efficiency**
0.95 (default) | scalar

Forward efficiency, $\eta_{fwd}$, dimensionless.

**Reverse efficiency, eta_rev — Efficiency**
0.95 (default) | scalar

Reverse efficiency, $\eta_{rev}$, dimensionless.

**Reverse gear ratio, N_rev — Ratio**
2 (default) | scalar

Reverse gear ratio, $N_{rev}$, dimensionless.

**Shift time constant, tau_s — Constant**
.01 (default) | scalar

Shift time constant, $\tau_s$, in s.

**Output gear ratio, N_o — Ratio**
2 (default) | scalar

Output gear ratio, $N_o$, dimensionless.

**Output gear efficiency, eta_o — Efficiency**
0.98 (default) | scalar

Output gear efficiency, $\eta_o$, dimensionless.

## References

[1] Ambekar, Ashok G. *Mechanism and Machine Theory*. New Delhi: Prentice-Hall of India, 2007.

[2] Bonsen, B. *Efficiency optimization of the push-belt CVT by variator slip control*. Ph.D. Thesis. Eindhoven University of Technology, 2006.

[3] *CVT How Does It Work*. CVT New Zealand 2010 Ltd, 10 Feb. 2011. Web. 25 Apr. 2016.

[4] Klaassen, T. W. G. L. *The Empact CVT: Dynamics and Control of an Electromechanically Actuated CVT*. Ph.D. Thesis. Eindhoven University of Technology, 2007.

[5] Sakagami, K. *Prediction of Friction Drive Limit of Metal V-Belt*. Warrendale, PA: SAE International Journal of Engines 8(3):1408-1416, 2015.

## Extended Capabilities

### C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

## See Also

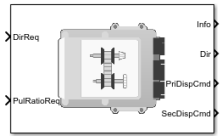
CVT Controller

**Introduced in R2017a**

# CVT Controller

Continuously variable transmission controller

**Library:** Powertrain Blockset / Transmission / Transmission Controllers



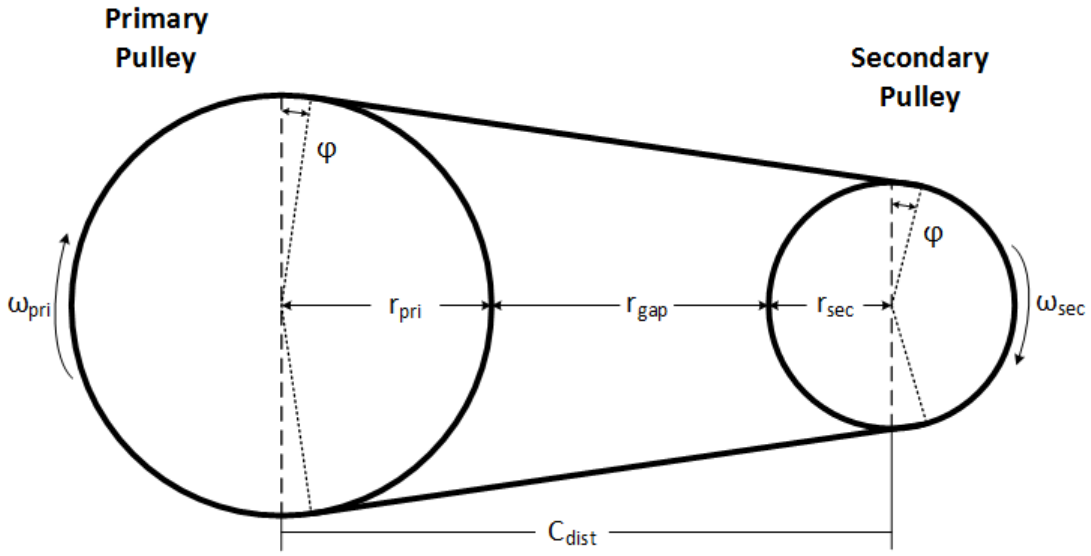

## Description

The CVT Controller block implements a push belt continuously variable transmission (CVT) controller. The block uses standard pulley and geometric equations to calculate the kinematic setpoints for the CVT variator. You can use the block to control a CVT.

### Pulley Kinematics

Using the physical dimensions of the system, the block calculates the primary and secondary variator positions that meet the pulley ratio request.

The figure and equations summarize the geometric dependencies.



$$C_{dist} = rp_{max} + r_{gap} + r_{sec_max}$$
$$L_0 = f(rp_{max}, rs_{max}, rp_{min}, rs_{min}, C_{dist})$$
$$ratio_{command} = f\left(ratio_{request}, ratio_{max}, ratio_{min}\right)$$
$$r_{pri} = f(r_0, ratio_{command}, C_{dist})$$
$$r_{\sec} = f(r_0, ratio_{command}, C_{dist})$$
$$x_{pri} = f\left(r_0, r_{pri}, \theta_{wedge}\right)$$
$$x_{\sec} = f\left(r_0, r_{\sec}, \theta_{wedge}\right)$$

The equations use these variables.

| | |
|---|---|
| $ratio_{request}$ | Pulley gear ratio request |
| $ratio_{command}$ | Pulley gear ratio command, based on request and physical limitations |
| $r_{gap}$ | Gap distance between variator pulleys |
| $C_{dist}$ | Distance between variator pulley centers |
| $rp_{max}$ | Maximum variator primary pulley radius |
| $rs_{max}$ | Maximum variator secondary pulley radius |
| $rp_{min}$ | Minimum variator primary pulley radius |
| $rs_{min}$ | Minimum variator secondary pulley radius |
| $r_o$ | Initial pulley radii with gear ratio of 1 |
| $L_o$ | Initial belt length, resulting from variator specification |
| $x_{pri}$ | Variator primary pulley displacement, resulting from controller request |
| $x_{sec}$ | Variator secondary pulley displacement, resulting from controller request |
| $r_{pri}$ | Variator primary pulley radius, resulting from controller request |
| $r_{sec}$ | Variator secondary pulley radius, resulting from controller request |
| $\Theta_{wedge}$ | Variator wedge angle |
| $\Phi$ | Angle of belt to pulley contact point |
| $L$ | Belt length, resulting from variator position |

## Ports

### Inputs

#### `DirReq` — Direction request

`scalar`

Direction request, $Dir_{req}$, controlling the direction, either forward or reverse. $Dir$ equals 1 for forward motion. $Dir$ equals -1 for reverse.

$$Dir = \begin{cases} 1 & \text{when } Dir_{req} \geq 0 \\ -1 & \text{when } Dir_{req} < 0 \end{cases}$$

#### `PllyRatioReq` — Pulley ratio request

`scalar`

CVT pulley ratio request, $ratio_{request}$.

### Output

#### `Info` — Bus signal

bus

Bus signal containing these block calculations.

| Signal | | Description | Variable | Units |
|---|---|---|---|---|
| `Radius` | `PriRadius` | Variator primary pulley radius, resulting from controller request | $r_{pri}$ | m |
| | `SecRadius` | Variator secondary pulley radius, resulting from controller request | $r_{sec}$ | m |
| | `InitPllyRadius` | Initial pulley radii with gear ratio of 1 | $r_o$ | m |
| `RatioAdj` | | Pulley gear ratio command, based on request and physical limitations | $ratio_{command}$ | N/A |
| `RatioMax` | | Maximum pulley ratio | $ratio_{max}$ | N/A |
| `RatioMin` | | Minimum pulley ratio | $ratio_{min}$ | N/A |
| `PriDispCmd` | | Variator primary pulley displacement, resulting from controller request | $x_{pri}$ | m |
| `SecDispCmd` | | Variator secondary pulley displacement, resulting from controller request | $x_{sec}$ | m |

#### `Dir` — Direction request

scalar

Direction request, $Dir_{req}$, controlling the direction, either forward or reverse. *Dir* equals 1 for forward motion. *Dir* equals -1 for reverse.

$$Dir = \begin{cases} 1 & \text{when } Dir_{req} \geq 0 \\ -1 & \text{when } Dir_{req} < 0 \end{cases}$$

#### `PriDispCmd` — Primary pulley displacement

scalar

Variator primary pulley displacement, $x_{pri}$, in m.

#### `SecDispCmd` — Secondary pulley displacement

scalar

Variator secondary pulley displacement, $x_{sec}$, in m.

## Parameters

### Kinematics

**`Maximum variator primary pulley radius, rp_max` — Radius**
`.08` (default) | `scalar`

Maximum variator primary pulley radius, $rp_{max}$, in m.

**`Maximum variator secondary pulley radius, rs_max` — Radius**
`.07` (default) | `scalar`

Maximum variator secondary pulley radius, $rs_{max}$, in m.

**`Minimum variator primary pulley radius, rp_min` — Radius**
`.03` (default) | `scalar`

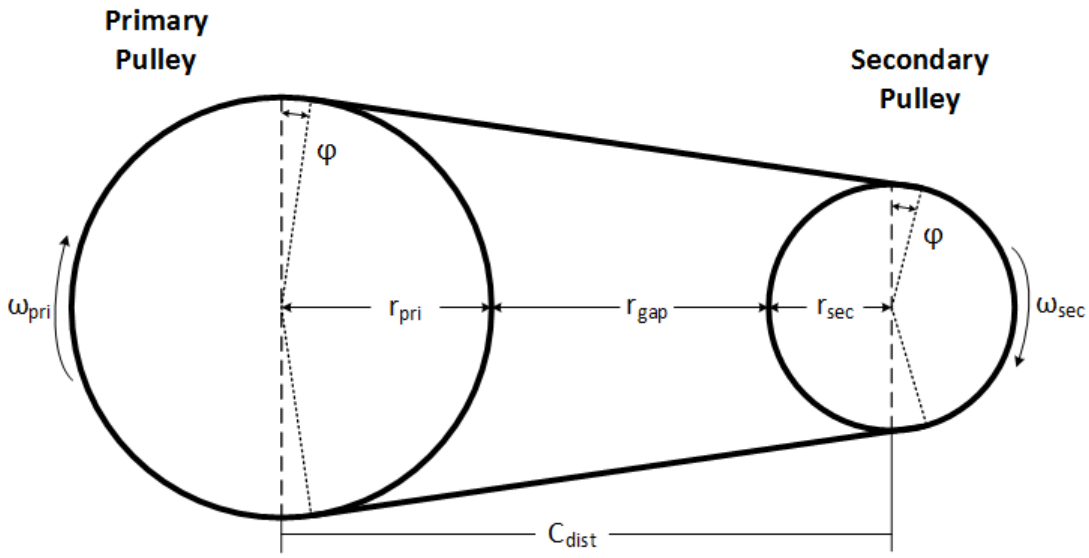
Minimum variator primary pulley radius, $rp_{min}$, in m.

**`Minimum variator secondary pulley radius, rs_min` — Radius**
`.03` (default) | `scalar`

Minimum variator secondary pulley radius, $rs_{min}$, in m.

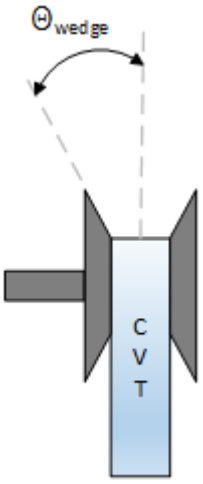
**`Gap distance between variator pulleys, rgap` — Specify crown wheel connection**
`.025` (default) | `scalar`

The gap between the secondary and primary pulleys, $r_{gap}$, in m. The figure shows the pulley geometry.




**`Variator wedge angle, thetawedge` — Specify crown wheel connection**
`11` (default) | `scalar`

Variator wedge angle, $\Theta_{wedge}$, in deg.




## References

[1] Ambekar, Ashok G. *Mechanism and Machine Theory*. New Delhi: Prentice-Hall of India, 2007.

[2] Bonsen, B. *Efficiency optimization of the push-belt CVT by variator slip control*. Ph.D. Thesis. Eindhoven University of Technology, 2006.

[3] *CVT How Does It Work*. CVT New Zealand 2010 Ltd. February 10, 2011. Accessed April 25, 2016.

[4] Klaassen, T. W. G. L. *The Empact CVT: Dynamics and Control of an Electromechanically Actuated CVT*. Ph.D. Thesis. Eindhoven University of Technology, 2007.

## Extended Capabilities

### C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

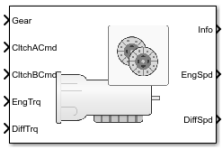
## See Also

Continuously Variable Transmission

**Introduced in R2017a**

# Dual Clutch Transmission

Dual clutch transmission that applies torque to the drive shaft
**Library:** Powertrain Blockset / Transmission / Transmission Systems




## Description

The Dual Clutch Transmission block implements a dual clutch transmission (DCT). In a DCT, two clutches apply mechanical torque to the drive shaft. Odd gears engage one clutch, while even gears engage the secondary clutch. The number of gears is specified via an integer vector with corresponding gear ratios, inertias, viscous damping, and efficiency factors. The clutch and synchronization engagement rates are linear and adjustable. You can provide external clutch signals or configure the block to generate idealized internal clutch signals. The block implements the transmission model with minimal parameterization or computational cost.

Use the block to model a simplified automated manual transmission (AMT) for:

- Power and torque capacity sizing
- Determining gear ratio impact on fuel economy and performance

To determine the rotational drive shaft speed and reaction torque, the Dual Clutch Transmission block calculates:

- Clutch lock-up and clutch friction
- Locked rotational dynamics
- Unlocked rotational dynamics

To specify the block efficiency calculation, for **Efficiency factors**, select either of these options.

| Setting | Block Implementation |
|---|---|
| `Gear only` | Efficiency determined from a 1D lookup table that is a function of the gear. |
| `Gear, input torque, input speed, and temperature` | Efficiency determined from a 4D lookup table that is a function of:<br>• Gear<br>• Input torque<br>• Input speed<br>• Oil temperature |

### Clutch Control

The DCT delivers drive shaft torque continuously by controlling the pressure signals from both clutches. If you select **Control mode** parameter `Ideal integrated controller`, the block generates idealized clutch pressure signals. The block uses the maximum pressure from each clutch

to approximate the single-clutch commands that result in equivalent drive shaft torque. To use your own clutch control signals, select **Control mode** parameter `External control`.

### Clutch Lock-Up and Clutch Friction

Based on the clutch lock-up condition, the block implements one of these friction models.

| If | Clutch Condition | Friction Model |
|---|---|---|
| $\omega_i \neq N\omega_d$<br>or<br>$T_S < \left\|T_f - Nw_ib_i\right\|$ | Unlocked | $T_f = T_k$<br>where,<br>$T_k = F_cR_{eff}\mu_k\tanh\left[4\left(\frac{w_i}{N} - w_d\right)\right]$<br>$T_s = F_cR_{eff}\mu_s$<br>$R_{eff} = \frac{2(R_o{}^3 - R_i{}^3)}{3(R_o{}^2 - R_i{}^2)}$ |
| $\omega_i = N\omega_t$<br>and<br>$T_S \geq \left\|T_f - Nb_i\omega_i\right\|$ | Locked | $T_f = T_s$ |

The equations use these variables.

| | |
|---|---|
| $\omega_t$ | Output drive shaft speed |
| $\omega_i$ | Input drive shaft speed |
| $\omega_d$ | Drive shaft speed |
| $b_i$ | Viscous damping |
| $F_c$ | Applied clutch force |
| $N$ | Engaged gear |
| $T_f$ | Frictional torque |
| $T_k$ | Kinetic frictional torque |
| $T_s$ | Static frictional torque |
| $R_{eff}$ | Effective clutch radius |
| $R_o$ | Annular disk outer radius |
| $R_i$ | Annular disk inner radius |
| $\mu_s$ | Coefficient of static friction |
| $\mu_k$ | Coefficient of kinetic friction |

### Locked Rotational Dynamics

To model the rotational dynamics when the clutch is locked, the block implements these equations.

$$\dot{\omega}_d J_N = \eta_N T_d - \frac{\omega_i}{N} b_N + NT_i$$

$$\omega_i = N\omega_d$$

The block determines the input torque, $T_i$, through differentiation.

The equations use these variables.

| | |
|---|---|
| $\omega_i$ | Input drive shaft speed |
| $\omega_d$ | Drive shaft speed |
| $N$ | Engaged gear |
| $b_N$ | Engaged gear viscous damping |
| $J_N$ | Engaged gear inertia |
| $\eta_N$ | Engaged gear efficiency |
| $T_d$ | Drive shaft torque |
| $T_i$ | Applied input torque |

### Unlocked Rotational Dynamics

To model the rotational dynamics when the clutch is unlocked, the block implements this equation.

$$\dot{\omega}_d J_N = N T_f - \omega_d b_N + T_d$$

where:

| | |
|---|---|
| $\omega_d$ | Drive shaft speed |
| $N$ | Engaged gear |
| $b_N$ | Engaged gear viscous damping |
| $J_N$ | Engaged gear inertia |
| $T_d$ | Drive shaft torque |
| $T_i$ | Applied input torque |

### Power Accounting

For the power accounting, the block implements these equations.

| Bus Signal | | | Description | Variable | Equations |
|---|---|---|---|---|---|
| `PwrInfo` | `PwrTrnsfrd` — Power transferred between blocks<br>• Positive signals indicate flow into block<br>• Negative signals indicate flow out of block | `PwrEng` | Engine power | $P_{eng}$ | $\omega_i T_i$ |
| | | `PwrDiffrntl` | Differential power | $P_{diff}$ | $\omega_d T_d$ |
| | `PwrNotTrnsfrd` — Power crossing the block boundary, but not transferred<br>• Positive signals indicate an input | `PwrEffLoss` | Mechanical power loss | $P_{effloss}$ | $\omega_d T_d(\eta_N - 1)$ |
| | | `PwrDampLoss` | Mechanical damping loss | $P_{damploss}$ | $-b_N\omega_d^2 - b_{in}\omega_i^2$ |

| Bus Signal | | | Description | Variable | Equations |
|---|---|---|---|---|---|
| | • Negative signals indicate a loss | `PwrCltchLoss` | Clutch power loss | $P_{mech}$ | When locked: 0<br><br>When unlocked: $-T_k(\omega_i - N\omega_d)$ |
| | `PwrStored` — Stored energy rate of change<br><br>• Positive signals indicate an increase<br>• Negative signals indicate a decrease | `PwrStoredTrans` | Rate change in rotational kinetic energy | $P_{str}$ | When locked: $\dot{\omega}_i\omega_i(J_{in} + \frac{J_N}{N^2})$<br><br>When unlocked: $J_{in}\dot{\omega}_i\omega_i + J_N\dot{\omega}_d\omega_d$ |

The equations use these variables.

| | |
|---|---|
| $b_N$ | Engaged gear viscous damping |
| $J_N$ | Engaged gear rotational inertia |
| $J_{in}$ | Flywheel rotational inertia |
| $\eta_N$ | Engaged gear efficiency |
| $N$ | Engaged gear ratio |
| $T_i$ | Applied input torque, typically from the engine crankshaft or dual mass flywheel damper |
| $T_d$ | Applied load torque, typically from the differential or drive shaft |
| $\omega_d$ | Initial input drive shaft rotational velocity |
| $\omega_i$, $\acute{\omega}_i$ | Applied drive shaft angular speed and acceleration |

## Ports

### Inputs

#### `Gear` — Gear number to engage

`scalar`

Integer value of gear number to engage.

#### `CltchACmd` — Command for odd-numbered gears

`scalar`

Clutch pressure command for odd-numbered gears, between `0` and `1`.

##### Dependencies

To create this port, select **Control mode** parameter `External control`.

#### `CltchBCmd` — Command for even-numbered gears

`scalar`

Clutch pressure command for even-numbered gears, between `0` and `1`.

#### Dependencies

To create this port, select **Control mode** parameter `External control`.

### `EngTrq` — Applied torque
`scalar`

Applied input torque, $T_i$, typically from the engine crankshaft or dual mass flywheel damper, in N·m.

### `DiffTrq` — Applied torque
`scalar`

Applied load torque, $T_d$, typically from the drive shaft, in N·m.

### `Temp` — Oil temperature
`scalar`

Oil temperature, in K. To determine the efficiency, the block uses a 4D lookup table that is a function of:

- Gear
- Input torque
- Input speed
- Oil temperature

#### Dependencies

To create this port, set **Efficiency factors** to `Gear, input torque, input speed, and temperature`.

### Output

### `Info` — Bus signal
bus

Bus signal containing these block calculations.

| Signal | | Description | Variable | Units |
|---|---|---|---|---|
| `Eng` | `EngTrq` | Applied input torque, typically from the engine crankshaft or dual mass flywheel damper | $T_i$ | N·m |
| | `EngSpd` | Applied drive shaft angular speed input | $\omega_i$ | rad/s |
| `Diff` | `DiffTrq` | Applied load torque, typically from the differential | $T_d$ | N·m |
| | `DiffSpd` | Drive shaft angular speed output | $\omega_d$ | rad/s |
| `Cltch` | `CltchForce` | Applied clutch force | $F_c$ | N |
| | `CltchLocked` | Clutch state | NA | NA |

| Signal | | | Description | Variable | Units |
|---|---|---|---|---|---|
| `Trans` | `TransSpd Ratio` | | Input to output speed ratio at time t | $\Phi(t)$ | NA |
| | `TransEta` | | Ratio of output power to input power | $\eta_N$ | NA |
| | `TransGearCmd` | | Commanded gear | $N_{cmd}$ | NA |
| | `TransGear` | | Engaged gear | $N$ | NA |
| `PwrInfo` | `PwrTrnsfrd` | `PwrEng` | Engine power | $P_{eng}$ | W |
| | | `PwrDiffrntl` | Differential power | $P_{diff}$ | W |
| | `PwrNotTrnsfrd` | `PwrEffLoss` | Mechanical power loss | $P_{effloss}$ | W |
| | | `PwrDampLoss` | Mechanical damping loss | $P_{damploss}$ | W |
| | | `PwrClutchLoss` | Clutch power loss | $P_{mech}$ | W |
| | `PwrStored` | `PwrStoredTrans` | Rate change in rotational kinetic energy | $P_{str}$ | W |

**`EngSpd` — Angular speed**
`scalar`

Drive shaft angular speed, $\omega_d$, in rad/s.

**`DiffSpd` — Angular speed**
`scalar`

Drive shaft angular speed, $\omega_d$, in rad/s.

## Parameters

**`Control mode` — Specify control mode**
`External control` (default) | `Ideal integrated controller`

The DCT delivers drive shaft torque continuously by controlling the pressure signals from both clutches. If you select **Control mode** parameter `Ideal integrated controller`, the block generates idealized clutch pressure signals. The block uses the maximum pressure from each clutch to approximate the single-clutch commands that result in equivalent drive shaft torque. To use your own clutch control signals, select **Control mode** parameter `External control`.

**Dependencies**

This table summarizes the port configurations.

| Control Mode | Creates Ports |
|---|---|
| `External control` | `CltchACmd` |
| | `CltchBCmd` |

**`Efficiency factors` — Specify efficiency calculation**
`Gear only` (default) | `Gear, input torque, input speed, and temperature`

To specify the block efficiency calculation, for **Efficiency factors**, select either of these options.

| Setting | Block Implementation |
|---|---|
| `Gear only` | Efficiency determined from a 1D lookup table that is a function of the gear. |
| `Gear, input torque, input speed, and temperature` | Efficiency determined from a 4D lookup table that is a function of:<br>• Gear<br>• Input torque<br>• Input speed<br>• Oil temperature |

#### Dependencies

| Setting Parameter To | Enables |
|---|---|
| `Gear only` | **Efficiency vector, eta** |
| `Gear, input torque, input speed, and temperature` | **Efficiency torque breakpoints, Trq_bpts**<br>**Efficiency speed breakpoints, omega_bpts**<br>**Efficiency temperature breakpoints, Temp_bpts**<br>**Efficiency lookup table, eta_tbl** |

#### Transmission

**`Input shaft inertia, Jin` — Inertia**
`0.1` (default) | `scalar`

Input shaft inertia, in kg·m^2.

**`Input shaft damping, bin` — Damping**
`0.001` (default) | `scalar`

Input shaft damping, in N·m·s/rad.

**`Initial input velocity, omegain_o` — Angular velocity**
`0` (default) | `scalar`

Angular velocity, in rad/s.

**`Efficiency torque breakpoints, Trq_bpts` — Breakpoints**
`[25 50 75 100 150 200 250]` (default) | `vector`

Torque breakpoints for efficiency table, in N·m.

**Dependencies**

To enable this parameter, set **Efficiency factors** to `Gear, input torque, input speed, and temperature`.

### `Efficiency speed breakpoints, omega_bpts` — Breakpoints

`[52.4 78.5 105 131 157 183 209 262 314 419 524]` (default) | `vector`

Speed breakpoints for efficiency table, in rad/s.

**Dependencies**

To enable this parameter, set **Efficiency factors** to `Gear, input torque, input speed, and temperature`.

### `Efficiency temperature breakpoints, Temp_bpts` — Breakpoints

`[313 358]` (default) | `vector`

Temperature breakpoints for efficiency table, in K.

**Dependencies**

To enable this parameter, set **Efficiency factors** to `Gear, input torque, input speed, and temperature`.

### `Gear number vector, G` — Specify number of transmission speeds

`[-1, 0, 1, 2, 3, 4, 5, 6, 7, 8]` (default) | `vector`

Vector of integers used to specify the number of transmission speeds. Neutral gear is `0`. For example, you can set these parameter values.

| To Specify | Set Gear number, G to |
|---|---|
| Four transmission speeds, including neutral | `[0,1,2,3,4]` |
| Three transmission speeds, including neutral and reverse | `[-1,0,1,2,3]` |
| Five transmission speeds, including neutral and reverse | `[-1,0,1,2,3,4,5]` |

Vector dimensions for the **Gear number vector**, **Gear ratio vector**, **Transmission inertia vector**, **Damping vector**, and **Efficiency vector** parameters must be equal.

### `Gear ratio vector, N` — Ratio of input speed to output speed

`[-4.70, 4.70, 4.700, 3.130,2.100, 1.670,1.290, 1.000,0.840, 0.670]` (default) | `vector`

Vector of gear ratios (that is, input speed to output speed) with indices corresponding to the ratios specified in **Gear number, G**. For neutral, set the gear ratio to `1`. For example, you can set these parameter values.

| To Specify Gear Ratios for | Set Gear number, G to | Set Gear ratio, N to |
|---|---|---|
| Four transmission speeds, including neutral | `[0,1,2,3,4]` | `[1,4.47,2.47,1.47,1]` |

| To Specify Gear Ratios for | Set Gear number, G to | Set Gear ratio, N to |
|---|---|---|
| Five transmission speeds, including neutral and reverse | `[-1,0,1,2,3,4,5]` | `[-4.47,1,4.47,2.47,1.47,1,0.8]` |

Vector dimensions for the **Gear number vector**, **Gear ratio vector**, **Transmission inertia vector**, **Damping vector**, and **Efficiency vector** parameters must be equal.

**`Transmission inertia vector, Jout` — Gear rotational inertia**
`[0.08 0.08 0.08 0.04 0.02 0.01 0.01 0.01 0.01 0.01]` (default) | `vector`

Vector of gear rotational inertias, with indices corresponding to the inertias specified in **Gear number, G**, in kg·m^2. For example, you can set these parameter values.

| To Specify Inertia for | Set Gear number, G to | Set Inertia, J to |
|---|---|---|
| Four gears, including neutral | `[0,1,2,3,4]` | `[0.01,2.28,2.04,0.32,0.028]` |
| Inertia for five gears, including reverse and neutral | `[-1,0,1,2,3,4,5]` | `[2.28,0.01,2.28,2.04,0.32,0.028,0.01]` |

Vector dimensions for the **Gear number vector**, **Gear ratio vector**, **Transmission inertia vector**, **Damping vector**, and **Efficiency vector** parameters must be equal.

**`Damping vector, bout` — Gear viscous damping coefficient**
`[.003 .001 .003 .0025 .002 .001 .001 .001 .001 .001]` (default) | `vector`

Vector of gear viscous damping coefficients, with indices corresponding to the coefficients specified in **Gear number, G**, in N·m·s/rad. For example, you can set these parameter values.

| To Specify Damping for | Set Gear number, G to | Set Damping, b to |
|---|---|---|
| Four gears, including neutral | `[0,1,2,3,4]` | `[0.001,0.003,0.0025, 0.002,0.001]` |
| Five gears, including reverse and neutral | `[-1,0,1,2,3,4,5]` | `[0.003,0.001,0.003,0.0025, 0.002,0.001,0.001]` |

Vector dimensions for the **Gear number vector**, **Gear ratio vector**, **Transmission inertia vector**, **Damping vector**, and **Efficiency vector** parameters must be equal.

**`Efficiency vector, eta` — Gear efficiency**
`[0.930, 0.930, 0.930, 0.940,0.947, 0.948,0.946, 0.943,0.940, 0.935]` (default) | `vector`

Vector of gear mechanical efficiency, with indices corresponding to the efficiencies specified in **Gear number, G**. For example, you can set these parameter values.

| To Specify Efficiency for | Set Gear number, G to | Set Efficiency, eta to |
|---|---|---|
| Four gears, including neutral | `[0,1,2,3,4]` | `[0.9,0.9,0.9,0.9,0.95]` |
| Five gears, including reverse and neutral | `[-1,0,1,2,3,4,5]` | `[0.9,0.9,0.9, 0.9,0.9,0.95,0.95]` |

Vector dimensions for the **Gear number vector**, **Gear ratio vector**, **Transmission inertia vector**, **Damping vector**, and **Efficiency vector** parameters must be equal.

#### Dependencies

To enable this parameter, set **Efficiency factors** to `Gear only`.

### `Efficiency lookup table, eta_tbl` — Gear efficiency

`array`

Table of gear mechanical efficiency, $\eta_N$ as a function of gear, input torque, input speed, and temperature.

#### Dependencies

To enable this parameter, set **Efficiency factors** to `Gear, input torque, input speed, and temperature`.

### `Initial output velocity, omegaout_o` — Transmission

`0` (default) | `scalar`

Transmission initial output rotational velocity, $\omega_{to}$, in rad/s. If you select **Clutch initially locked**, the block ignores the **Initial output velocity, omega_o** parameter value.

### `Initial gear, G_o` — Engaged gear

`0` (default) | `scalar`

Initial gear to engage, $G_o$.

#### Clutch and Synchronizer

### `Clutch pressure time constant, tauc` — Time

`.02` (default) | `scalar`

Time required to engage and disengage the clutch during shift events, $t_c$, in s.

### `Synchronization time, ts` — Time

`.2` (default) | `scalar`

Time required for gear selection and synchronization, $t_s$, in s.

### `Clutch time, tc` — Time

`.5` (default) | `scalar`

Time required to engage clutch, $t_c$, in s.

#### Dependencies

To create this parameter, select **Control mode** parameter `Ideal integrated controller`.

### `Effective clutch radius, R` — Radius

`.25` (default) | `scalar`

The effective radius, $R_{eff}$, used with the applied clutch friction force to determine the friction force, in m. The effective radius is defined as:

$$R_{eff} = \frac{2(R_o{}^3 - R_i{}^3)}{3(R_o{}^2 - R_i{}^2)}$$

The equation uses these variables.

| | |
|---|---|
| $R_o$ | Annular disk outer radius |
| $R_i$ | Annular disk inner radius |

**`Clutch force gain, K_c` — Force**
`5e4` (default) | `scalar`

Open loop lock-up clutch gain, $K_c$, in N.

**`Clutch static friction coefficient, mus` — Coefficient**
`0.3` (default) | `scalar`

Dimensionless clutch disc coefficient of static friction, $\mu_s$.

**`Clutch kinematic friction coefficient, muk` — Coefficient**
`0.25` (default) | `scalar`

Dimensionless clutch disc coefficient of kinetic friction, $\mu_k$.

**`Clutch initially locked` — Select to initially lock clutch**
`off` (default) | `on`

Selecting this parameter initially locks the clutch.

#### Dependencies

To create this parameter, select **Control mode** parameter `Ideal integrated controller`.

**`Synchronizer initially locked` — Select to initially lock synchronizer**
`off` (default) | `on`

Selecting this parameter initially locks the synchronizer.

## Extended Capabilities

#### C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

## See Also

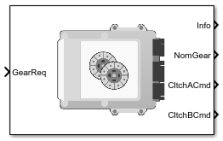
Automated Manual Transmission | DCT Controller

#### Introduced in R2017a

# DCT Controller

Dual clutch transmission controller

**Library:** Powertrain Blockset / Transmission / Transmission Controllers




## Description

The DCT Controller block implements a dual clutch transmission (DCT) controller. You can specify the clutch open, close, and synchronization timing parameters. The block determines the clutch commands using integrator-based timers and latching logic that is based on the specified timing parameters and gear request.

## Ports

### Inputs

#### `GearReq` — Gear number to engage

`scalar`

Gear number request, $G_{req}$.

### Output

#### `Info` — Bus signal

bus

Bus signal containing these block calculations.

| Signal | Description | Variable |
|---|---|---|
| `GearReq` | Gear number request | $G_{req}$ |
| `GearEngd` | Nominal gear commanded by the controller | $G_o$ |
| `GearEffct` | Effective gear | NA |
| `CltchACmd` | Clutch pressure command for odd-numbered gears, between `0` and `1` | NA |
| `CltchBCmd` | Clutch pressure command for even-numbered gears, between `0` and `1` | NA |

#### `NomGear` — Nominal gear for shifting

`scalar`

Nominal gear for shifting. The Dual Clutch Transmission block uses this signal for the smooth application of inertial, efficiency, gear ratio, and damping parameters.

**`CltchACmd` — Command for odd-numbered gears**
`scalar`

Clutch pressure command for odd-numbered gears, between `0` and `1`.

**`CltchBCmd` — Command for even-numbered gears**
`scalar`

Clutch pressure command for even-numbered gears, between `0` and `1`.

## Parameters

**`Initial gear, G_o` — Engaged gear**
`0` (default) | `scalar`

Initial gear to engage, $G_o$.

**`Clutch actuation time, tc` — Time**
`.1` (default) | `scalar`

Time required to engage and disengage the clutch during shift events, $t_c$, in s.

**`Synchronizer time, ts` — Time**
`.01` (default) | `scalar`

Time required for gear selection and synchronization, $t_s$, in s.

**`Sample period, dt` — Time**
`-1` (default) | `scalar`

Sample period, *dt*, in s.

**`Clutch initially locked` — Select to initially lock clutch**
`off` (default) | `on`

Selecting this parameter initially locks the clutch.

**`Synchronizer initially locked` — Select to initially lock synchronizer**
`off` (default) | `on`

Selecting this parameter initially locks the synchronizer.

## Extended Capabilities

**C/C++ Code Generation**
Generate C and C++ code using Simulink® Coder™.

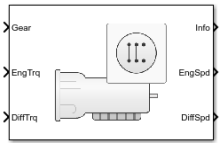
## See Also
AMT Controller | Dual Clutch Transmission

**Introduced in R2017a**

# Ideal Fixed Gear Transmission

Ideal fixed gear transmission without clutch or synchronization

**Library:** Powertrain Blockset / Transmission / Transmission Systems
Vehicle Dynamics Blockset / Powertrain / Transmission




## Description

The Ideal Fixed Gear Transmission implements an idealized fixed-gear transmission without a clutch or synchronization. Use the block to model the overall gear ratio and power loss when you do not need a detailed transmission model, for example, in component-sizing, fuel economy, and emission studies. The block implements a transmission model with minimal parameterization or computational cost.

To specify the block efficiency calculation, for **Efficiency factors**, select either of these options.

| Setting | Block Implementation |
|---|---|
| `Gear only` | Efficiency determined from a 1D lookup table that is a function of the gear. |
| `Gear, input torque, input speed, and temperature` | Efficiency determined from a 4D lookup table that is a function of:<br>• Gear<br>• Input torque<br>• Input speed<br>• Oil temperature |

The block uses this equation to determine the transmission dynamics:

$$\dot{\omega}_i \frac{J_N}{N^2} = \eta_N \left( \frac{T_o}{N} + T_i \right) - \frac{\omega_i}{N^2} b_N$$

$$\omega_i = N \omega_o$$

The block filters the gear command signal:

$$\frac{G}{G_{cmd}}(s) = \frac{1}{\tau_s s + 1}$$

### Neutral Gear

When **Initial gear number, G_o** is equal to `0`, the initial gear is neutral. The block uses these parameters to decouple the input flywheel from the downstream gearing.

- **Initial input velocity, omega_o**
- **Initial neutral input velocity, omegainN_o**

The block uses these equations for the neutral gear speed and flywheel.

$$\dot{\omega}_{neutral}\frac{J_N}{N^2} = \eta_N\frac{T_o}{N} - \frac{\omega_{neutral}}{N^2}b_N$$

$$\omega_{neutral} = N\omega_o$$

$$\dot{\omega}_1 J_F = \eta_{@N=0}T_i - b_{@N=0}\omega_i$$

$$J_F = J_{@N=1} - J_{@N=0}$$

### Power Accounting

For the power accounting, the block implements these equations.

| Bus Signal | | | Description | Variable | Equations |
|---|---|---|---|---|---|
| `PwrInfo` | `PwrTrnsfrd` — Power transferred between blocks<br>• Positive signals indicate flow into block<br>• Negative signals indicate flow out of block | `PwrEng` | Engine power | $P_{eng}$ | $\omega_i T_i$ |
| | | `PwrDiffrntl` | Differential power | $P_{diff}$ | $\omega_o T_o$ |
| | `PwrNotTrnsfrd` — Power crossing the block boundary, but not transferred<br>• Positive signals indicate an input<br>• Negative signals indicate a loss | `PwrEffLoss` | Mechanical power loss | $P_{effloss}$ | $\omega_o T_o(\eta_N - 1)$ |
| | | `PwrDamplOss` | Mechanical damping loss | $P_{damploss}$ | For G=0: $-\frac{b_N\omega_i^2}{\lvert N^2\rvert}$<br>For G ≠ 0: $-b_N\omega_i^2 - \frac{b_N\omega_{neutral}^2}{\lvert N^2\rvert}$ |
| | `PwrStored` — Stored energy rate of change<br>• Positive signals indicate an increase<br>• Negative signals indicate a decrease | `PwrStoredTrns` | Rate change in rotational kinetic energy | $P_{str}$ | For G=0: $\frac{J_N}{N^2}\dot{\omega}_i\omega_i$<br>For G ≠ 0: $J_F\dot{\omega}_i\omega_i + \frac{J_N}{N^2}\dot{\omega}_{neutral}\omega_{neutr}$ |

The equations use these variables.

| | |
|---|---|
| $b_N$ | Engaged gear viscous damping |
| $J_N$ | Engaged gear rotational inertia |
| $J_F$ | Flywheel rotational inertia |
| $\eta_N$ | Engaged gear efficiency |
| $G$ | Engaged gear number |
| $G_{cmd}$ | Gear number to engage |
| $N$ | Engaged gear ratio |

| | |
|---|---|
| $T_i$ | Applied input torque, typically from the engine crankshaft or dual mass flywheel damper |
| $T_o$ | Applied load torque, typically from the differential or drive shaft |
| $\omega_o$ | Initial input drive shaft rotational velocity |
| $\omega_i$, $\acute{\omega}_i$ | Applied drive shaft angular speed and acceleration |
| $\omega_{No}$ | Initial neutral gear input rotational velocity |
| $\omega_{neutral}$ | Neutral gear drive shaft rotational velocity |
| $\tau_s$ | Shift time constant |

## Ports

### Inputs

#### `Gear` — Gear number to engage

`scalar`

Integer value of gear number to engage, $G_{cmd}$.

#### `EngTrq` — Applied input torque

`scalar`

Applied input torque, $T_i$, typically from the engine crankshaft or dual mass flywheel damper, in N·m.

#### `DiffTrq` — Applied load torque

`scalar`

Applied load torque, $T_o$, typically from the differential, in N·m.

#### `Temp` — Oil temperature

`scalar`

Oil temperature, in K. To determine the efficiency, the block uses a 4D lookup table that is a function of:

- Gear
- Input torque
- Input speed
- Oil temperature

##### Dependencies

To enable this port, set **Efficiency factors** to `Gear, input torque, input speed, and temperature`.

### Output

#### `Info` — Bus signal

bus

Bus signal containing these block calculations.

| Signal | | | Description | Variable | Units |
|---|---|---|---|---|---|
| `Eng` | `EngTrq` | | Applied input torque, typically from the engine crankshaft or dual mass flywheel damper | $T_i$ | N·m |
| | `EngSpd` | | Applied drive shaft angular speed input | $\omega_i$ | rad/s |
| `Diff` | `DiffTrq` | | Applied load torque, typically from the differential | $T_o$ | N·m |
| | `DiffSpd` | | Drive shaft angular speed output | $\omega_o$ | rad/s |
| `Trans` | `TransSpdRatio` | | Input to output speed ratio at time t | $\Phi(t)$ | N/A |
| | `TransEta` | | Ratio of output power to input power | $\eta_N$ | N/A |
| | `TransGearCmd` | | Commanded gear | $N_{cmd}$ | N/A |
| | `TransGear` | | Engaged gear | $N$ | N/A |
| `PwrInfo` | `PwrTrnsfrd` | `PwrEng` | Engine power | $P_{eng}$ | W |
| | | `PwrDiffrntl` | Differential power | $P_{diff}$ | W |
| | `PwrNotTrnsfrd` | `PwrEffLoss` | Mechanical power loss | $P_{effloss}$ | W |
| | | `PwrDampLoss` | Mechanical damping loss | $P_{damploss}$ | W |
| | `PwrStored` | `PwrStoredTrans` | Rate change in rotational kinetic energy | $P_{str}$ | W |

**`EngSpd` — Angular speed**
`scalar`

Applied drive shaft angular speed input, $\omega_i$, in rad/s.

**`DiffSpd` — Angular speed**
`scalar`

Drive shaft angular speed output, $\omega_o$, in rad/s.

## Parameters

### `Efficiency factors` — Specify efficiency calculation

`Gear only` (default) | `Gear, input torque, input speed, and temperature`

To specify the block efficiency calculation, for **Efficiency factors**, select either of these options.

| Setting | Block Implementation |
|---|---|
| `Gear only` | Efficiency determined from a 1D lookup table that is a function of the gear. |
| `Gear, input torque, input speed, and temperature` | Efficiency determined from a 4D lookup table that is a function of:<br>• Gear<br>• Input torque<br>• Input speed<br>• Oil temperature |

#### Dependencies

| Setting Parameter To | Enables |
|---|---|
| `Gear only` | **Efficiency vector, eta** |
| `Gear, input torque, input speed, and temperature` | **Efficiency torque breakpoints, Trq_bpts**<br>**Efficiency speed breakpoints, omega_bpts**<br>**Efficiency temperature breakpoints, Temp_bpts**<br>**Efficiency lookup table, eta_tbl** |

### `Gear property interpolation method` — Interpolation

`Nearest` (default) | `Linear` | `Flat` | `Cubic spline`

Method that the block uses to switch the gear ratio during gear shifting.

#### Transmission

### `Gear number vector, G` — Specify number of transmission speeds

`[-1,0,1,2,3,4,5]` (default) | `vector`

Vector of integer gear commands used to specify the number of transmission speeds. Neutral gear is `0`. For example, you can set these parameter values.

| To Specify | Set Gear number, G To |
|---|---|
| Four transmission speeds, including neutral | `[0,1,2,3,4]` |
| Three transmission speeds, including neutral and reverse | `[-1,0,1,2,3]` |
| Five transmission speeds, including neutral and reverse | `[-1,0,1,2,3,4,5]` |

Vector dimensions for the **Gear number vector**, **Gear ratio vector**, **Inertia vector**, **Damping vector**, and **Efficiency vector** parameters must be equal.

**`Efficiency torque breakpoints, Trq_bpts` — Breakpoints**
`[25,50,75,100,150,200,250]` (default) | `vector`

Torque breakpoints for efficiency table.

**Dependencies**

To enable this parameter, set **Efficiency factors** to `Gear, input torque, input speed, and temperature`.

**`Efficiency speed breakpoints, omega_bpts` — Breakpoints**
`[52.4 78.5 105 131 157 183 209 262 314 419 524]` (default) | `vector`

Speed breakpoints for efficiency table.

**Dependencies**

To enable this parameter, set **Efficiency factors** to `Gear, input torque, input speed, and temperature`.

**`Efficiency temperature breakpoints, Temp_bpts` — Breakpoints**
`[313 358]` (default) | `vector`

Temperature breakpoints for efficiency table.

**Dependencies**

To enable this parameter, set **Efficiency factors** to `Gear, input torque, input speed, and temperature`.

**`Gear ratio vector, N` — Ratio of input speed to output speed**
`[-4.47,4.47,4.47,2.47,1.47,1,0.8]` (default) | `vector`

Vector of gear ratios (that is, input speed to output speed) with indices corresponding to the ratios specified in **Gear number, G**. For neutral, set the gear ratio to 1. For example, you can set these parameter values.

| To Specify Gear Ratios For | Set Gear number, G To | Set Gear ratio, N To |
|---|---|---|
| Four transmission speeds, including neutral | `[0,1,2,3,4]` | `[1,4.47,2.47,1.47,1]` |
| Five transmission speeds, including neutral and reverse | `[-1,0,1,2,3,4,5]` | `[-4.47,1,4.47,2.47, 1.47,1,0.8]` |

Vector dimensions for the **Gear number vector**, **Gear ratio vector**, **Inertia vector**, **Damping vector**, and **Efficiency vector** parameters must be equal.

**`Inertia vector, Jout` — Gear rotational inertia**
`[0.128 0.01 0.128 0.1 0.062 0.028 0.01]` (default) | `vector`

Vector of gear rotational inertias, $J_N$, with indices corresponding to the inertias specified in **Gear number, G**, in kg*m^2. For example, you can set these parameter values.

| To Specify Inertia For | Set Gear number, G To | Set Inertia, J To |
|---|---|---|
| Four gears, including neutral | `[0,1,2,3,4]` | `[0.01,2.28,2.04, 0.32,0.028]` |
| Inertia for five gears, including reverse and neutral | `[-1,0,1,2,3,4,5]` | `[2.28,0.01,2.28, 2.04,0.32,0.028,0.01]` |

Vector dimensions for the **Gear number vector**, **Gear ratio vector**, **Inertia vector**, **Damping vector**, and **Efficiency vector** parameters must be equal.

### `Damping vector, bout` — Gear viscous damping coefficient

`[.003 .001 .003 .0025 .002 .001 .001]` (default) | `vector`

Vector of gear viscous damping coefficients, $b_N$, with indices corresponding to the coefficients specified in **Gear number, G**, in N·m·s/rad. For example, you can set these parameter values.

| To Specify Damping For | Set Gear number, G To | Set Damping, b To |
|---|---|---|
| Four gears, including neutral | `[0,1,2,3,4]` | `[0.001,0.003, 0.0025,0.002,0.001]` |
| Five gears, including reverse and neutral | `[-1,0,1,2,3,4,5]` | `[0.003,0.001, 0.003,0.0025, 0.002,0.001,0.001]` |

Vector dimensions for the **Gear number vector**, **Gear ratio vector**, **Inertia vector**, **Damping vector**, and **Efficiency vector** parameters must be equal.

### `Efficiency vector, eta` — Gear efficiency

`[0.9,0.9,0.9,0.9,0.9,0.95,0.95]` (default) | `vector`

Vector of gear mechanical efficiency, $\eta_N$, with indices corresponding to the efficiencies specified in **Gear number, G**. For example, you can set these parameter values.

| To Specify Efficiency For | Set Gear number, G To | Set Efficiency, eta To |
|---|---|---|
| Four gears, including neutral | `[0,1,2,3,4]` | `[0.9,0.9,0.9,0.9,0.95]` |
| Five gears, including reverse and neutral | `[-1,0,1,2,3,4,5]` | `[0.9,0.9,0.9, 0.9,0.9,0.95,0.95]` |

Vector dimensions for the **Gear number vector**, **Gear ratio vector**, **Inertia vector**, **Damping vector**, and **Efficiency vector** parameters must be equal.

#### Dependencies

To enable this parameter, set **Efficiency factors** to `Gear only`.

### `Efficiency lookup table, eta_tbl` — Gear efficiency

`array`

Table of gear mechanical efficiency, $\eta_N$ as a function of gear, input torque, input speed, and temperature.

#### Dependencies

To enable this parameter, set **Efficiency factors** to `Gear, input torque, input speed, and temperature`.

**`Initial gear number, G_o` — Gear**
`0` (default) | `scalar`

Initial gear number, $G_o$, dimensionless.

**`Initial output velocity, omega_o` — Output speed**
`0` (default) | `scalar`

Transmission initial output rotational velocity, $\omega_o$, in rad/s.

**`Initial neutral input velocity, omegainN_o` — Neutral gear input speed**
`0` (default) | `scalar`

Initial neutral gear input rotational velocity, $\omega_{No}$, in rad/s.

**`Shift time constant, tau_s` — Time**
`.01` (default) | `scalar`

Shift time constant, $\tau_s$, in s.

## Extended Capabilities

**C/C++ Code Generation**
Generate C and C++ code using Simulink® Coder™.

## See Also

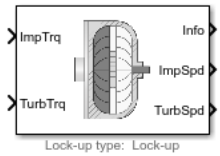
Automated Manual Transmission | Continuously Variable Transmission | Dual Clutch Transmission

**Introduced in R2017a**

# Torque Converter

Three-part torque converter consisting of an impeller, turbine, and stator

**Library:** Powertrain Blockset / Transmission / Torque Converters




## Description

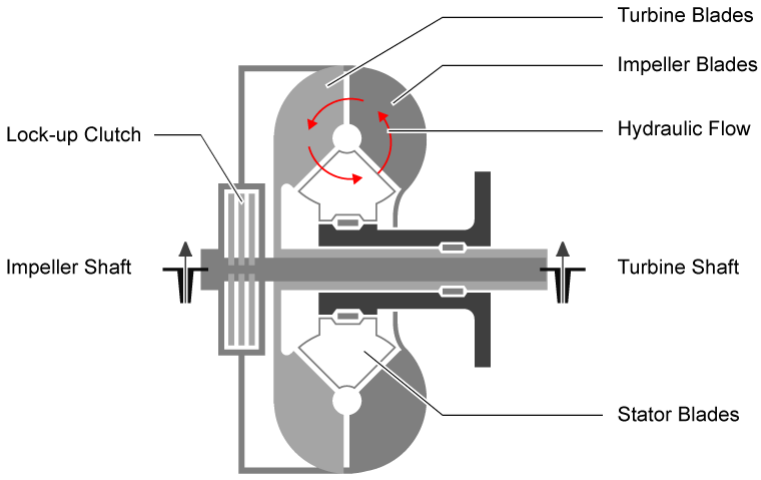
The Torque Converter block implements a three-part torque converter consisting of an impeller, turbine, and stator with an optional clutch lock-up capability. The block can simulate driving (power flowing from impeller to turbine) and coasting (power from turbine dissipated in torque converter hydraulic fluid).

You can specify torque converter characteristics:

- Speed ratio — Ratio of turbine angular speed to impeller angular speed
- Torque ratio — Ratio of turbine torque to impeller torque
- Capacity factor parameterization — Function of input speed or input torque

Optional clutch lock-up configurations include:

- No lock-up — Model fluid-coupling only
- Lock-up — Model automatic clutch engagement
- External lock-up — Model clutch pressure as input from an external signal




### Dynamics

#### Clutch Lock-Up Condition and Clutch Friction

Based on the clutch lock-up condition, the block implements these friction models.

| If | Clutch Condition | Friction Model |
|---|---|---|
| $\omega_i \neq \omega_t$<br>or<br>$T_S < \left\lvert \frac{J_t}{(J_i + J_t)}[T_i + T_f - \omega_i(b_t + b_i)] \right\rvert$ | Unlocked | $T_f = T_k$<br>where:<br>$T_k = F_c R_{eff} m_k \tanh[4(\omega_i - \omega_t)]$<br>$T_s = F_c R_{eff} m_s$<br>$R_{eff} = \frac{2({R_o}^3 - {R_i}^3)}{3({R_o}^2 - {R_i}^2)}$ |
| $\omega_i = \omega_t$<br>and<br>$T_S \geq \left\lvert \frac{J_t}{(J_i + J_t)}[T_i + T_f - w_t(b_t + b_i) + w_t b_t] \right\rvert$ | Locked | $T_f = T_s$ |

#### Locked Rotational Dynamics

To model the rotational dynamics if the clutch is locked, the block implements equations.

$$\dot{\omega}(J_i + J_t) = T_i - \omega(b_i + b_t) + T_{ext}$$
$$\omega = \omega_i = \omega_t$$

The rotational velocity represents both the impeller and turbine rotational velocities.

#### Unlocked Rotational Dynamics

To model the rotational dynamics if the clutch is unlocked, the block implements equations.

$$\dot{\omega}_i J_i = \mathrm{T}_i - \omega_i b_i - T_f - T_p$$
$$\dot{\omega}_t J_t = \mathrm{T}_{ext} - \omega_t b_t + T_f + T_t$$
$$T_p = \omega_i{}^2 \psi(\phi)$$
$$T_t = T_p \zeta(\phi)$$

To approximate the torque multiplication lag between the impeller and turbine, you can specify the parameter **Fluid torque response time constant (set to 0 to disable), tauc [s]**.

### Power Accounting

For the power accounting, the block implements these equations.

| Bus Signal | | | Description | Variable | Equations |
|---|---|---|---|---|---|
| `PwrInfo` | `PwrTrnsfrd` — Power transferred between blocks<br>• Positive signals indicate flow into block<br>• Negative signals indicate flow out of block | `PwrImp` | Applied impeller power | $P_{imp}$ | $\omega_i T_i$ |
| | | `PwrTurb` | Applied turbine output power | $P_{turb}$ | $\omega_t T_t$ |

| Bus Signal | | | Description | Variable | Equations |
|---|---|---|---|---|---|
| | `PwrNotTrnsfrd` — Power crossing the block boundary, but not transferred<br>• Positive signals indicate an input<br>• Negative signals indicate a loss | `PwrDampLoss` | Mechanical damping loss | $P_{damploss}$ | $-b_t\omega_t^2 - b_i\omega_i^2$ |
| | | `PwrFluidCplingLoss` | Heat loss to transmission fluid | $P_{flloss}$ | $-\left(T_p\omega_i - T_{hyd}\omega_t\right)$ |
| | | `PwrCltchLoss` | Clutch slip power loss | $P_{cltloss}$ | $-T_k(\omega_i - \omega_t)$ |
| | `PwrStored` — Stored energy rate of change<br>• Positive signals indicate an increase<br>• Negative signals indicate a decrease | `PwrStoredImp` | Rate change in impeller rotational kinetic energy | $P_{strimp}$ | $\dot{\omega}_i\omega_i J_i$ |
| | | `PwrStoredTurb` | Rate change in turbine rotational kinetic energy | $P_{strturb}$ | $\dot{\omega}_t\omega_t J_t$ |

The block implements equations that use these variables.

| | |
|---|---|
| $T_f$ | Frictional torque |
| $T_k$ | Kinetic frictional torque |
| $T_s$ | Static frictional torque |
| $T_i$ | Applied input torque |
| $T_p$ | Impeller reaction torque |
| $T_{ext}$ | Externally applied turbine torque |
| $\psi(\phi)$ | Torque conversion capacity factor |
| $\zeta(\phi)$ | Torque ratio |
| $\omega_i$ | Impeller rotational shaft speed |
| $\omega_t$ | Turbine rotational shaft speed |
| $J_i$ | Impeller rotational inertia |
| $J_t$ | Turbine rotational inertia |
| $b_i$ | Impeller rotational viscous damping |
| $b_t$ | Turbine rotational viscous damping |
| $R_{eff}$ | Effective clutch radius |
| $R_o$ | Annular disk outer radius |
| $R_i$ | Annular disk inner radius |

## Ports

### Inputs

#### `ImpTrq` — Applied impeller torque

`scalar`

Applied input torque, typically from the engine crankshaft or dual mass flywheel, in N·m.

### `TurbTrq` — Applied turbine torque

`scalar`

Applied turbine torque, typically from the transmission, in N·m.

### `Clutch Force` — Applied clutch force

`scalar`

Applied clutch force, typically from a hydraulic actuator, in N.

#### Dependencies

To create this port, select `External lock-up input` for the **Lock-up clutch configuration** parameter.

## Output

### `Info` — Bus signal

bus

Bus signal containing these block calculations.

| Signal | | | Description | Units |
|---|---|---|---|---|
| `Imp` | `ImpTrq` | | Applied input torque | N·m |
| | `ImpSpd` | | Impeller rotational shaft speed | rad/s |
| `Turb` | `TurbTrq` | | Applied turbine torque | N·m |
| | `TurbSpd` | | Turbine rotational shaft speed | rad/s |
| `Cltch` | `CltchForce` | | Applied clutch force | N |
| | `CltchLocked` | | Clutch locked or unlocked state | N/A |
| `TrqConv` | `TrqConvSpdRatio` | | Turbine to impeller speed ratio | N/A |
| | `TrqConvEta` | | Torque conversion efficiency | N/A |
| `PwrInfo` | `PwrTrnsfrd` | `PwrImp` | Applied impeller power | W |
| | | `PwrTurb` | Applied turbine output power | W |
| | `PwrNotTrnsfrd` | `PwrDampLoss` | Mechanical damping loss | W |
| | | `PwrFluidCplingLoss` | Heat loss to transmission fluid | W |
| | | `PwrCltchLoss` | Clutch slip power loss | W |
| | `PwrStored` | `PwrStoredImp` | Rate change in impeller rotational kinetic energy | W |
| | | `PwrStoredTurb` | Rate change in turbine rotational kinetic energy | W |

### `ImpSpd` — Impeller speed

`scalar`

Impeller rotational shaft speed, $\omega_i$, in rad/s.

**`TrbSpd` — Turbine speed**
`scalar`

Turbine rotational shaft speed, $\omega_t$, in rad/s.

## Parameters

#### Configuration

**`Lock-up clutch configuration` — Select lock-up clutch configuration**
`Lock-up` (default) | `No lock-up` | `External lock-up input`

| To Model | Select |
|---|---|
| Fluid-coupling only | `No lock-up` |
| Automatic clutch engagement | `Lock-up` |
| Clutch pressure as input from an external signal | `External lock-up input` |

#### Dependencies

To enable the **Clutch** parameters, select `Lock-up` or `External lock-up input` for the **Lock-up clutch configuration** parameter.

#### Torque Converter

**`Impeller shaft inertia, Ji` — Inertia**
`.1` (default) | `scalar`

Impeller shaft inertia, in kg·m^2.

**`Impeller shaft viscous damping, bi` — Viscous damping coefficient**
`.001` (default) | `scalar`

Impeller shaft viscous damping, in N·m·s/rad.

**`Turbine shaft inertia, Jt` — Inertia**
`.1` (default) | `scalar`

Turbine shaft inertia, in kg·m^2.

**`Turbine shaft viscous damping, bt` — Viscous damping coefficient**
`.001` (default) | `scalar`

Turbine shaft viscous damping, in N·m·s/rad.

**`Initial impeller shaft velocity, omegai_o` — Angular velocity**
`0` (default) | `scalar`

Initial impeller shaft velocity, in rad/s.

**`Initial turbine shaft velocity, omegat_o` — Angular velocity**
`0` (default) | `scalar`

Initial turbine shaft velocity, in rad/s.

**`Speed ratio vector, phi` — Ratio**
`[ 0 0.50 0.60 0.70 0.80 0.87 0.92 0.94 0.96 0.97]` (default) | `vector`

Vector of turbine speed to impeller speed ratios. Breakpoints for the capacity and torque multiplication vectors.

**`Capacity factor parameterization` — Select factor ratio type**
`Input speed / sqrt(input torque)` (default) | `Absorbed torque / input speed^2`

| To Set Factor Ratio to | Select |
|---|---|
| Impeller angular velocity to square root impeller torque | `Input speed / sqrt(input torque)` |
| Impeller absorbed torque to square of impeller angular velocity | `Absorbed torque / input speed^2` |

**`Capacity vector, psi` — Vector**
`[12.2938 12.8588 13.1452 13.6285 14.6163 16.2675 19.3503 22.1046 29.9986 50.00]` (default) | `vector`

| Capacity factor parameterization Setting | Capacity Vector Units |
|---|---|
| `Input speed / sqrt(input torque)` | (rad/s)/(N·m)^0.5 |
| `Absorbed torque / input speed^2` | N·m/(rad/s)^2 |

**`Torque ratio vector, zeta` — Vector**
`[2.2320 1.5462 1.4058 1.2746 1.1528 1.0732 1.0192 0.9983 0.9983 0.9983]` (default) | `vector`

Vector of turbine torque to impeller speed ratios.

**`Fluid torque response time constant (set to 0 to disable), tauTC` — Time constant**
`.02` (default) | `scalar`

To account for the delay in torque calculations due to changing input torque, specify the fluid torque transfer time constant, in s.

**`Interpolation method` — Select interpolation method**
`Linear` (default) | `Flat` | `Nearest`

Interpolates the torque ratio and capacity factor functions between the discrete relative velocity values.

#### Clutch

**`Clutch force equivalent net radius, Reff` — Effective radius**
`.3` (default) | `scalar`

The effective radius, $R_{eff}$, used with the applied clutch friction force to determine the friction force, in m. The effective radius is defined as:

$$R_{eff} = \frac{2(R_o 3 - R_i 3)}{3(R_o 2 - R_i 2)}$$

The equation uses these variables.

| | |
|---|---|
| $R_o$ | Annular disk outer radius |
| $R_i$ | Annular disk inner radius |

#### Dependencies

To enable the **Clutch** parameters, select `Lock-up` or `External lock-up input` for the **Lock-up clutch configuration** parameter.

### `Static friction coefficient, mus` — Coefficient

`1.2` (default) | `scalar`

Dimensionless clutch disc coefficient of static friction.

#### Dependencies

To enable the **Clutch** parameters, select `Lock-up` or `External lock-up input` for the **Lock-up clutch configuration** parameter.

### `Kinetic friction coefficient, muk` — Coefficient

`1` (default) | `scalar`

Dimensionless clutch disc coefficient of kinetic friction.

To enable the **Clutch** parameters, select `Lock-up` or `External lock-up input` for the **Lock-up clutch configuration** parameter.

### `Initially lock clutch` — Select to initially lock clutch

`off` (default) | `on`

#### Dependencies

To enable this parameter, select `Lock-up` or `External lock-up input` for the **Lock-up clutch configuration** parameter.

### `Lock-up speed ratio threshold, philu` — Threshold

`.85` (default) | `scalar`

Set speed ratio threshold that engages clutch lock-up.

#### Dependencies

To enable this parameter, select `Lock-up` for the **Lock-up clutch configuration** parameter.

### `Minimum lock-up engagement speed, omegal` — Angular velocity

`900*pi/30` (default) | `scalar`

Set the minimum impeller speed that engages clutch lock-up, in rad/s.

#### Dependencies

To enable this parameter, select `Lock-up` for the **Lock-up clutch configuration** parameter.

### `Lock-up disengagement speed, omegau` — Angular velocity

`800*pi/30` (default) | `scalar`

Set the minimum impeller speed that disengages clutch lock-up, in rad/s.

**Dependencies**

To enable this parameter, select `Lock-up` for the **Lock-up clutch configuration** parameter.

**`Lock-up clutch force gain, K_c` — Gain**
`5000` (default) | `scalar`

Open loop clutch lock-up force gain, in N.

**Dependencies**

To enable this parameter, select `Lock-up` for the **Lock-up clutch configuration** parameter.

**`Lock-up clutch time constant, tauC` — Time constant**
`.0500` (default) | `scalar`

Open loop clutch lock-up time constant, in s.

**Dependencies**

To enable this parameter, select `Lock-up` for the **Lock-up clutch configuration** parameter.

## Extended Capabilities

### C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

## See Also

CI Core Engine | SI Core Engine

**Introduced in R2017a**

# Functions

# mdf

Access information contained in MDF-file

## Syntax

```
mdfObj = mdf(mdfFileName)
```

## Description

The `mdf` function creates an object for accessing a measurement data format (MDF) file. See “Measurement Data Format (MDF)” on page 8-4.

`mdfObj = mdf(mdfFileName)` identifies a measurement data format (MDF) file and returns an MDF-file object, which you can use to access information and data contained in the file. You can specify a full or partial path to the file.

## Examples

### Create MDF-File Object for Specified MDF-File

Create an MDF object for a given file, and view the object display.

```
mdfObj = mdf('MDFFile.mf4')
```

```
MDF with properties:

   File Details
                  Name: 'MDFFile.mf4'
                  Path: 'c:\temp\MDFFile.mf4'
                Author: 'HOK'
            Department: 'Research'
               Project: 'MDF'
               Subject: 'CAN bus'
               Comment: 'This file contains CAN messages'
               Version: '4.10'
              DataSize: 32100
     InitialTimestamp: 2016-02-27 12:09:02

   Creator Details
    ProgramIdentifier: 'mmddff.04'
              Creator: [1×1 struct]

   File Contents
            Attachment: [1×1 struct]
```

```
    ChannelNames: {6×1 cell}
    ChannelGroup: [1×6 struct]
```

## Input Arguments

### `mdfFileName` — MDF-file name

char vector | string

MDF-file name, specified as a character vector or string, including the necessary full or relative path.

Example: `'MDFFile.mf4'`

Data Types: `char` | `string`

## Output Arguments

### `mdfObj` — MDF-file

MDF-file object

MDF-file, returned as an MDF-file object. The object provides access to the MDF-file information contained in the following properties.

| Property | Description |
|---|---|
| `Name` | Name of the MDF-file, including extension |
| `Path` | Full path to the MDF-file, including file name |
| `Author` | Author who originated the MDF-file |
| `Department` | Department that originated the MDF-file |
| `Project` | Project that originated the MDF-file |
| `Subject` | Subject matter in the MDF-file |
| `Comment` | Open comment field from the MDF-file |
| `Version` | MDF standard version of the file |
| `DataSize` | Total size of the data in the MDF-file, in bytes |
| `InitialTimestamp` | Time when file data acquisition began in UTC or local time |
| `ProgramIdentifier` | Originating program of the MDF-file |
| `Creator` | Structure containing details about creator of the MDF-file, with these fields: `VendorName`, `ToolName`, `ToolVersion`, `UserName`, and `Comment` |
| `Attachment` | Structure of information about attachments contained within the MDF-file, with these fields: `Name`, `Path`, `Comment`, `Type`, `MIMEType`, `Size`, `EmbeddedSize`, and `MD5CheckSum` |
| `ChannelNames` | Cell array of the channel names in each channel group |
| `ChannelGroup` | Structure of information about channel groups contained within the MDF-file, with these fields: `AcquisitionName`, `Comment`, `NumSamples`, `DataSize`, `Sorted`, and `Channel` |

## More About

### Measurement Data Format (MDF)

Measurement data format (MDF) files are binary format files for storing measurement data. The format standard is defined by the Association for Standardization of Automation and Measuring Systems (ASAM), which you can read about at ASAM MDF.

Vehicle Network Toolbox™ and Powertrain Blockset provide access to MDF-files through an object you create with the `mdf` function.

## See Also

### Functions

`read` | `saveAttachment`

### Introduced in R2016b

# read

Read channel data from MDF-file

## Syntax

```
data = read(mdfObj)
data = read(mdfObj,chanList)
data = read(mdfObj,chanGroupIndex,chanName)
data = read(mdfObj,chanGroupIndex,chanName,startPosition)
data = read(mdfObj,chanGroupIndex,chanName,startPosition,endPosition)
data = read(mdfObj,chanGroupIndex,chanName,startPosition,
endPosition,'OutputFormat',fmtType)
[data,time] = read(mdfObj,chanGroupIndex,chanName,startPosition,
endPosition,'OutputFormat','Vector')
```

## Description

`data = read(mdfObj)` reads all data for all channels from the MDF-file identified by the MDF-file object `mdfObj`, and assigns the output to `data`. If the file data is one channel group, the output is a timetable; multiple channel groups are returned as a cell array of timetables, where the cell array index corresponds to the channel group number.

`data = read(mdfObj,chanList)` reads all data for all channels specified in the channel list table `chanList`.

`data = read(mdfObj,chanGroupIndex,chanName)` reads all data for the specified channel from the MDF-file identified by the MDF-file object `mdfObj`.

`data = read(mdfObj,chanGroupIndex,chanName,startPosition)` reads data from the position specified by `startPosition`.

`data = read(mdfObj,chanGroupIndex,chanName,startPosition,endPosition)` reads data for the range specified from `startPosition` to `endPosition`.

`data = read(mdfObj,chanGroupIndex,chanName,startPosition,
endPosition,'OutputFormat',fmtType)` returns data with the specified output format.

`[data,time] = read(mdfObj,chanGroupIndex,chanName,startPosition,
endPosition,'OutputFormat','Vector')` returns two vectors of channel data and corresponding timestamps.

## Examples

### Read All Data from MDF-File

Read all available data from the MDF-file.

```
mdfObj = mdf('MDFFile.mf4');
data = read(mdfObj);
```

### Read All Data from Specified Channel List

Read all available data from the MDF-file for channels specified as part of a channel list.

```
mdfObj = mdf('MDFFile.mf4');
chanList = channelList(mdfObj) % Channel table
data = read(mdfObj,chanList(1:3,:)); % First 3 channels
```

### Read All Data from Multiple Channels

Read all available data from the MDF-file for specified channels.

```
mdfObj = mdf('MDFFile.mf4');
data = read(mdfObj,1,{'Channel1','Channel2'});
```

### Read Range of Data from Specified Index Values

Read a range of data from the MDF-file using indexing for `startPosition` and `endPosition` to specify the data range.

```
mdfObj = mdf('MDFFile.mf4');
data = read(mdfObj,1,{'Channel1','Channel2'},1,10);
```

### Read Range of Data from Specified Time Values

Read a range of data from the MDF-file using time values for `startPosition` and `endPosition` to specify the data range.

```
mdfObj = mdf('MDFFile.mf4');
data = read(mdfObj,1,{'Channel1','Channel2'},seconds(5.5),seconds(7.3));
```

### Read All Data in Vector Format

Read all available data from the MDF-file, returning data and time vectors.

```
mdfObj = mdf('MDFFile.mf4');
[data,time] = read(mdfObj,1,'Channel1','OutputFormat','Vector');
```

### Read All Data in Time Series Format

Read all available data from the MDF-file, returning time series data.

```
mdfObj = mdf('MDFFile.mf4');
data = read(mdfObj,1,'Channel1','OutputFormat','TimeSeries');
```

### Read Data from Channel List Entry

Read data from a channel identified by the `channelList` function.

Get list of channels and display their names and group numbers.

```
mdfObj = mdf('File05.mf4');
chlist = channelList(mdfObj);
chlist(1:2,1:2) % Display 2 channels, 2 columns
```

```
  2×2 table

              ChannelName                ChannelGroupNumber
    _____________________________________    __________________

    "Float_32_LE_Offset_64"                         2
    "Float_64_LE_Master_Offset_0"                   2
```

Read data from the first channel in the list.

```
data = read(mdfObj,chlist{1,2},chlist{1,1});
data(1:5,:)
```

```
  5×1 timetable

      Time       Float_32_LE_Offset_64
    _________    _____________________

    0 sec                    5
    0.01 sec               5.1
    0.02 sec               5.2
    0.03 sec               5.3
    0.04 sec               5.4
```

## Input Arguments

### `mdfObj` — MDF-file

MDF-file object

MDF-file, specified as an MDF-file object.

Example: `mdf('MDFFile.mf4')`

### `chanList` — List of channels

table

List of channels, specified as a table in the format returned by the `channelList` function.

Example: `channelList()`

Data Types: `table`

### `chanGroupIndex` — Index of the channel group

numeric value

Index of channel group, specified as a numeric value that identifies the channel group from which to read.

Example: 1

Data Types: `single` | `double` | `int8` | `int16` | `int32` | `int64` | `uint8` | `uint16` | `uint32` | `uint64`

**`chanName` — Name of channel**
char vector | string

Name of channel, specified as a character vector, string, or array. `chanName` identifies the name of a channel in the channel group. Use a cell array of character vectors or array of string to identify multiple channels.

Example: `'Channel1'`

Data Types: `char` | `string` | `cell`

**`startPosition` — First position of channel data**
numeric value | duration

First position of channel data, specified as a numeric value or duration. The `startPosition` option specifies the first position from which to read channel data. Provide a numeric value to specify an index position; use a duration to specify a time position. If only `startPosition` is provided without the `endPosition` option, the data value at that location is returned. When used with `endPosition` to specify a range, the function returns data from the `startPosition` (inclusive) to the `endPosition` (noninclusive).

Example: 1

Data Types: `single` | `double` | `int8` | `int16` | `int32` | `int64` | `uint8` | `uint16` | `uint32` | `uint64` | `duration`

**`endPosition` — Last position of channel data range**
numeric value | duration

Last position of channel data range, specified as a numeric value or duration. The `endPosition` option specifies the last position for reading a range of channel data. Provide both the `startPosition` and `endPosition` to specify retrieval of a range of data. The function returns up to but not including `endPosition` when reading a range. Provide a numeric value to specify an index position; use a duration to specify a time position.

Example: 1000

Data Types: `single` | `double` | `int8` | `int16` | `int32` | `int64` | `uint8` | `uint16` | `uint32` | `uint64` | `duration`

**`fmtType` — Format for output data**
`'Timetable'` (default) | `'Vector'` | `'TimeSeries'`

Format for output data, specified as a character vector or string. This option formats the output according to the following table.

| OutputFormat | Description |
|---|---|
| `'Timetable'` | Return a timetable from one or more channels into one output variable. This is the only format allowed when reading from multiple channels at the same time. (Default.)<br><br>Note: The timetable format includes columns for the MDF channels. Because the column titles must be valid MATLAB identifiers, they might not be exactly the same as those values in the MDF object `ChannelNames` property. The column headers are derived from the property using the function `matlab.lang.makeValidName`. The original channel names are available in the `VariableDescriptions` property of the `timetable` object. |
| `'Vector'` | Return a vector of numeric data values, and optionally a vector of time values from one channel. Use one output variable to return only data, or two output variables to return both data and time vectors. |
| `'TimeSeries'` | Return a time series of data from one channel. |

Example: `'Vector'`

Data Types: `char` | `string`

## Output Arguments

### `data` — Channel data

timetable (default) | double | time series | cell array

Channel data, returned as vector of doubles, a time series, a timetable, or cell array of timetables, according to the `'OutputFormat'` option setting and the number of channel groups.

### `time` — Channel data times

double

Channel data times, returned as a vector of double elements. The time vector is returned only when the `'OutputFormat'` is set to `'Vector'`.

## See Also

**Functions**

`channelList` | `mdf` | `saveAttachment`

**Topics**

“Time Series”
“Represent Dates and Times in MATLAB”
“Tables”

**Introduced in R2016b**

# saveAttachment

Save attachment from MDF-file

## Syntax

```
saveAttachment(mdfObj,AttachmentName)
saveAttachment(mdfObj,AttachmentName,DestFile)
```

## Description

`saveAttachment(mdfObj,AttachmentName)` saves the specified attachment from the MDF-file to the current MATLAB working folder. The attachment is saved with its existing name.

`saveAttachment(mdfObj,AttachmentName,DestFile)` saves the specified attachment from the MDF-file to the given destination. You can specify relative or absolute paths to place the attachment in a specific folder.

## Examples

### Save Attachment with Original Name

Save an MDF-file attachment with its original name in the current folder.

```
mdfObj = mdf('MDFFile.mf4');
saveAttachment(mdfObj,'AttachmentName.ext')
```

### Save Attachment with New Name

Save an MDF-file attachment with a new name in the current folder.

```
mdfObj = mdf('MDFFile.mf4');
saveAttachment(mdfObj,'AttachmentName.ext','MyFile.ext')
```

### Save Attachment in Parent Folder

Save an MDF-file attachment in a folder specified with a relative path name, in this case in the parent of the current folder.

```
mdfObj = mdf('MDFFile.mf4');
saveAttachment(mdfObj,'AttachmentName.ext','..\MyFile.ext')
```

### Save Attachment in Specified Folder

This example saves an MDF-file attachment using an absolute path name.

```
mdfObj = mdf('MDFFile.mf4');
saveAttachment(mdfObj,'AttachmentName.ext','C:\MyDir\MyFile.ext')
```

## Input Arguments

### `mdfObj` — MDF-file

MDF-file object

MDF-file, specified as an MDF-file object.

Example: `mdf('MDFFile.mf4')`

### `AttachmentName` — MDF-file attachment name

char vector | string

MDF-file attachment name, specified as a character vector or string. The name of the attachment is available in the `Name` field of the MDF-file object `Attachment` property.

Example: `'file1.dbc'`

Data Types: `char` | `string`

### `DestFile` — Destination file name for the saved attachment

existing attachment name (default) | char vector | string

Destination file name for the saved attachment, specified as a character vector or string. The specified destination can include an absolute or relative path, otherwise the attachment is saved in the current folder.

Example: `'MyFile.ext'`

Data Types: `char` | `string`

## See Also

### Functions

`mdf` | `read`

### Introduced in R2016b

# mdfDatastore

Datastore for collection of MDF-files

## Description

Use the MDF datastore object to access data from a collection of MDF-files.

## Creation

### Syntax

```
mdfds = mdfDatastore(location)
mdfds = mdfDatastore(__,'Name1',Value1,'Name2',Value2,...)
```

#### Description

`mdfds = mdfDatastore(location)` creates an MDFDatastore based on an MDF-file or a collection of files in the folder specified by `location`. All files in the folder with extensions `.mdf`, `.dat`, or `.mf4` are included.

`mdfds = mdfDatastore(__,'Name1',Value1,'Name2',Value2,...)` specifies function options and properties of `mdfds` using optional name-value pairs.

**Note** This function is supported only on 64-bit Windows® operating systems.

#### Input Arguments

**`location` — Location of MDF datastore files**
character vector | cell array | `DsFileSet` object

Location of MDF datastore files, specified as a character vector, cell array of character vectors, or `matlab.io.datastore.DsFileSet` object identifying either files or folders. The path can be relative or absolute, and can contain the wildcard character *. If `location` specifies a folder, by default the datastore includes all files in that folder with the extensions `.mdf`, `.dat`, or `.mf4`.

Example: `'CANape.MF4'`

Data Types: `char` | `cell` | `DsFileSet`

#### Name-Value Pair Arguments

Specify optional comma-separated pairs of Name,Value arguments to set file information or object “Properties” on page 8-13. Allowed options are `IncludeSubfolders`, `FileExtensions`, and the properties `ReadSize`, `SelectedChannelGroupNumber`, and `SelectedChannelNames`.

Example: `'SelectedChannelNames','Counter_B4'`

**`IncludeSubfolders` — Include files in subfolders**
`false` (default) | `true`

Include files in subfolders, specified as a logical. Specify `true` to include files in each folder and recursively in subfolders.

Example: `'IncludeSubfolders',true`

Data Types: `logical`

**`FileExtensions` — Custom extensions for filenames to include in MDF datastore**
`{'.mdf','.dat','.mf4'}` (default) | char | cell

Custom extensions for filenames to include in the MDF datastore, specified as a character vector or cell array of character vectors. By default, the supported extensions include `.mdf`, `.dat`, and `.mf4`. If your files have custom or nonstandard extensions, use this Name-Value setting to include files with those extensions.

Example: `'FileExtensions',{'.myformat1','.myformat2'}`

Data Types: `char` | `cell`

## Properties

**`ChannelGroups` — All channel groups present in first MDF-file (read-only)**
table

All channel groups present in first MDF-file, returned as a table.

Data Types: `table`

**`Channels` — All channels present in first MDF-file (read-only)**
table

All channels present in first MDF-file, returned as a table.

Those channels targeted for reading must have the same name and belong to the same channel group in each file of the MDF datastore.

Data Types: `table`

**`Files` — Files included in datastore**
char | string | cell

Files included in the datastore, specified as a character vector, string, or cell array.

Example: `{'file1.mf4','file2.mf4'}`

Data Types: `char` | `string` | `cell`

**`ReadSize` — Size of data returned by `read`**
`'file'` (default) | numeric | duration

Size of data returned by the `read` function, specified as `'file'`, a numeric value, or a duration. A character vector value of `'file'` causes the entire file to be read; a numeric double value specifies the number of records to read; and a duration value specifies a time range to read.

If you later change the `ReadSize` property value type, the datastore resets.

Example: `50`

Data Types: `double` | `char` | `duration`

**`SelectedChannelGroupNumber` — Channel group to read**
numeric scalar

Channel group to read, specified as a numeric scalar value.

Example: `1`

Data Types: `single` | `double` | `int8` | `int16` | `int32` | `int64` | `uint8` | `uint16` | `uint32` | `uint64`

**`SelectedChannelNames` — Names of channels to read**
char | string | cell

Names of channels to read, specified as a character vector, string, or cell array.

Those channels targeted for reading must have the same name and belong to the same channel group in each file of the MDF datastore.

Example: `'Counter_B4'`

Data Types: `char` | `string` | `cell`

## Object Functions

| | |
|---|---|
| read | Read data in MDF datastore |
| readall | Read all data in MDF datastore |
| preview | Subset of data from MDF datastore |
| reset | Reset MDF datastore to initial state |
| hasdata | Determine if data is available to read from MDF datastore |
| partition | Partition MDF datastore |
| numpartitions | Number of partitions for MDF datastore |
| combine (MATLAB) | Combine data from multiple datastores |
| transform (MATLAB) | Transform datastore |
| isPartitionable (MATLAB) | Determine whether datastore is partitionable |
| isShuffleable (MATLAB) | Determine whether datastore is shuffleable |

## Examples

### Create an MDF Datastore

Create an MDF datastore from the sample file `CANape.MF4`, and read it into a timetable.

```
mdfds = mdfDatastore(fullfile(matlabroot,'examples','vnt','CANape.MF4'));
while hasdata(mdfds)
    m = read(mdfds);
end
```

## See Also

### Introduced in R2017b

# hasdata (MDFDatastore)

Determine if data is available to read from MDF datastore

## Syntax

```
tf = hasdata(mdfds)
```

## Description

`tf = hasdata(mdfds)` returns logical `1` (`true`) if there is data available to read from the MDF datastore specified by `mdfds`. Otherwise, it returns logical `0` (`false`).

## Examples

### Check MDF Datastore for Readable Data

Use `hasdata` in a loop to control read iterations.

```
mdfds = mdfDatastore(fullfile(matlabroot,'examples','vnt','CANape.MF4'));
while hasdata(mdfds)
    m = read(mdfds);
end
```

## Input Arguments

### `mdfds` — MDF datastore

MDF datastore object

MDF datastore, specified as an MDF datastore object.

Example: `mdfds = mdfDatastore('CANape.MF4')`

## Output Arguments

### `tf` — Indicator of data to read

`1` | `0`

Indicator of data to read, returned as a logical `1` (`true`) or `0` (`false`).

## See Also

### Functions

`mdfDatastore` | `read` | `readall` | `reset`

### Introduced in R2017b

# numpartitions (MDFDatastore)

Number of partitions for MDF datastore

## Syntax

```
N = numpartitions(mdfds)
N = numpartitions(mdfds,pool)
```

## Description

`N = numpartitions(mdfds)` returns the recommended number of partitions for the MDF datastore `mdfds`. Use the result as an input to the `partition` function.

`N = numpartitions(mdfds,pool)` returns a reasonable number of partitions to parallelize `mdfds` over the parallel pool, `pool`, based on the number of files in the datastore and the number of workers in the pool.

## Examples

### Find Recommended Number of Partitions for MDF Datastore

Determine the number of partitions you should use for your MDF datastore.

```
mdfds = mdfDatastore(fullfile(matlabroot,'examples','vnt','CANape.MF4'));
N = numpartitions(mdfds);
```

## Input Arguments

### `mdfds` — MDF datastore

MDF datastore object

MDF datastore, specified as an MDF datastore object.

Example: `mdfds = mdfDatastore('CANape.MF4')`

### `pool` — Parallel pool

parallel pool object

Parallel pool specified as a parallel pool object.

Example: `gcp`

## Output Arguments

### `N` — Number of partitions

double

Number of partitions, returned as a double. This number is the calculated recommendation for the number of partitions for your MDF datastore. Use this when partitioning your datastore with the `partition` function.

## See Also

**Functions**
`mdfDatastore` | `partition` | `read` | `reset`

**Introduced in R2017b**

# partition (MDFDatastore)

Partition MDF datastore

## Syntax

```
subds = partition(mdfds,N,index)

subds = partition(mdfds,'Files',index)
subds = partition(mdfds,'Files',filename)
```

## Description

`subds = partition(mdfds,N,index)` partitions the MDF datastore `mdfds` into the number of parts specified by `N`, and returns the partition corresponding to the index `index`.

`subds = partition(mdfds,'Files',index)` partitions the MDF datastore by files and returns the partition corresponding to the file of index `index` in the `Files` property.

`subds = partition(mdfds,'Files',filename)` partitions the datastore by files and returns the partition corresponding to the specified `filename`.

## Examples

### Partition an MDF Datastore into Default Parts

Partition an MDF datastore from the sample file `CANape.MF4`, and return the first part.

```
mdfds = mdfDatastore(fullfile(matlabroot,'examples','vnt','CANape.MF4'));
N = numpartitions(mdfds);
subds1 = partition(mdfds,N,1);
```

### Partition an MDF Datastore by Its Files

Partition an MDF datastore according to its files, and return partitions by index and file name.

```
cd c:\temp
mdfds = mdfDatastore({'CANape1.MF4','CANape2.MF4','CANape3.MF4'});
mdfds.Files

ans =
  3×1 cell array
    'c:\temp\CANape1.MF4'
    'c:\temp\CANape2.MF4'
    'c:\temp\CANape3.MF4'
```

```
subds2 = partition(mdfds,'files',2);
subds3 = partition(mdfds,'files','c:\temp\CANape3.MF4');
```

## Input Arguments

### `mdfds` — MDF datastore

MDF datastore object

MDF datastore, specified as an MDF datastore object.

Example: `mdfds = mdfDatastore('CANape.MF4')`

### `N` — Number of partitions

positive integer

Number of partitions, specified as a double of positive integer value. Use the `numpartitions` function for the recommended number or partitions.

Example: `numpartitions(mdfds)`

Data Types: `double`

### `index` — Index

positive integer

Index, specified as a double of positive integer value. When using the `'files'` partition scheme, this value corresponds to the index of the MDF datastore object `Files` property.

Example: `1`

Data Types: `double`

### `filename` — File name

character vector

File name, specified as a character vector. The argument can specify a relative or absolute path.

Example: `'CANape.MF4'`

Data Types: `char`

## Output Arguments

### `subds` — MDF datastore partition

MDF datastore object

MDF datastore partition, returned as an MDF datastore object. This output datastore is of the same type as the input datastore `mdfds`.

## See Also

### Functions

`mdfDatastore` | `numpartitions` | `read` | `reset`

### Introduced in R2017b

# preview (MDFDatastore)

Subset of data from MDF datastore

## Syntax

```
data = preview(mdfds)
```

## Description

`data = preview(mdfds)` returns a subset of data from MDF datastore `mdfds` without changing the current position in the datastore.

## Examples

### Examine Preview of MDF Datastore

```
mdfds = mdfDatastore(fullfile(matlabroot,'examples','vnt','CANape.MF4'));
data = preview(mdfds)
```

```
data2 =

  10×74 timetable

        Time         Counter_B4    Counter_B5    Counter_B6    Counter_B7    PWM
    _____________    __________    __________    __________    __________    ___

    0.00082554 sec    0             0             1             0             100
      0.010826 sec    0             0             1             0             100
      0.020826 sec    0             0             1             0             100
      0.030826 sec    0             0             1             0             100
      0.040826 sec    0             0             1             0             100
      0.050826 sec    0             0             1             0             100
      0.060826 sec    0             0             1             0             100
      0.070826 sec    0             0             1             0             100
```

## Input Arguments

### `mdfds` — MDF datastore

MDF datastore object

MDF datastore, specified as an MDF datastore object.

Example: `mdfds = mdfDatastore('CANape.MF4')`

## Output Arguments

### `data` — Subset of data

timetable

Subset of data, returned as a timetable of MDF records.

## See Also

**Functions**
`hasdata` | `mdfDatastore` | `read`

**Introduced in R2017b**

# read (MDFDatastore)

Read data in MDF datastore

## Syntax

```
data = read(mdfds)
[data,info] = read(mdfds)
```

## Description

`data = read(mdfds)` returns data from the MDF datastore `mdfds` into the timetable `data`.

The `read` function returns a subset of data from the datastore. The size of the subset is determined by the `ReadSize` property of the datastore object. On the first call, `read` starts reading from the beginning of the datastore, and subsequent calls continue reading from the endpoint of the previous call. Use `reset` to read from the beginning again.

`[data,info] = read(mdfds)` also returns to the output argument `info` information, including metadata, about the extracted data.

## Examples

### Read Datastore by Files

Read data from an MDF datastore one file at a time.

```
mdfds = mdfDatastore({'CANape1.MF4','CANape2.MF4','CANape3.MF4'});
mdfds.ReadSize = 'file';
data = read(mdfds);
```

Read the second file and view information about the data.

```
[data2,info2] = read(mdfds);
info2
```

```
  struct with fields:

            Filename: 'CANape2.MF4'
            FileSize: 57592
    MDFFileProperties: [1×1 struct]
```

## Input Arguments

### `mdfds` — MDF datastore

MDF datastore object

MDF datastore, specified as an MDF datastore object.

Example: `mdfds = mdfDatastore('CANape.MF4')`

## Output Arguments

### `data` — Output data

timetable

Output data, returned as a timetable of MDF records.

### `info` — Information about data

structure array

Information about data, returned as a structure array with the following fields:

```
Filename
FileSize
MDFFileProperties
```

## See Also

### Functions

`hasdata` | `mdfDatastore` | `preview` | `readall` | `reset`

### Introduced in R2017b

# readall (MDFDatastore)

Read all data in MDF datastore

## Syntax

```
data = readall(mdfds)
```

## Description

`data = readall(mdfds)` reads all the data in the datastore specified by `mdfds` and returns it to timetable `data`.

After the `readall` function returns all the data, it resets `mdfds` to point to the beginning of the datastore.

If all the data in the datastore does not fit in memory, then `readall` returns an error.

## Examples

### Read All Data in Datastore

Read all the data from a multiple file MDF datastore into a timetable.

```
mdfds = mdfDatastore({'CANape1.MF4','CANape2.MF4','CANape3.MF4'});
data = readall(mdfds);
```

## Input Arguments

### `mdfds` — MDF datastore

MDF datastore object

MDF datastore, specified as an MDF datastore object.

Example: `mdfds = mdfDatastore('CANape.MF4')`

## Output Arguments

### `data` — Output data

timetable

Output data, returned as a timetable of MDF records.

## See Also

**Functions**

`hasdata` | `mdfDatastore` | `preview` | `read` | `reset`

**Introduced in R2017b**

# reset (MDFDatastore)

Reset MDF datastore to initial state

## Syntax

```
reset(mdfds)
```

## Description

`reset(mdfds)` resets the MDF datastore specified by `mdfds` to its initial read state, where no data has been read from it. Resetting allows your to reread from the same datastore.

## Examples

### Reset MDF Datastore

Reset an MDF datastore so that you can read from it again.

```
mdfds = mdfDatastore(fullfile(matlabroot,'examples','vnt','CANape.MF4'));
data = read(mdfds);
reset(mdfds);
data = read(mdfds);
```

## Input Arguments

### `mdfds` — MDF datastore

MDF datastore object

MDF datastore, specified as an MDF datastore object.

Example: `mdfds = mdfDatastore('CANape.MF4')`

## See Also

### Functions

`hasdata` | `mdfDatastore` | `read`

### Introduced in R2017b

# channelList

Information on available MDF groups and channels

## Syntax

```
chans = channelList(mdfobj)
channelList(mdfObj,chanName)
channelList(mdfObj,chanName,'ExactMatch',true)
```

## Description

`chans = channelList(mdfobj)` returns a table of information about channels and groups in the specified MDF-file.

`channelList(mdfObj,chanName)` searches the MDF-file to generate a list of channels matching the specified channel name. The search by default is case-insensitive and identifies partial matches. A table is returned containing information about the matched channels and the containing channel groups. If no matches are found, an empty table is returned.

`channelList(mdfObj,chanName,'ExactMatch',true)` searches the channels for an exact match, including case sensitivity. This is useful if a channel name is a substring of other channel names.

## Examples

### View Available MDF Channels

View all available MDF channels.

```
mdfObj = mdf('File01.mf4');
chans = channelList(mdfObj)
```

```
chans =

  4×9 table

                ChannelName                 ChannelGroupNumber    ChannelGroupNumSamples
    ____________________________________    __________________    ______________________

    "Float_32_LE_Offset_64"                         2                    10000
    "Float_64_LE_Master_Offset_0"                   2                    10000
    "Sigend_Int16_LE_Offset_32"                     1                    10000
    "Unsigend_UInt32_LE_Master_Offset_0"            1                    10000
```

### View Specific MDF Channels

Filter on channel names.

```
chans = channelList(mdfObj,'Float')
```

```
chans =
```

```
  2×9 table

            ChannelName             ChannelGroupNumber    ChannelGroupNumSamples
    ____________________________    __________________    ______________________

    "Float_32_LE_Offset_64"                 2                     10000
    "Float_64_LE_Master_Offset_0"           2                     10000
```

```
chans = channelList(mdfObj,'Float','ExactMatch',true)
```

```
chans =

  0×9 empty table
```

## Input Arguments

### `mdfObj` — MDF-file

MDF-file object

MDF-file, specified as an MDF-file object.

Example: `mdf('File01.mf4')`

### `chanName` — Name of channel

char vector | string

Name of channel, specified as a character vector or string. By default, case-insensitive and partial matches are returned.

Example: `'Channel1'`

Data Types: `char` | `string`

## Output Arguments

### `chans` — Information on available MDF channels

table

Information on available MDF channels, returned as a table. To access specific elements, you can index into the table.

## See Also

**Functions**

`mdf`

**Introduced in R2018b**

# mdfVisualize

View channel data from MDF-file

## Syntax

```
mdfVisualize(mdfFileName)
```

## Description

`mdfVisualize(mdfFileName)` opens an MDF-file in the Simulation Data Inspector for viewing and interacting with channel data. `mdfFileName` is the name of the MDF-file, specified as a full or partial path.

**Note** This function is supported only on 64-bit Windows operating systems.

## Examples

### View MDF Data

View the data from a specified MDF-file in the Simulation Data Inspector.

```
mdfVisualize('File01.mf4')
```

## Input Arguments

### `mdfFileName` — MDF-file name

char vector | string

MDF-file name, specified as a character vector or string, including the necessary full or relative path.

Example: `'MDFFile.mf4'`

Data Types: `char` | `string`

## See Also

### Functions

`mdf` | `read`

### Topics

“View and Analyze Simulation Results”

### Introduced in R2019a

# autoblks.pwr.PlantInfo

Analyze powertrain power and energy

## Description

To assess powertrain efficiencies, use the `autoblks.pwr.PlantInfo` object to evaluate and report power and energy for component-level blocks and system-level reference applications.

## Creation

### Syntax

```
VehPwrAnalysis = autoblks.pwr.PlantInfo(SysName)
```

#### Description

MATLAB creates an `autoblks.pwr.PlantInfo` object for the system that you specify. `VehPwrAnalysis = autoblks.pwr.PlantInfo(SysName)` where `SysName` is the name of the model or subsystem that you want to analyze.

#### Input Arguments

**`SysName` — Model name**
character vector

Model that you want to analyze.

Example: `'SiCiPtReferenceApplication'`

Data Types: `char`

### Properties

**`AvgEff` — Average efficiency**
`double`

This property is read-only.

Average efficiency, dimensionless.

**`Eff` — Time series of efficiency**
`time series`

This property is read-only.

Efficiency, $\eta$, dimensionless. To calculate the efficiency, the `Eff` property implements this equation.

$$\eta = \left| \frac{\sum P_{output} - \sum P_{store}(P_{store} > 0)}{\sum P_{input} - \sum P_{store}(P_{store} < 0)} \right|$$

The equation uses these variables.

| | |
|---|---|
| $P_{store}$ | Stored power |
| $P_{input}$, $P_{output}$ | Input and output power logged by Power Accounting Bus Creator block |

### `EnrgyBalanceAbsTol` — Energy balance absolute tolerance

`0.0100` (default)

Energy balance absolute tolerance, $EnrgyBal_{AbsTol}$.

To determine if the system conserves energy, the `isEnrgyBalanced` method checks the energy conservation at each time step.

$$E_{Err} = \sum E_{trans} + \sum E_{nottrans} - \sum E_{store}$$

Blocks change the input energy plus released stored energy to output energy plus stored energy. For example, a mapped engine block uses fuel (not transferred energy) to produce torque (transferred energy) and heat loss (not transferred energy). The total modified energy represents the average between the input fuel energy and the energy exiting the system (torque and heat loss). To calculate the total energy modified by the block, the method uses the integral of the average transferred, not transferred, and stored power.

$$E_{total} = \frac{1}{2} \left( \int_0^{t_{end}} \left( \sum |P_{trans}| + \sum |P_{nottrans}| + \sum |P_{store}| \right) dt \right) \Bigg|_{t = t_{end}}$$

If the energy conservation error is within an error tolerance, the method returns true. Specifically, if either condition is met, the method returns true.

| **Condition** | | |
|---|---|---|
| $\frac{\lvert E_{Err} \rvert}{E_{total}} < EnrgyBal_{RelTol}$ | *or* | $E_{total} < EnrgyBal_{AbsTol}$ |

The equations use these variables.

| | |
|---|---|
| $E_{Err}$ | Energy conservation error |
| $E_{total}$ | Total energy modified by block |
| $EnrgyBal_{RelTol}$, $EnrgyBal_{AbsTol}$ | Energy balance relative and absolute tolerance, respectively |
| $P_{trans}$, $E_{trans}$ | Transferred power and energy, respectively |
| $P_{nottrans}$, $E_{nottrans}$ | Not transferred power and energy, respectively |

| | |
|---|---|
| $P_{store}$, $E_{store}$ | Stored power and energy, respectively |
| $P_{input}$, $P_{output}$ | Input and output power logged by Power Accounting Bus Creator block |

Data Types: `double`

### `EnrgyBalanceRelTol` — Energy balance relative tolerance

`0.0100` (default)

Energy balance relative tolerance, $EnrgyBal_{RelTol}$.

To determine if the system conserves energy, the `isEnrgyBalanced` method checks the energy conservation at each time step.

$$E_{Err} = \sum E_{trans} + \sum E_{nottrans} - \sum E_{store}$$

Blocks change the input energy plus released stored energy to output energy plus stored energy. For example, a mapped engine block uses fuel (not transferred energy) to produce torque (transferred energy) and heat loss (not transferred energy). The total modified energy represents the average between the input fuel energy and the energy exiting the system (torque and heat loss). To calculate the total energy modified by the block, the method uses the integral of the average transferred, not transferred, and stored power.

$$E_{total} = \frac{1}{2}\left( \int_0^{t_{end}} \left( \sum |P_{trans}| + \sum |P_{nottrans}| + \sum |P_{store}| \right) dt \right) \Bigg|_{t = t_{end}}$$

If the energy conservation error is within an error tolerance, the method returns true. Specifically, if either condition is met, the method returns true.

| **Condition** | | |
|---|---|---|
| $\frac{\lvert E_{Err} \rvert}{E_{total}} < EnrgyBal_{RelTol}$ | *or* | $E_{total} < EnrgyBal_{AbsTol}$ |

The equations use these variables.

| | |
|---|---|
| $E_{Err}$ | Energy conservation error |
| $E_{total}$ | Total energy modified by block |
| $EnrgyBal_{RelTol}$, $EnrgyBal_{AbsTol}$ | Energy balance relative and absolute tolerance, respectively |
| $P_{trans}$, $E_{trans}$ | Transferred power and energy, respectively |
| $P_{nottrans}$, $E_{nottrans}$ | Not transferred power and energy, respectively |
| $P_{store}$, $E_{store}$ | Stored power and energy, respectively |
| $P_{input}$, $P_{output}$ | Input and output power logged by Power Accounting Bus Creator block |

Data Types: `double`

### `EnrgyUnits` — Energy units

`MJ` (default) | `J`

Energy units.

Example: `VehPwrAnalysis.EnrgyUnits = 'MJ';`

Data Types: `char`

### `PwrUnits` — Power units

`kW` (default) | `W`

Power units.

Example: `VehPwrAnalysis.PwrUnits = 'kW';`

Data Types: `char`

## Object Methods

| | |
|---|---|
| addLoggedData | Add logged data |
| dispSignalSummary | Display powertrain subsystem energy analysis |
| dispSysSummary | Display powertrain system efficiency |
| findChildSys | Powertrain subsystem energy analysis |
| histogramEff | Display powertrain subsystem efficiency histogram |
| isEnrgyBalanced | Logical flag for energy conservation |
| loggingOff | Turn signal logging off |
| loggingOn | Turn signal logging on |
| run | Run powertrain energy and power analysis |
| sdiSummary | Display Simulation Data Inspector plots of powertrain energy and power |
| xlsSysSummary | Write powertrain energy analysis to spreadsheet |

## Examples

### Create `PlantInfo` Object for Powertrain Energy Analysis

Analyze the power and energy in the conventional vehicle reference application. To create a `PlantInfo` object, see “step 2” on page 8-33.

1. Open the conventional vehicle reference application. By default, the application has a mapped 1.5 L spark-ignition (SI) engine and a dual clutch transmission. Project files open in a writable location.

   ```
   autoblkConVehStart
   ```

2. Set the system name to `SiCiPtReferenceApplication`.

   Create the `autoblks.pwr.PlantInfo` object.

   Use the `PwrUnits` and `EnrgyUnits` properties to specify the units.

   ```
   SysName = 'SiCiPtReferenceApplication';
   VehPwrAnalysis = autoblks.pwr.PlantInfo(SysName);
   VehPwrAnalysis.PwrUnits = 'kW';
   VehPwrAnalysis.EnrgyUnits = 'MJ';
   ```

3. Use the `run` method to turn on logging, run simulation, and add logged data to the object.

```
run(VehPwrAnalysis);
```

4. Use the `dispSysSummary` method to display the results.

```
dispSysSummary(VehPwrAnalysis);
```

5. Use the `xlsSysSummary` method to write the results to a spreadsheet.

```
xlsSysSummary(VehPwrAnalysis,'EnergySummary.xlsx');
```

6. Use the `findChildSys` method to retrieve the `autoblks.pwr.PlantInfo` object for the Engine subsystem.

   To display the results, use the `dispSignalSummary` method.

   Use the `histogramEff` method to display a histogram of the time spent at each engine plant efficiency.

```
EngSysName = 'SiCiPtReferenceApplication/Passenger Car/Engine';
EngPwrAnalysis = findChildSys(VehPwrAnalysis,EngSysName);
dispSignalSummary(EngPwrAnalysis);
histogramEff(EngPwrAnalysis);
```

7. Use the `findChildSys` method to retrieve the `autoblks.pwr.PlantInfo` object for the Drivetrain subsystem.

   To display the results, use the `dispSignalSummary` method.

```
DrvtrnSysName = 'SiCiPtReferenceApplication/Passenger Car/Drivetrain';
DrvtrnPwrAnalysis = findChildSys(VehPwrAnalysis,DrvtrnSysName);
dispSignalSummary(DrvtrnPwrAnalysis);
```

8. To plot the results, use the `sdiSummary` method.

```
sdiSummary(VehPwrAnalysis,{EngSysName,DrvtrnSysName})
```

## See Also

Power Accounting Bus Creator

### Topics

“Conventional Vehicle Powertrain Efficiency”
“Analyze Power and Energy”

### Introduced in R2019a

# dispSignalSummary

Display powertrain subsystem energy analysis

## Syntax

`dispSignalSummary(SubSystem)`

## Description

The `dispSignalSummary(SubSystem)` method displays the subsystem energy for the `autoblks.pwr.PlantInfo` object. Use the `autoblks.pwr.PlantInfo` object to evaluate and report power an energy for component-level blocks and system-level models.

After you use the `findChildSys` method to retrieve the `autoblks.pwr.PlantInfo` object for the subsystem that you want to analyze, use the `dispSignalSummary(SubSystem)` method to display the results.

## Examples

### Use `dispSignalSummary` Method to Display Subsystem Results

Analyze the power and energy in the conventional vehicle reference application. To use the `dispSignalSummary` method to display the engine and drivetrain subsystem results, see “step 6” on page 8-35 and “step 7” on page 8-36.

1. Open the conventional vehicle reference application. By default, the application has a mapped 1.5 L spark-ignition (SI) engine and a dual clutch transmission. Project files open in a writable location.

   ```
   autoblkConVehStart
   ```

2. Set the system name to `SiCiPtReferenceApplication`.

   Create the `autoblks.pwr.PlantInfo` object.

   Use the `PwrUnits` and `EnrgyUnits` properties to specify the units.

   ```
   SysName = 'SiCiPtReferenceApplication';
   VehPwrAnalysis = autoblks.pwr.PlantInfo(SysName);
   VehPwrAnalysis.PwrUnits = 'kW';
   VehPwrAnalysis.EnrgyUnits = 'MJ';
   ```

3. Use the `run` method to turn on logging, run simulation, and add logged data to the object.

   ```
   run(VehPwrAnalysis);
   ```

4. Use the `dispSysSummary` method to display the results.

   ```
   dispSysSummary(VehPwrAnalysis);
   ```

5. Use the `xlsSysSummary` method to write the results to a spreadsheet.

   ```
   xlsSysSummary(VehPwrAnalysis,'EnergySummary.xlsx');
   ```

6. Use the `findChildSys` method to retrieve the `autoblks.pwr.PlantInfo` object for the Engine subsystem.

To display the results, use the `dispSignalSummary` method.

Use the `histogramEff` method to display a histogram of the time spent at each engine plant efficiency.

```
EngSysName = 'SiCiPtReferenceApplication/Passenger Car/Engine';
EngPwrAnalysis = findChildSys(VehPwrAnalysis,EngSysName);
dispSignalSummary(EngPwrAnalysis);
histogramEff(EngPwrAnalysis);
```

7 Use the `findChildSys` method to retrieve the `autoblks.pwr.PlantInfo` object for the Drivetrain subsystem.

To display the results, use the `dispSignalSummary` method.

```
DrvtrnSysName = 'SiCiPtReferenceApplication/Passenger Car/Drivetrain';
DrvtrnPwrAnalysis = findChildSys(VehPwrAnalysis,DrvtrnSysName);
dispSignalSummary(DrvtrnPwrAnalysis);
```

8 To plot the results, use the `sdiSummary` method.

```
sdiSummary(VehPwrAnalysis,{EngSysName,DrvtrnSysName})
```

## Input Arguments

### `SubSystem` — Subsystem name

character vector

Subsystem that you want to analyze.

Example: `'SiCiPtReferenceApplication/Passenger Car/Engine'`

Example: `'SiCiPtReferenceApplication/Passenger Car/Drivetrain'`

Data Types: `char`

## See Also

`autoblks.pwr.PlantInfo`

### Topics

“Analyze Power and Energy”

### Introduced in R2019a

# dispSysSummary

Display powertrain system efficiency

## Syntax

`dispSysSummary(PlantInfoObj)`

## Description

After you use the `run` method to analyze the powertrain power and energy, use the `dispSysSummary(PlantInfoObj)` method to display the system efficiency for the `autoblks.pwr.PlantInfo` object.

Use instances of the `autoblks.pwr.PlantInfo` object to evaluate and report power an energy for component-level blocks and system-level models.

## Examples

### Use `dispSysSummary` Method to Display Energy Analysis Results

Analyze the power and energy in the conventional vehicle reference application. To use the `dispSysSummary` method to display the results, see “step 4” on page 8-37.

1. Open the conventional vehicle reference application. By default, the application has a mapped 1.5 L spark-ignition (SI) engine and a dual clutch transmission. Project files open in a writable location.

   ```
   autoblkConVehStart
   ```

2. Set the system name to `SiCiPtReferenceApplication`.

   Create the `autoblks.pwr.PlantInfo` object.

   Use the `PwrUnits` and `EnrgyUnits` properties to specify the units.

   ```
   SysName = 'SiCiPtReferenceApplication';
   VehPwrAnalysis = autoblks.pwr.PlantInfo(SysName);
   VehPwrAnalysis.PwrUnits = 'kW';
   VehPwrAnalysis.EnrgyUnits = 'MJ';
   ```

3. Use the `run` method to turn on logging, run simulation, and add logged data to the object.

   ```
   run(VehPwrAnalysis);
   ```

4. Use the `dispSysSummary` method to display the results.

   ```
   dispSysSummary(VehPwrAnalysis);
   ```

5. Use the `xlsSysSummary` method to write the results to a spreadsheet.

   ```
   xlsSysSummary(VehPwrAnalysis,'EnergySummary.xlsx');
   ```

6. Use the `findChildSys` method to retrieve the `autoblks.pwr.PlantInfo` object for the Engine subsystem.

   To display the results, use the `dispSignalSummary` method.

Use the `histogramEff` method to display a histogram of the time spent at each engine plant efficiency.

```
EngSysName = 'SiCiPtReferenceApplication/Passenger Car/Engine';
EngPwrAnalysis = findChildSys(VehPwrAnalysis,EngSysName);
dispSignalSummary(EngPwrAnalysis);
histogramEff(EngPwrAnalysis);
```

7 Use the `findChildSys` method to retrieve the `autoblks.pwr.PlantInfo` object for the Drivetrain subsystem.

To display the results, use the `dispSignalSummary` method.

```
DrvtrnSysName = 'SiCiPtReferenceApplication/Passenger Car/Drivetrain';
DrvtrnPwrAnalysis = findChildSys(VehPwrAnalysis,DrvtrnSysName);
dispSignalSummary(DrvtrnPwrAnalysis);
```

8 To plot the results, use the `sdiSummary` method.

```
sdiSummary(VehPwrAnalysis,{EngSysName,DrvtrnSysName})
```

## Input Arguments

### `PlantInfoObj` — Instance of `PlantInfo` object

`autoblks.pwr.PlantInfo` object

`autoblks.pwr.PlantInfo` object for the system that you want to analyze.

## See Also

`autoblks.pwr.PlantInfo`

### Topics

“Analyze Power and Energy”

### Introduced in R2019a

# findChildSys

Powertrain subsystem energy analysis

## Syntax

`findChildSys(PlantInfoObj,SubSystem)`

## Description

The `findChildSys(PlantInfoObj,SubSystem)` method finds and returns an `autoblks.pwr.PlantInfo` object for the subsystem. Use the `autoblks.pwr.PlantInfo` object to evaluate and report power an energy for component-level blocks and system-level reference applications.

After you use the `run` method to analyze the powertrain power and energy, use the `findChildSys` method to evaluate specific subsystems.

## Examples

### Use `findChildSys` Method to Analyze Subsystems

Analyze the power and energy in the conventional vehicle reference application. To use the `findChildSys` method to analyze the engine and drivetrain subsystems, see “step 6” on page 8-39 and “step 7” on page 8-40.

1. Open the conventional vehicle reference application. By default, the application has a mapped 1.5 L spark-ignition (SI) engine and a dual clutch transmission. Project files open in a writable location.

   ```
   autoblkConVehStart
   ```

2. Set the system name to `SiCiPtReferenceApplication`.

   Create the `autoblks.pwr.PlantInfo` object.

   Use the `PwrUnits` and `EnrgyUnits` properties to specify the units.

   ```
   SysName = 'SiCiPtReferenceApplication';
   VehPwrAnalysis = autoblks.pwr.PlantInfo(SysName);
   VehPwrAnalysis.PwrUnits = 'kW';
   VehPwrAnalysis.EnrgyUnits = 'MJ';
   ```

3. Use the `run` method to turn on logging, run simulation, and add logged data to the object.

   ```
   run(VehPwrAnalysis);
   ```

4. Use the `dispSysSummary` method to display the results.

   ```
   dispSysSummary(VehPwrAnalysis);
   ```

5. Use the `xlsSysSummary` method to write the results to a spreadsheet.

   ```
   xlsSysSummary(VehPwrAnalysis,'EnergySummary.xlsx');
   ```

6. Use the `findChildSys` method to retrieve the `autoblks.pwr.PlantInfo` object for the Engine subsystem.

To display the results, use the `dispSignalSummary` method.

Use the `histogramEff` method to display a histogram of the time spent at each engine plant efficiency.

```
EngSysName = 'SiCiPtReferenceApplication/Passenger Car/Engine';
EngPwrAnalysis = findChildSys(VehPwrAnalysis,EngSysName);
dispSignalSummary(EngPwrAnalysis);
histogramEff(EngPwrAnalysis);
```

7 Use the `findChildSys` method to retrieve the `autoblks.pwr.PlantInfo` object for the Drivetrain subsystem.

To display the results, use the `dispSignalSummary` method.

```
DrvtrnSysName = 'SiCiPtReferenceApplication/Passenger Car/Drivetrain';
DrvtrnPwrAnalysis = findChildSys(VehPwrAnalysis,DrvtrnSysName);
dispSignalSummary(DrvtrnPwrAnalysis);
```

8 To plot the results, use the `sdiSummary` method.

```
sdiSummary(VehPwrAnalysis,{EngSysName,DrvtrnSysName})
```

## Input Arguments

### `PlantInfoObj` — Instance of `PlantInfo` object

`autoblks.pwr.PlantInfo` object

`autoblks.pwr.PlantInfo` object for the system that you want to analyze.

### `SubSystem` — Subsystem name

character vector

Subsystem that you want to analyze.

Example: `'SiCiPtReferenceApplication/Passenger Car/Engine'`

Example: `'SiCiPtReferenceApplication/Passenger Car/Drivetrain'`

Data Types: `char`

## See Also

`autoblks.pwr.PlantInfo`

### Topics

“Analyze Power and Energy”

### Introduced in R2019a

# histogramEff

Display powertrain subsystem efficiency histogram

## Syntax

```
histogramEff(SubSystem)
```

## Description

The `histogramEff(SubSystem)` method displays a histogram of the powertrain subsystem efficiency for the `autoblks.pwr.PlantInfo` object. Use instances of the `autoblks.pwr.PlantInfo` object to evaluate and report power an energy for component-level blocks and system-level models.

After you use the `findChildSys` method to analyze the powertrain subsystem power and energy, use the `histogramEff` method to display a histogram of the efficiency.

## Examples

### Use `histogramEff` Method to Display Results

Analyze the power and energy in the conventional vehicle reference application. To use the `histogramEff` method to display a histogram of the time spent at each engine plant efficiency, see “step 6” on page 8-41.

1. Open the conventional vehicle reference application. By default, the application has a mapped 1.5 L spark-ignition (SI) engine and a dual clutch transmission. Project files open in a writable location.

   ```
   autoblkConVehStart
   ```
2. Set the system name to `SiCiPtReferenceApplication`.

   Create the `autoblks.pwr.PlantInfo` object.

   Use the `PwrUnits` and `EnrgyUnits` properties to specify the units.

   ```
   SysName = 'SiCiPtReferenceApplication';
   VehPwrAnalysis = autoblks.pwr.PlantInfo(SysName);
   VehPwrAnalysis.PwrUnits = 'kW';
   VehPwrAnalysis.EnrgyUnits = 'MJ';
   ```
3. Use the `run` method to turn on logging, run simulation, and add logged data to the object.

   ```
   run(VehPwrAnalysis);
   ```
4. Use the `dispSysSummary` method to display the results.

   ```
   dispSysSummary(VehPwrAnalysis);
   ```
5. Use the `xlsSysSummary` method to write the results to a spreadsheet.

   ```
   xlsSysSummary(VehPwrAnalysis,'EnergySummary.xlsx');
   ```
6. Use the `findChildSys` method to retrieve the `autoblks.pwr.PlantInfo` object for the Engine subsystem.

To display the results, use the `dispSignalSummary` method.

Use the `histogramEff` method to display a histogram of the time spent at each engine plant efficiency.

```
EngSysName = 'SiCiPtReferenceApplication/Passenger Car/Engine';
EngPwrAnalysis = findChildSys(VehPwrAnalysis,EngSysName);
dispSignalSummary(EngPwrAnalysis);
histogramEff(EngPwrAnalysis);
```

7 Use the `findChildSys` method to retrieve the `autoblks.pwr.PlantInfo` object for the Drivetrain subsystem.

To display the results, use the `dispSignalSummary` method.

```
DrvtrnSysName = 'SiCiPtReferenceApplication/Passenger Car/Drivetrain';
DrvtrnPwrAnalysis = findChildSys(VehPwrAnalysis,DrvtrnSysName);
dispSignalSummary(DrvtrnPwrAnalysis);
```

8 To plot the results, use the `sdiSummary` method.

```
sdiSummary(VehPwrAnalysis,{EngSysName,DrvtrnSysName})
```

## Input Arguments

### `SubSystem` — Subsystem name

character vector

Subsystem that you want to analyze.

Example: `'SiCiPtReferenceApplication/Passenger Car/Engine'`

Example: `'SiCiPtReferenceApplication/Passenger Car/Drivetrain'`

Data Types: `char`

## See Also

`autoblks.pwr.PlantInfo`

### Topics

“Analyze Power and Energy”

### Introduced in R2019a

# run

Run powertrain energy and power analysis

## Syntax

`run(PlantInfoObj)`

## Description

Use the `run(PlantInfoObj)` method to turn signal logging on, run a powertrain energy and power analysis, and add data to the `autoblks.pwr.PlantInfo` object. Use instances of the `autoblks.pwr.PlantInfo` object to evaluate and report power an energy for component-level blocks and system-level models.

## Examples

### Use `run` Method for Powertrain Energy Analysis

Analyze the power and energy in the conventional vehicle reference application. To use the `run` method for the analysis, see “step 3” on page 8-43.

1. Open the conventional vehicle reference application. By default, the application has a mapped 1.5 L spark-ignition (SI) engine and a dual clutch transmission. Project files open in a writable location.

   ```
   autoblkConVehStart
   ```

2. Set the system name to `SiCiPtReferenceApplication`.

   Create the `autoblks.pwr.PlantInfo` object.

   Use the `PwrUnits` and `EnrgyUnits` properties to specify the units.

   ```
   SysName = 'SiCiPtReferenceApplication';
   VehPwrAnalysis = autoblks.pwr.PlantInfo(SysName);
   VehPwrAnalysis.PwrUnits = 'kW';
   VehPwrAnalysis.EnrgyUnits = 'MJ';
   ```

3. Use the `run` method to turn on logging, run simulation, and add logged data to the object.

   ```
   run(VehPwrAnalysis);
   ```

4. Use the `dispSysSummary` method to display the results.

   ```
   dispSysSummary(VehPwrAnalysis);
   ```

5. Use the `xlsSysSummary` method to write the results to a spreadsheet.

   ```
   xlsSysSummary(VehPwrAnalysis,'EnergySummary.xlsx');
   ```

6. Use the `findChildSys` method to retrieve the `autoblks.pwr.PlantInfo` object for the Engine subsystem.

   To display the results, use the `dispSignalSummary` method.

   Use the `histogramEff` method to display a histogram of the time spent at each engine plant efficiency.

```
EngSysName = 'SiCiPtReferenceApplication/Passenger Car/Engine';
EngPwrAnalysis = findChildSys(VehPwrAnalysis,EngSysName);
dispSignalSummary(EngPwrAnalysis);
histogramEff(EngPwrAnalysis);
```

7 Use the `findChildSys` method to retrieve the `autoblks.pwr.PlantInfo` object for the Drivetrain subsystem.

To display the results, use the `dispSignalSummary` method.

```
DrvtrnSysName = 'SiCiPtReferenceApplication/Passenger Car/Drivetrain';
DrvtrnPwrAnalysis = findChildSys(VehPwrAnalysis,DrvtrnSysName);
dispSignalSummary(DrvtrnPwrAnalysis);
```

8 To plot the results, use the `sdiSummary` method.

```
sdiSummary(VehPwrAnalysis,{EngSysName,DrvtrnSysName})
```

## Input Arguments

### `PlantInfoObj` — Instance of `PlantInfo` object

`autoblks.pwr.PlantInfo` object

`autoblks.pwr.PlantInfo` object for the system that you want to analyze.

## See Also

`autoblks.pwr.PlantInfo`

### Topics

“Analyze Power and Energy”

### Introduced in R2019a

# sdiSummary

Display Simulation Data Inspector plots of powertrain energy and power

## Syntax

`sdiSummary(PlantInfoObj,blocknames)`

## Description

The `sdiSummary(PlantInfoObj,blocknames)` method plots the powertrain energy and power analysis results for the `autoblks.pwr.PlantInfo` object.

Use instances of the `autoblks.pwr.PlantInfo` object to evaluate and report power an energy for component-level blocks and system-level models.

## Examples

### Use `sdiSummary` Method to Plot Results

Analyze the power and energy in the conventional vehicle reference application. To use the `sdiSummary` method to display the Simulation Data Inspector plots of the engine and drivetrain results, see “step 8” on page 8-46.

1. Open the conventional vehicle reference application. By default, the application has a mapped 1.5 L spark-ignition (SI) engine and a dual clutch transmission. Project files open in a writable location.

   ```
   autoblkConVehStart
   ```

2. Set the system name to `SiCiPtReferenceApplication`.

   Create the `autoblks.pwr.PlantInfo` object.

   Use the `PwrUnits` and `EnrgyUnits` properties to specify the units.

   ```
   SysName = 'SiCiPtReferenceApplication';
   VehPwrAnalysis = autoblks.pwr.PlantInfo(SysName);
   VehPwrAnalysis.PwrUnits = 'kW';
   VehPwrAnalysis.EnrgyUnits = 'MJ';
   ```

3. Use the `run` method to turn on logging, run simulation, and add logged data to the object.

   ```
   run(VehPwrAnalysis);
   ```

4. Use the `dispSysSummary` method to display the results.

   ```
   dispSysSummary(VehPwrAnalysis);
   ```

5. Use the `xlsSysSummary` method to write the results to a spreadsheet.

   ```
   xlsSysSummary(VehPwrAnalysis,'EnergySummary.xlsx');
   ```

6. Use the `findChildSys` method to retrieve the `autoblks.pwr.PlantInfo` object for the Engine subsystem.

   To display the results, use the `dispSignalSummary` method.

Use the `histogramEff` method to display a histogram of the time spent at each engine plant efficiency.

```
EngSysName = 'SiCiPtReferenceApplication/Passenger Car/Engine';
EngPwrAnalysis = findChildSys(VehPwrAnalysis,EngSysName);
dispSignalSummary(EngPwrAnalysis);
histogramEff(EngPwrAnalysis);
```

7 Use the `findChildSys` method to retrieve the `autoblks.pwr.PlantInfo` object for the Drivetrain subsystem.

To display the results, use the `dispSignalSummary` method.

```
DrvtrnSysName = 'SiCiPtReferenceApplication/Passenger Car/Drivetrain';
DrvtrnPwrAnalysis = findChildSys(VehPwrAnalysis,DrvtrnSysName);
dispSignalSummary(DrvtrnPwrAnalysis);
```

8 To plot the results, use the `sdiSummary` method.

```
sdiSummary(VehPwrAnalysis,{EngSysName,DrvtrnSysName})
```

## Input Arguments

### `PlantInfoObj` — Instance of `PlantInfo` object

`autoblks.pwr.PlantInfo` object

`autoblks.pwr.PlantInfo` object for the system that you want to analyze.

### `blocknames` — Block or name

character vector | string | 'all'

Block or subsystem names, specified as a character vector or a string, separated by a comma.

Example: `'SiCiPtReferenceApplication/Passenger Car/Engine'`

Example: `'SiCiPtReferenceApplication/Passenger Car/Engine','SiCiPtReferenceApplication/Passenger Car/Drivetrain'`

Data Types: `char` | `string`

## See Also

`autoblks.pwr.PlantInfo`

### Topics

“Analyze Power and Energy”
Simulation Data Inspector

### Introduced in R2019a

# xlsSysSummary

Write powertrain energy analysis to spreadsheet

## Syntax

`xlsSysSummary(PlantInfoObj,filename,sheet)`

## Description

The `xlsSysSummary(PlantInfoObj,filename,sheet)` method exports the system energy and efficiency for the `autoblks.pwr.PlantInfo` object. Use the `autoblks.pwr.PlantInfo` object to evaluate and report power an energy for component-level blocks and system-level models.

After you use the `run` method to analyze the powertrain power and energy, use the `xlsSysSummary` method to write the results to a spreadsheet.

## Examples

### Use `xlsSysSummary` Method to Write Results to Spreadsheet

Analyze the power and energy in the conventional vehicle reference application. To use the `xlsSysSummary` method to write the results to a spreadsheet, see “step 5” on page 8-47.

1. Open the conventional vehicle reference application. By default, the application has a mapped 1.5 L spark-ignition (SI) engine and a dual clutch transmission. Project files open in a writable location.

   ```
   autoblkConVehStart
   ```

2. Set the system name to `SiCiPtReferenceApplication`.

   Create the `autoblks.pwr.PlantInfo` object.

   Use the `PwrUnits` and `EnrgyUnits` properties to specify the units.

   ```
   SysName = 'SiCiPtReferenceApplication';
   VehPwrAnalysis = autoblks.pwr.PlantInfo(SysName);
   VehPwrAnalysis.PwrUnits = 'kW';
   VehPwrAnalysis.EnrgyUnits = 'MJ';
   ```

3. Use the `run` method to turn on logging, run simulation, and add logged data to the object.

   ```
   run(VehPwrAnalysis);
   ```

4. Use the `dispSysSummary` method to display the results.

   ```
   dispSysSummary(VehPwrAnalysis);
   ```

5. Use the `xlsSysSummary` method to write the results to a spreadsheet.

   ```
   xlsSysSummary(VehPwrAnalysis,'EnergySummary.xlsx');
   ```

6. Use the `findChildSys` method to retrieve the `autoblks.pwr.PlantInfo` object for the Engine subsystem.

   To display the results, use the `dispSignalSummary` method.

Use the `histogramEff` method to display a histogram of the time spent at each engine plant efficiency.

```
EngSysName = 'SiCiPtReferenceApplication/Passenger Car/Engine';
EngPwrAnalysis = findChildSys(VehPwrAnalysis,EngSysName);
dispSignalSummary(EngPwrAnalysis);
histogramEff(EngPwrAnalysis);
```

7. Use the `findChildSys` method to retrieve the `autoblks.pwr.PlantInfo` object for the Drivetrain subsystem.

   To display the results, use the `dispSignalSummary` method.

```
DrvtrnSysName = 'SiCiPtReferenceApplication/Passenger Car/Drivetrain';
DrvtrnPwrAnalysis = findChildSys(VehPwrAnalysis,DrvtrnSysName);
dispSignalSummary(DrvtrnPwrAnalysis);
```

8. To plot the results, use the `sdiSummary` method.

```
sdiSummary(VehPwrAnalysis,{EngSysName,DrvtrnSysName})
```

## Input Arguments

### `PlantInfoObj` — Instance of `PlantInfo` object

`autoblks.pwr.PlantInfo` object

`autoblks.pwr.PlantInfo` object for the system that you want to analyze.

### `filename` — File name

character vector | string

File name, specified as a character vector or a string.

If `filename` does not exist, `xlsSysSummary` creates a file, determining the format based on the specified extension. To create a file compatible with Excel® 97–2003 software, specify an extension of `.xls`. To create files in Excel 2007 formats, specify an extension of `.xlsx`, `.xlsb`, or `.xlsm`. If you do not specify an extension, `xlsSysSummary` uses the default, `.xls`.

Example: `'myFile.xlsx'` or `"myFile.xlsx"`

Example: `'C:\myFolder\myFile.xlsx'`

Example: `'myFile.csv'`

Data Types: `char` | `string`

### `sheet` — Worksheet name

character vector | string | positive integer

Worksheet name, specified as one of the following:

- Character vector or string that contains the worksheet name. The name cannot contain a colon (`:`). To determine the names of the sheets in a spreadsheet file, use `xlsfinfo`.
- Positive integer that indicates the worksheet index.

If `sheet` does not exist, `xlswrite` adds a sheet at the end of the worksheet collection. If `sheet` is an index larger than the number of worksheets, `xlswrite` appends empty sheets until the number of worksheets in the workbook equals `sheet`. In either case, `xlswrite` generates a warning indicating that it has added a worksheet.

Data Types: `char` | `string` | `single` | `double` | `int8` | `int16` | `int32` | `int64` | `uint8` | `uint16` | `uint32` | `uint64`

## See Also

`autoblks.pwr.PlantInfo` | `xlswrite`

**Topics**

“Analyze Power and Energy”

**Introduced in R2019a**

# addLoggedData

Add logged data

## Syntax

```
addLoggedData(PlantInfoObj,logsout)
```

## Description

`addLoggedData(PlantInfoObj,logsout)` adds logged signal data to the `autoblks.pwr.PlantInfo` object specified by the `Simulink.SimulationData.Dataset` signal data object.

If the data logged for the system does not conserve energy, the method returns a warning.

If the `Simulink.SimulationData.Dataset` object does not include data for the Power Accounting Bus Creator blocks in the system, the method returns an error.

## Input Arguments

**`PlantInfoObj` — Instance of `PlantInfo` object**
`autoblks.pwr.PlantInfo` object

`autoblks.pwr.PlantInfo` object for the system that you want to analyze.

**`logsout` — Dataset object for signals**
`Simulink.SimulationData.Dataset` object

`Simulink.SimulationData.Dataset` object for signals that you want to log.

## See Also

Power Accounting Bus Creator | `autoblks.pwr.PlantInfo`

**Topics**
“Analyze Power and Energy”

**Introduced in R2019a**

# isEnrgyBalanced

Logical flag for energy conservation

## Syntax

```
flag=isEnrgyBalanced(PlantInfoObj)
```

## Description

`flag=isEnrgyBalanced(PlantInfoObj)` returns logical `1` (`true`) if the system conserves energy. Otherwise, it returns logical `0` (`false`).

## Input Arguments

### `PlantInfoObj` — Instance of `PlantInfo` object

`autoblks.pwr.PlantInfo` object

`autoblks.pwr.PlantInfo` object for the system that you want to analyze.

## Output Arguments

### `flag` — Indicator of energy conservation

`1` (`true`) | `0` (`false`)

Indicator of energy conservation, returned as a logical `1` (`true`) or `0` (`false`).

Data Types: `logical`

## Algorithms

To determine if the system conserves energy, the `isEnrgyBalanced` method checks the energy conservation at each time step.

$$E_{Err} = \sum E_{trans} + \sum E_{nottrans} - \sum E_{store}$$

Blocks change the input energy plus released stored energy to output energy plus stored energy. For example, a mapped engine block uses fuel (not transferred energy) to produce torque (transferred energy) and heat loss (not transferred energy). The total modified energy represents the average between the input fuel energy and the energy exiting the system (torque and heat loss). To calculate the total energy modified by the block, the method uses the integral of the average transferred, not transferred, and stored power.

$$E_{total} = \frac{1}{2}\left( \int_0^{t_{end}} \left( \sum |P_{trans}| + \sum |P_{nottrans}| + \sum |P_{store}| \right) dt \right) \Bigg|_{t = t_{end}}$$

If the energy conservation error is within an error tolerance, the method returns true. Specifically, if either condition is met, the method returns true.

| Condition | | |
|---|---|---|
| $\frac{\lvert E_{Err}\rvert}{E_{total}} < EnrgyBal_{RelTol}$ | *or* | $E_{total} < EnrgyBal_{AbsTol}$ |

The equations use these variables.

| | |
|---|---|
| $E_{Err}$ | Energy conservation error |
| $E_{total}$ | Total energy modified by block |
| $EnrgyBal_{RelTol}$, $EnrgyBal_{AbsTol}$ | Energy balance relative and absolute tolerance, respectively |
| $P_{trans}$, $E_{trans}$ | Transferred power and energy, respectively |
| $P_{nottrans}$, $E_{nottrans}$ | Not transferred power and energy, respectively |
| $P_{store}$, $E_{store}$ | Stored power and energy, respectively |
| $P_{input}$, $P_{output}$ | Input and output power logged by Power Accounting Bus Creator block |

## See Also

Power Accounting Bus Creator | `autoblks.pwr.PlantInfo`

**Topics**

“Analyze Power and Energy”

**Introduced in R2019a**

# loggingOff

Turn signal logging off

## Syntax

```
loggingOff(PlantInfoObj)
```

## Description

`loggingOff(PlantInfoObj)` turns signal logging off for all Power Accounting Bus Creator blocks in the `autoblks.pwr.PlantInfo` system object.

## Input Arguments

### `PlantInfoObj` — Instance of `PlantInfo` object

`autoblks.pwr.PlantInfo` object

`autoblks.pwr.PlantInfo` object for the system that you want to analyze.

## See Also

Power Accounting Bus Creator | `autoblks.pwr.PlantInfo`

### Topics

"Analyze Power and Energy"

### Introduced in R2019a

# loggingOn

Turn signal logging on

## Syntax

```
loggingOn(PlantInfoObj)
```

## Description

`loggingOn(PlantInfoObj)` turns signal logging on for all Power Accounting Bus Creator blocks in the `autoblks.pwr.PlantInfo` system object.

## Input Arguments

### `PlantInfoObj` — Instance of `PlantInfo` object

`autoblks.pwr.PlantInfo` object

`autoblks.pwr.PlantInfo` object for the system that you want to analyze.

## See Also

Power Accounting Bus Creator | `autoblks.pwr.PlantInfo`

**Topics**

"Analyze Power and Energy"

**Introduced in R2019a**

# Battery.PulseSequence

Define a single pulse sequence

## Description

Use the `Battery.PulseSequence` object to define a single experimental pulse sequence at a specific temperature and pulse current magnitude.

You can place multiple experimental pulse sequences into an array of `Battery.PulseSequence` objects. To do so, create a `Battery.PulseSequence` object for each experimental pulse sequence instance.

To use the `Battery.PulseSequence` object and methods, you need these products:

- Powertrain Blockset
- Curve Fitting Toolbox™
- Optimization Toolbox™
- Parallel Computing Toolbox™
- Simulink Design Optimization

## Creation

### Syntax

```
psObj = Battery.PulseSequence
```

#### Description

MATLAB creates a `psObj = Battery.PulseSequence` object that defines a pulse sequence.

### Properties

**`Data` — Raw data**
`m`-by-5 array

An `m`-by-5 array of pulse sequence data. Use the `addData` object function to add the data. `addData` computes the charge and state of charge (SOC), using the assumption that the experimental test ranges is 0% to 100% SOC.

| Array Element | Description | Unit |
|---|---|---|
| `Data(m,1)` | Time | s |
| `Data(m,2)` | Voltage | V |
| `Data(m,3)` | Current | A |
| `Data(m,4)` | Charge | A·s |

| Array Element | Description | Unit |
|---|---|---|
| `Data(m,5)` | State of charge (SOC) | Dimensionless |

Data Types: `double`

**`ModelName` — Name of model**
character vector

Name of the model to use for simulation

Example: `'BatteryEstim3RC_PTBS'`

Data Types: `char`

**`MetaData` — `Battery.MetaData` object properties**
`0`-by-`1` array

`Battery.MetaData` object properties containing metadata for the data.

Data Types: `function_handle`

**`Capacity` — Pulse sequence capacity**
scalar

Capacity observed as the difference between lowest and highest energy, in A·s. Calculated by the `addData` method, but can be overwritten.

Example: `0.0`

Data Types: `double`

**`Parameters` — `Battery.Parameters` object properties**
`0`-by-`1` array

`Battery.Parameters` object containing the most recently determined battery equivalent circuit parameters.

Data Types: `function_handle`

**`ParametersHistory` — `Battery.ParametersHistory` object properties**
`0`-by-`1` array

`Battery.ParametersHistory` object array containing the history of the battery equivalent circuit parameters through different estimation steps. The last element is the most recent parameter set.

Data Types: `function_handle`

## Object Functions

| | |
|---|---|
| addData | Import pulse sequence experimental data |
| createPulses | Identify pulses and create pulse objects from experimental data |
| estimateInitialEmR0 | Estimate open circuit voltage and series resistance |
| estimateInitialEmRx | Estimate open circuit voltage and RC pair resistance |
| estimateInitialTau | Estimate RC pair time constant |
| estimateParameters | Estimate parameters |
| getSocIdxForPulses | Return state of charge index for pulses |
| loadDataFromMatFile | Load pulse data from a MAT-file |

| | |
|---|---|
| plot | Plot pulse sequence data |
| plotIdentifiedPulses | Plot identified pulses |
| plotLatestParameters | Plot latest pulse sequence parameters |
| plotSimulationResults | Plot pulse sequence simulation results |
| populatePulseParameters | Populate pulse parameters |
| removePulses | Remove pulses from sequence |
| repairTimeVector | Repair time vector |

## Examples

### Add File Data to `Battery.PulseSequence` Object

This example shows how to add data to a `Battery.PulseSequence` object.

Create a pulse sequence object.

```
psObj = Battery.PulseSequence;
disp(psObj)
```

Load data from a file.

```
FileName = 'Synthetic_LiPo_PulseDischarge.mat';
[time,voltage,current] = Battery.loadDataFromMatFile(FileName);
```

Add the data to the pulse sequence.

```
addData(psObj,time,voltage,current);
```

## See Also

`Battery.MetaData` | `Battery.Parameters` | `Battery.Pulse` | `sdo.OptimizeOptions`

### Topics

“Generate Parameter Data for Datasheet Battery Block”
“Generate Parameter Data for Equivalent Circuit Battery Block”

### Introduced in R2016b

# addData

Import pulse sequence experimental data

## Syntax

```
addData(psObj,Time,Voltage,Current)
```

## Description

`addData(psObj,Time,Voltage,Current)` adds the pulse sequence experimental data to the `Battery.PulseSequence` object. The `Time`, `Voltage`, and `Current` input arrays must have equal lengths. `addData` computes the charge and state of charge (SOC), using the assumption that the experimental test range is 0% to 100% SOC.

## Examples

### Add Data to `Battery.PulseSequence` Object

This example shows how to add data to a `Battery.PulseSequence` object.

Create a pulse sequence object.

```
psObj = Battery.PulseSequence;
disp(psObj)
```

Load data from a file.

```
FileName = 'Synthetic_LiPo_PulseDischarge.mat';
[time,voltage,current] = Battery.loadDataFromMatFile(FileName);
```

Add the data to the pulse sequence.

```
addData(psObj,time,voltage,current);
```

## Input Arguments

### `psObj` — Instance of `Battery.PulseSequence` class

`Battery.PulseSequence` object

`Battery.PulseSequence` object for the pulse sequence that you want to analyze.

### `Time` — Time

`m`-by-`1` array

`m`-by-`1` array of time data, in s.

Data Types: `double`

### `Voltage` — Voltage

`m`-by-`1` array

`m`-by-1 array of voltage data, in V.

Data Types: `double`

**`Current` — Current**
`m`-by-1 array

`m`-by-1 array of current data, in A.

Data Types: `double`

## See Also

`Battery.PulseSequence`

**Topics**
“Generate Parameter Data for Datasheet Battery Block”
“Generate Parameter Data for Equivalent Circuit Battery Block”

**Introduced in R2016b**

# loadDataFromMatFile

Load pulse data from a MAT-file

## Syntax

```
[Time,Voltage,Current] = loadDataFromMatFile(FileName)
[Time,Voltage,Current] = loadDataFromMatFile(FileName,Name,Value)
```

## Description

`[Time,Voltage,Current] = loadDataFromMatFile(FileName)` function loads pulse data from a MAT-file.

`[Time,Voltage,Current] = loadDataFromMatFile(FileName,Name,Value)` function loads pulse data from a MAT-file with additional options specified by one or more `Name,Value` pair arguments.

## Examples

### Load File Data to `Battery.PulseSequence` Object

This example shows how to add data to a `Battery.PulseSequence` object.

Create a pulse sequence object.

```
psObj = Battery.PulseSequence;
disp(psObj);
```

Load data from a file.

```
FileName = 'Synthetic_LiPo_PulseDischarge.mat';
[time,voltage,current] = Battery.loadDataFromMatFile(FileName);
```

Add the data to the pulse sequence.

```
addData(psObj,time,voltage,current);
```

## Input Arguments

### `FileName` — Path or file name

`untitled.mat` (default) | path, or MAT-file name

Path or file name of the MAT-file that contains the pulse sequence data.

Example: `'Synthetic_LiPo_PulseDischarge.mat'`

Data Types: `char`

#### Name-Value Pair Arguments

Specify optional comma-separated pairs of `Name,Value` arguments. `Name` is the argument name and `Value` is the corresponding value. `Name` must appear inside quotes. You can specify several name and value pair arguments in any order as `Name1,Value1,...,NameN,ValueN`.

Example: `Battery.loadDataFromMatFile(FileName,'TimeVariable','myTimeVariable')`

**`TimeVariable` — Time variable in MAT-file**
`time` (default) | `character vector`

Use this value to specify the time variable to search for in the MAT-file. If unspecified, the method searches for variables containing `'time'`.

Example: `Battery.loadDataFromMatFile(FileName,'TimeVariable','myTimeVariable')`

Data Types: `char`

**`VoltageVariable` — Voltage variable in MAT-file**
`volt` (default) | `character vector`

Use this value to specify the voltage variable to search for in the MAT-file. If unspecified, the method searches for variables containing `'voltage'`.

Example:
`Battery.loadDataFromMatFile(FileName,'VoltageVariable','myVoltageVariable')`

Data Types: `char`

**`CurrentVariable` — Current variable in MAT-file**
`current` (default) | `character vector`

Use this value to specify the current variable to search for in the MAT-file. If unspecified, the method searches for variables containing `'current'`.

Example:
`Battery.loadDataFromMatFile(FileName,'CurrentVariable','myCurrentVariable')`

Data Types: `char`

## Output Arguments

**`Time` — Time**
m-by-1 array

m-by-1 array of time data, in s.

Data Types: `double`

**`Voltage` — Voltage**
m-by-1 array

m-by-1 array of voltage data, in V.

Data Types: `double`

**`Current` — Current**
m-by-1 array

`m`-by-1 array of current data, in A.

Data Types: `double`

## See Also

`Battery.PulseSequence`

**Topics**

“Generate Parameter Data for Datasheet Battery Block”
“Generate Parameter Data for Equivalent Circuit Battery Block”

**Introduced in R2016b**

# createPulses

Identify pulses and create pulse objects from experimental data

## Syntax

```
createPulses(psObj)
createPulses(psObj,Name,Value)
```

## Description

`createPulses(psObj)` identifies the location of pulse events. Creates separate pulse objects from the `Battery.PulseSequence` object experimental data.

`createPulses(psObj,Name,Value)` identifies the location of pulse events. Creates separate pulse objects from the `Battery.PulseSequence` object experimental data with additional options specified by one or more `Name,Value` pair arguments.

## Examples

### Create Pulse Objects from Data

This example shows how to create pulse objects from data.

Create a pulse sequence object.

```
psObj = Battery.PulseSequence;
disp(psObj)
```

Load data from a file.

```
FileName = 'Synthetic_LiPo_PulseDischarge.mat';
[time,voltage,current] = Battery.loadDataFromMatFile(FileName);
```

Add the data to the pulse sequence.

```
addData(psObj,time,voltage,current);
```

Create pulse objects from data.

```
createPulses(psObj,...
    'CurrentOnThreshold',0.1,...
    'NumRCBranches',3,...
    'RCBranchesUse2TimeConstants',false,...
    'PreBufferSamples',10,...
    'PostBufferSamples',15);
```

## Input Arguments

**`psObj` — Instance of `Battery.PulseSequence` class**
`Battery.PulseSequence` object

`Battery.PulseSequence` object for the pulse sequence that you want to analyze.

**Name-Value Pair Arguments**

Specify optional comma-separated pairs of `Name,Value` arguments. `Name` is the argument name and `Value` is the corresponding value. `Name` must appear inside quotes. You can specify several name and value pair arguments in any order as `Name1,Value1,...,NameN,ValueN`.

Example: `createPulses(psObj,'CurrentOnThreshold',0.1)`

**`CurrentOnThreshold` — Minimum current magnitude**
`0.025` (default) | `scalar`

Use this value to specify the minimum current magnitude for identifying the pulse locations, in A. The `createPulses` function considers values below the `CurrentOnThrehsold` as relaxation or measurement noise.

Example: `createPulses(psObj,'CurrentOnThreshold',0.1)`

Data Types: `double`

**`NumRCBranches` — Number of RC branches**
`3` (default) | `scalar`

Use this value to specify the number of RC branches. To change the number of branches after an estimation, you must rerun `createPulses` along with any estimation steps. Rerunning ensures that the estimation parameters are the right size.

Example: `createPulses(psObj,'NumRCBranches',4)`

Data Types: `uint32`

**`RCBranchesUse2TimeConstants` — Use load and relaxation time constants**
`false`

The `createPulses` function does not support using separate time constants for load and relaxation when it estimates each RC branch. If you set the value to true, the `createPulses` function might produce an error.

Example: `createPulses(psObj,'RCBranchesUse2TimeConstants',false)`

Data Types: `logical`

**`PreBufferSamples` — Pulses to retain before pulse estimation**
`10` (default) | `scalar`

Use this value to specify the number of pulses to retain before pulse estimation. The buffer allows the estimation to focus on matching the measured data before the pulse begins.

Example: `createPulses(psObj,'PreBufferSamples',5)`

Data Types: `uint32`

**`PostBufferSamples` — Pulses to retain for next estimation**
`15` (default) | `scalar`

Use this value to specify the number of pulses to retain before the next pulse estimation. The buffer allows the estimation to focus on matching the transition when the next pulse begins. Typically, the end transition of one pulse and the starting transition at the next pulse are at the same state of

charge (SOC). Therefore, both transitions help determine the parameter values at that SOC breakpoint.

Example: `createPulses(psObj,'PostBufferSamples',14)`

Data Types: `uint32`

**`PulseRequires2Samples` — Pulse requires two consecutive samples under current**
`false` (default)

Use this value to specify that there must be two consecutive samples under current to define a pulse. Set to `true` if occasional noise spikes in the current measurement trigger a false pulse detection. By default, the value is `false`, indicating that a single sample above the threshold detects a pulse event.

Example: `createPulses(psObj,'PulseRequires2Samples',true)`

Data Types: `logical`

## See Also

`Battery.PulseSequence`

**Topics**
“Generate Parameter Data for Datasheet Battery Block”
“Generate Parameter Data for Equivalent Circuit Battery Block”

**Introduced in R2016b**

# estimateInitialEmR0

Estimate open circuit voltage and series resistance

## Syntax

```
estimateInitialEmR0(psObj)
estimateInitialEmR0(psObj,Name,Value)
```

## Description

`estimateInitialEmR0(psObj)` estimates the open circuit voltage, *Em*, and series resistance, *Ro*, for the `Battery.PulseSequence` object data. For the estimation, the method uses data points around each pulse transition. The method uses estimated values to determine the minimum and maximum constraint values. The method stores the results in an `Battery.Parameters` object.

`estimateInitialEmR0(psObj,Name,Value)` estimates the open circuit voltage, *Em*, and series resistance, *Ro*, for the `Battery.PulseSequence` object data with additional options specified by one or more `Name,Value` pair arguments.

## Input Arguments

### psObj — Instance of Battery.PulseSequence class

`Battery.PulseSequence` object

`Battery.PulseSequence` object for the pulse sequence that you want to analyze.

#### Name-Value Pair Arguments

Specify optional comma-separated pairs of `Name,Value` arguments. `Name` is the argument name and `Value` is the corresponding value. `Name` must appear inside quotes. You can specify several name and value pair arguments in any order as `Name1,Value1,...,NameN,ValueN`.

Example:
`estimateInitialEmR0(psObj,'SetEmConstraints',false,'EstimateEm',true,'EstimateR0',true)`

### SetEmConstraints — Use open circuit voltage constraints

`true` (default)

Use this value to specify if the method constrains the open circuit voltage, *Eo*, to within maximum or minimum values. To determine the maximum and minimum voltage, the method uses the voltage at the end of relaxation as a constraint for future estimation steps.

If the pulse is a discharge pulse, the voltage rises during relaxation. The final relaxation voltage is set to the minimum constraint for Eo at the corresponding state of charge (SOC).

If the pulse is a charge pulse, the voltage falls during relaxation. The final relaxation voltage is set to the maximum constraint at the corresponding SOC.

Example: `estimateInitialEmR0(psObj,'SetEmConstraints',false)`

Data Types: `logical`

**`EstimateEm` — Estimate open circuit voltage**
`true` (default) | `false`

Use this value to specify if the method estimates the open circuit voltage, *Em*. Use the default setting, `true`, unless you have already defined the *Em* values from outside analysis.

Example: `eestimateInitialEmR0(psObj,'EstimateEm',false)`

Data Types: `logical`

**`EstimateR0` — Estimate series resistance**
`true` (default) | `false`

Use this value to specify if the method estimates the series resistance, *R0*. Use the default setting, `true`, unless you have already defined the *R0* values from outside analysis.

Example: `estimateInitialEmR0(psObj,'EstimateR0',false)`

Data Types: `logical`

## See Also

`Battery.PulseSequence`

**Topics**
“Generate Parameter Data for Datasheet Battery Block”
“Generate Parameter Data for Equivalent Circuit Battery Block”

**Introduced in R2016b**

# estimateInitialEmRx

Estimate open circuit voltage and RC pair resistance

## Syntax

```
estimateInitialEmRx(psObj)
estimateInitialEmRx(psObj,Name,Value)
```

## Description

`estimateInitialEmRx(psObj)` estimates the open circuit voltage, *Em*, and RC pair resistance, *Ex*, for the `Battery.PulseSequence` object data. For the estimation, the method solves a linear system of equations throughout the pulse sequence. The method stores the results in a `Battery.Parameters` object.

`estimateInitialEmRx(psObj,Name,Value)` estimates the open circuit voltage, *Em*, and RC pair resistance, *Ex*, for the `Battery.PulseSequence` object data with additional options specified by one or more `Name,Value` pair arguments.

## Input Arguments

### `psObj` — Instance of `Battery.PulseSequence` class

`Battery.PulseSequence` object

`Battery.PulseSequence` object for the pulse sequence that you want to analyze.

#### Name-Value Pair Arguments

Specify optional comma-separated pairs of `Name,Value` arguments. `Name` is the argument name and `Value` is the corresponding value. `Name` must appear inside quotes. You can specify several name and value pair arguments in any order as `Name1,Value1,...,NameN,ValueN`.

Example:
```
estimateInitialEmRx(psObj,'IgnoreRelaxation',false,'ShowPlots',true,'ShowBefo
rePlots',true,'PlotDelay',0.5,'EstimateEm',true)
```

### `EstimateEm` — Estimate voltage

`true` (default) | `false`

Use this value to specify if the method estimates the open circuit voltage, *Em*.

Example: `estimateInitialEmRx(psObj,'EstimateEm',false)`

Data Types: `logical`

### `RetainEm` — Retain voltage estimate

`true` (default) | `false`

Use this value to specify if the method retains the open circuit voltage, *Em*, estimate. Set to `true` if you want the method to use an external open circuit voltage to state of charge (SOC) relationship. If `EstimateEm` is `false`, this option does not apply.

Example: `estimateInitialEmRx(psObj,'RetainEm',false)`

Data Types: `logical`

**`EstimateR0` — Estimate series resistance**
`true` (default) | `false`

Use this value to specify if the method estimates the series resistance, *Ro*.

Example: `estimateInitialEmRx(psObj,'EstimateR0',false)`

Data Types: `logical`

**`RetainR0` — Retain series resistance**
`true` (default) | `false`

Use this value to specify if the method retains the identified series resistance, *Ro*, estimate. Set to `true` if you want the method to use an existing series resistance to state of charge (SOC) relationship. If `EstimateEm` is `false`, this option does not apply.

Example: `estimateInitialEmRx(psObj,'RetainR0',false)`

Data Types: `logical`

**`ShowPlots` — Show estimation plots**
`false` (default) | `true`

Use this value to specify if the method shows plots during each estimation step.

Example: `estimateInitialEmRx(psObj,'ShowPlots',true)`

Data Types: `logical`

**`ShowBeforePlots` — Show before estimation plots**
`false` (default) | `true`

Use this value to specify if the method shows before plots during each estimation step. If `ShowPlots` is `false`, this option does not apply.

Example: `estimateInitialEmRx(psObj,'ShowBeforePlots',true)`

Data Types: `logical`

**`PlotDelay` — Plot delay**
`0.0` (default) | `scalar`

Use this value to specify the time delay after showing the plots, in s.

Example: `estimateInitialEmRx(psObj,'PlotDelay',0.1)`

Data Types: `double`

**`IgnoreRelaxation` — Estimate series resistance**
`false` (default) | `true`

Use this value to specify if the method completely ignores the relaxation and fits only the main pulse.

Example: `estimateInitialEmRx(psObj,'IgnoreRelaxation',true)`

Data Types: `logical`

## See Also

`Battery.PulseSequence`

**Topics**

“Generate Parameter Data for Datasheet Battery Block”
“Generate Parameter Data for Equivalent Circuit Battery Block”

**Introduced in R2016b**

# estimateInitialTau

Estimate RC pair time constant

## Syntax

```
estimateInitialTau(psObj)
estimateInitialTau(psObj,Name,Value)
```

## Description

`estimateInitialTau(psObj)` estimates the RC pair time constant, *Tau* for the `Battery.PulseSequence` object data. For the estimation, the method fits the relaxation curve for each pulse. The method stores the results in an `Battery.Parameters` object.

`estimateInitialTau(psObj,Name,Value)` estimates the RC pair time constant, *Tau* for the `Battery.PulseSequence` object data with additional options specified by one or more `Name,Value` pair arguments.

## Input Arguments

### `psObj` — Instance of `Battery.PulseSequence` class

`Battery.PulseSequence` object

`Battery.PulseSequence` object for the pulse sequence that you want to analyze.

#### Name-Value Pair Arguments

Specify optional comma-separated pairs of `Name,Value` arguments. `Name` is the argument name and `Value` is the corresponding value. `Name` must appear inside quotes. You can specify several name and value pair arguments in any order as `Name1,Value1,...,NameN,ValueN`.

Example:
`estimateInitialTau(psObj,'UpdateEndingEm',false,'ShowPlots',true,'ReusePlotFigure',true,'UseLoadData',false,'PlotDelay',0.5)`

#### `ShowPlots` — Show estimation plots

`false` (default) | `true`

Use this value to specify if the method shows plots during each estimation step.

Example: `estimateInitialTau(psObj,'ShowPlots',true)`

Data Types: `logical`

#### `PlotDelay` — Plot delay

`0.0` (default) | `scalar`

Use this value to specify the time delay after showing the plots, in s.

Example: `estimateInitialTau(psObj,'PlotDelay',0.5)`

Data Types: `double`

**`ReusePlotFigure` — Reuse plots**
`true` (default) | `false`

Use this value to specify if the method reuses the same plot figure. If `false`, the estimation plots are in separate figure windows. If `ShowPlots` is `false`, the option does not apply.

Example: `estimateInitialTau(psObj,'ReusePlotFigure',true)`

Data Types: `logical`

**`UpdateEndingEm` — Update voltage estimate**
`false` (default) | `true`

Use this value to specify if the method updates the open circuit voltage estimate at the end of the relaxation, based on the curve fits.

Example: `estimateInitialTau(psObj,'UpdateEndingEm',true)`

Data Types: `logical`

**`UseLoadData` — Plot delay**
`false` (default) | `true`

Use this value to specify if the method uses the pulse load data, instead of pulse relaxation data, to estimate the time constant, *Tau*. By default, the setting is `false`, and the method uses the pulse relaxation to estimate the time constant.

Example: `estimateInitialTau(psObj,'UseLoadData',true)`

Data Types: `logical`

## See Also

`Battery.PulseSequence`

**Topics**
“Generate Parameter Data for Datasheet Battery Block”
“Generate Parameter Data for Equivalent Circuit Battery Block”

**Introduced in R2016b**

# estimateParameters

Estimate parameters

## Syntax

```
estimateParameters(psObj)
estimateParameters(psObj,Name,Value)
```

## Description

`estimateParameters(psObj)` estimates the parameters in the `Battery.Parameters` object. The method stores the results in an `Battery.Parameters` object.

`estimateParameters(psObj,Name,Value)` estimates the parameters in the `Battery.Parameters` object data with additional options specified by one or more `Name,Value` pair arguments.

To use the `Battery.PulseSequence` object and methods, you need these products:

- Powertrain Blockset
- Curve Fitting Toolbox
- Optimization Toolbox
- Parallel Computing Toolbox
- Simulink Design Optimization

## Input Arguments

### `psObj` — Instance of `Battery.PulseSequence` class

`Battery.PulseSequence` object

`Battery.PulseSequence` object for the pulse sequence that you want to analyze.

#### Name-Value Pair Arguments

Specify optional comma-separated pairs of `Name,Value` arguments. `Name` is the argument name and `Value` is the corresponding value. `Name` must appear inside quotes. You can specify several name and value pair arguments in any order as `Name1,Value1,...,NameN,ValueN`.

Example:
`estimateParameters(psObj,'CarryParamToNextPulse',true,'ShowPlots',true,'EstimateEm',true,'RetainEm',true,'EstimateR0',true,'RetainR0',true)`

### `CarryParamsToNextPulse` — Use results for next SOC

`false` (default) | `true`

Use this value to specify if the method uses the identified current pulse final state of charge (SOC) parameter values as the initial estimate for the parameter values at the next SOC.

Example: `estimateParameters(psObj,'CarryParamsToNextPulse',true)`

Data Types: `logical`

**`EstimateEm` — Estimate voltage**
`true` (default) | `false`

Use this value to specify if the method estimates the open circuit voltage, *Em*.

Example: `estimateParameters(psObj,'EstimateEm',false)`

Data Types: `logical`

**`RetainEm` — Retain voltage estimate**
`true` (default) | `false`

Use this value to specify if the method retains the identified open circuit voltage, *Em*, estimate. If `EstimateEm` is `false`, this option does to apply.

Example: `estimateParameters(psObj,'RetainEm',false)`

Data Types: `logical`

**`EstimateR0` — Estimate series resistance**
`true` (default) | `false`

Use this value to specify if the method estimates the series resistance, *R0*.

Example: `estimateParameters(psObj,'EstimateR0',false)`

Data Types: `logical`

**`RetainR0` — Retain series resistance**
`true` (default) | `false`

Use this value to specify if the method retains the series resistance, *Ro*, estimate. If `EstimateR0` is `false`, this option does to apply.

Example: `estimateParameters(psObj,'RetainR0',false)`

Data Types: `logical`

**`RelConstrRx` — Rx deviation**
`inf` (default) | `scalar`

Use this value to specify the RC pair resistance, *Rx*, deviation from previous pulse values, in percentage.

Example: `estimateParameters(psObj,'RelConstrRx',100)`

Data Types: `double`

**`RelConstrTx` — Tx deviation**
`inf` (default) | `scalar`

Use this value to specify the time constant deviation from previous pulse values, in percentage.

Example: `estimateParameters(psObj,'RelConstrTx',100)`

Data Types: `double`

**`SDOOptimizeOptions` — Specify optimization options**
'`Method`' is `lsqnonlin` and '`UseParallel`' is `true` (default)

Use this value to specify the `sdo.OptimizeOptions` object options. For example:

```
SDOOptimizeOptions = sdo.OptimizeOptions(...
    'OptimizedModel',psObj.ModelName,...
    'Method','lsqnonlin',...
    'UseParallel','always')
```

**`ShowPlots` — Show estimation plots**
`false` (default) | `true`

Use this value to specify if the method shows plots during each estimation step.

Example: `estimateParameters(psObj,'ShowPlots',true)`

Data Types: `logical`

**`ReusePlotFigure` — Reuse plots**
`true` (default) | `false`

Use this value to specify if the method reuses the same plot figure. If `false`, the estimation plots are in separate figure windows. If `ShowPlots` is `false`, the option does not apply.

Example: `estimateParameters(psObj,'ReusePlotFigure',true)`

Data Types: `logical`

**`PlotDelay` — Plot delay**
`5.0` (default) | `scalar`

Use this value to specify the time delay after showing the plots, in s.

Example: `estimateParameters(psObj,'PlotDelay',0.1)`

Data Types: `double`

**`PulseNumbers` — Pulse numbers**
`1` (default) | `scalar`

Use this value to specify the pulse numbers to estimate. The default value, `1`, is set to estimate all the pulses.

Data Types: `uint32`

## See Also

`Battery.PulseSequence` | `sdo.OptimizeOptions`

**Topics**
“Generate Parameter Data for Datasheet Battery Block”
“Generate Parameter Data for Equivalent Circuit Battery Block”

**Introduced in R2016b**

# getSocIdxForPulses

Return state of charge index for pulses

## Syntax

```
idx=getSocIdxForPulses(psObj,pulseList)
```

## Description

`idx=getSocIdxForPulses(psObj,pulseList)` returns the row vector index of the state of charge (SOC) lookup table breakpoints.

## Input Arguments

### `psObj` — Instance of `Battery.PulseSequence` class

`Battery.PulseSequence` object

`Battery.PulseSequence` object for the pulse sequence that you want to analyze.

### `pulseList` — Index of pulses

`1:NumPulses` (default)

Index of pulses. For example, `1:10`.

Data Types: `int16`

## Output Arguments

### `idx` — Indices into SOC lookup table

1-by-`NumPulses` array

Indices into SOC lookup table.

Data Types: `int16`

## See Also

`Battery.PulseSequence`

**Topics**

“Generate Parameter Data for Datasheet Battery Block”
“Generate Parameter Data for Equivalent Circuit Battery Block”

**Introduced in R2016b**

# plot

Plot pulse sequence data

## Syntax

```
plot_handle = plot(psObj)
```

## Description

`plot_handle = plot(psObj)` plots the data from a `Battery.PulseSequence` object.

## Input Arguments

### `psObj` — Instance of `Battery.PulseSequence` class

`Battery.PulseSequence` object

`Battery.PulseSequence` object for the pulse sequence that you want to analyze.

## Output Arguments

### `plot_handle` — Plot handle

object handle

Handles to plot objects.

Data Types: `function_handle`

## See Also

`Battery.PulseSequence`

**Topics**

“Generate Parameter Data for Datasheet Battery Block”
“Generate Parameter Data for Equivalent Circuit Battery Block”

**Introduced in R2016b**

# plotIdentifiedPulses

Plot identified pulses

## Syntax

```
plot_handle = plotIdentifiedPulses(psObj)
```

## Description

`plot_handle = plotIdentifiedPulses(psObj)` plots identified pulses from a `Battery.PulseSequence` object.

## Input Arguments

### `psObj` — Instance of `Battery.PulseSequence` class

`Battery.PulseSequence` object

`Battery.PulseSequence` object for the pulse sequence that you want to analyze.

## Output Arguments

### `plot_handle` — Plot handle

object handle

Handles to plot objects.

Data Types: `function_handle`

## See Also

`Battery.PulseSequence`

**Topics**

“Generate Parameter Data for Datasheet Battery Block”
“Generate Parameter Data for Equivalent Circuit Battery Block”

**Introduced in R2016b**

# plotLatestParameters

Plot latest pulse sequence parameters

## Syntax

```
plot_handle = plotLatestParameters(psObj)
```

## Description

`plot_handle = plotLatestParameters(psObj)` plots the latest pulse sequence parameters from a `Battery.PulseSequence` object.

## Input Arguments

### `psObj` — Instance of `Battery.PulseSequence` class

`Battery.PulseSequence` object

`Battery.PulseSequence` object for the pulse sequence that you want to analyze.

## Output Arguments

### `plot_handle` — Plot handle

object handle

Handles to plot objects.

Data Types: `function_handle`

## See Also

`Battery.PulseSequence`

**Topics**

“Generate Parameter Data for Datasheet Battery Block”
“Generate Parameter Data for Equivalent Circuit Battery Block”

**Introduced in R2016b**

# plotSimulationResults

Plot pulse sequence simulation results

## Syntax

```
plot_handle=plotSimulationResults(psObj)
plot_handle=plotSimulationResults(psObj,param)
```

## Description

`plot_handle=plotSimulationResults(psObj)` plots the simulation results of the pulse sequence based on the current parameter values.

`plot_handle=plotSimulationResults(psObj,param)` plots the simulation results of the pulse sequence based on the parameter values specified by the `Battery.Parameter` object.

## Input Arguments

### `psObj` — Instance of `Battery.PulseSequence` class

`Battery.PulseSequence` object

`Battery.PulseSequence` object for the pulse sequence that you want to analyze.

### `param` — Instance of `Battery.Parameter` class

`Battery.Parameter` object

`Battery.Parameter` object for the parameters that you want to analyze.

## Output Arguments

### `plot_handle` — Plot handle

object handle

Handles to plot objects.

Data Types: `function_handle`

## See Also

`Battery.PulseSequence`

**Topics**

“Generate Parameter Data for Datasheet Battery Block”
“Generate Parameter Data for Equivalent Circuit Battery Block”

**Introduced in R2016b**

# populatePulseParameters

Populate pulse parameters

## Syntax

```
populatePulseParameters(psObj)
```

## Description

`populatePulseParameters(psObj)` populates parameters in the `Battery.PulseSequence` object based on the series of pulse objects. If the pulse objects are new, updated, or filtered, `populatePulseParameters` updates the identified pulse indices, SOC breakpoints, and parameters objects in `Battery.PulseSequence`.

## Input Arguments

### psObj — Instance of Battery.PulseSequence class

`Battery.PulseSequence` object

`Battery.PulseSequence` object for the pulse sequence that you want to analyze.

## See Also

`Battery.PulseSequence`

#### Topics

“Generate Parameter Data for Datasheet Battery Block”
“Generate Parameter Data for Equivalent Circuit Battery Block”

#### Introduced in R2016b

# removePulses

Remove pulses from sequence

## Syntax

```
removePulses(psObj,idxRemove)
```

## Description

`removePulses(psObj,idxRemove)` removes pulses from sequence specified by the `Battery.PulseSequence` object.

## Input Arguments

### `psObj` — Instance of `Battery.PulseSequence` class

`Battery.PulseSequence` object

`Battery.PulseSequence` object for the pulse sequence that you want to analyze.

### `idxRemove` — Index of pulse objects to remove

`1:NumPulses` (default)

Index of pulse objects to remove. For example, `1:10`.

Data Types: `int16`

## See Also

`Battery.PulseSequence`

**Topics**

“Generate Parameter Data for Datasheet Battery Block”
“Generate Parameter Data for Equivalent Circuit Battery Block”

**Introduced in R2016b**

# repairTimeVector

Repair time vector

## Syntax

```
repairTimeVector(psObj)
repairTimeVector(psObj,MinDeltaT)
```

## Description

`repairTimeVector(psObj)` repairs common problems with the experimental time vector on the `Battery.PulseSequence` object.

`repairTimeVector(psObj,MinDeltaT)` repairs common problems with the experimental time vector on the `Battery.PulseSequence` object using a minimum time difference.

## Input Arguments

**`psObj` — Instance of `Battery.PulseSequence` class**
`Battery.PulseSequence` object

`Battery.PulseSequence` object for the pulse sequence that you want to analyze.

**`MinDeltaT` — Minimum time difference**
scalar

Index of pulse objects to remove. For example, `1:10`.

Data Types: `double`

## See Also

`Battery.PulseSequence`

**Topics**
“Generate Parameter Data for Datasheet Battery Block”
“Generate Parameter Data for Equivalent Circuit Battery Block”

**Introduced in R2016b**

# Battery.Pulse

Define a single pulse event

## Description

Use the `Battery.Pulse` object to define a single experimental pulse event. To create a pulse object, use the `Battery.PulseSequence` object function `createPulses`.

To use the `Battery.Pulse` object and methods, you need these products:

- Powertrain Blockset
- Curve Fitting Toolbox
- Optimization Toolbox
- Parallel Computing Toolbox
- Simulink Design Optimization

## Creation

### Syntax

```
pulseObj = Battery.Pulse(Battery.PulseSequence)
```

#### Description

MATLAB creates a `pulseObj = Battery.Pulse(Battery.PulseSequence)` object that defines a single pulse event.

### Properties

#### `Data` — Raw data

1-by-5 array

An 1-by-5 array of pulse event data.

| Array Element | Description | Unit |
|---|---|---|
| `Data(1,1)` | Time | s |
| `Data(1,2)` | Voltage | V |
| `Data(1,3)` | Current | A |
| `Data(1,4)` | Charge | A·s |
| `Data(1,5)` | State of charge (SOC) | Dimensionless |

Data Types: `double`

**`InitialCapVoltage` — Initial capacitor voltage**
array

Initial voltage of each capacitor during a pulse event, in V. Property set by the `Battery.PulseSequence` object function `estimateParameters`, based on the simulated end voltage or a prior pulse.

Data Types: `double`

**`InitialChargeDeficit` — Initial charge deficit**
`0.0` (default) | scalar

Initial charge deficit at start of pulse event, in A·s. Property set by the `Battery.PulseSequence` object function `createPulses` when the function creates the series of `Battery.Pulse` objects.

Example: `0.0`

Data Types: `double`

**`idxLoad` — Indices to load data**
`[1 0]` (default)

Indices to load data where the pulse event load begins and ends. Property set by the `Battery.PulseSequence` object function `createPulses` when the function creates the series of `Battery.Pulse` objects.

Data Types: `int16`

**`idxRelax` — Indices to relaxation data**
`[1 0]` (default)

Indices to relaxation data where the pulse event relaxation begins and ends. Property set by the `Battery.PulseSequence` object function `createPulses` when the function creates the series of `Battery.Pulse` objects.

Data Types: `int16`

**`idxPulseSequence` — Index to first pulse event data**
`[]` (default)

Index to first pulse event data point in the `Battery.PulseSequence` object data. Property set by the `Battery.PulseSequence` object function `createPulses` when the function creates the series of `Battery.Pulse` objects.

Data Types: `int16`

**`IsDischarge` — Discharge pulse**
`true` (default)

Use this value to specify if pulse is a discharge pulse event. Property set by the `Battery.PulseSequence` object function `createPulses` when the function creates the series of `Battery.Pulse` objects.

Data Types: `logical`

**`Parameters` — `Battery.Parameters` object properties**
`0`-by-1 array

`Battery.Parameters` object containing the most recently determined battery equivalent circuit parameters. Property set by the `Battery.PulseSequence` object function `createPulses` when the function creates the series of `Battery.Pulse` objects.

Data Types: `function_handle`

**`ParametersHistory` — `Battery.ParametersHistory` object properties**
`0`-by-`1` array

`Battery.ParametersHistory` object array containing the history of the battery equivalent circuit parameters through different estimation steps. The last element is the most recent parameter set.

Data Types: `function_handle`

## Object Functions

| | |
|---|---|
| plot | Plot pulse event data |
| getLoadData | Retrieve experimental data during load phase of pulse |
| getRelaxationData | Retrieve experimental data during relaxation phase of pulse |
| getTransitionData | Retrieve experimental data during transition phase of pulse |

## Examples

### Create `Battery.Pulse` Object

This example shows how to create a `Battery.Pulse` object.

```
pulseObj = Battery.Pulse(psObj);
```

## See Also

`Battery.MetaData` | `Battery.Parameters` | `Battery.PulseSequence` | `sdo.OptimizeOptions`

**Topics**
“Generate Parameter Data for Datasheet Battery Block”
“Generate Parameter Data for Equivalent Circuit Battery Block”

**Introduced in R2016b**

# plot

Plot pulse event data

## Syntax

```
plot_handle=plot(pulseObj)
```

## Description

`plot_handle=plot(pulseObj)` plots the data from a `Battery.Pulse` object.

## Input Arguments

### `pulseObj` — Instance of `Battery.Pulse` class

`Battery.Pulse` object

`Battery.Pulse` object for the pulse event that you want to analyze.

## Output Arguments

### `plot_handle` — Plot handle

object handle

Handles to plot objects.

Data Types: `function_handle`

## See Also

`Battery.Pulse`

**Topics**

“Generate Parameter Data for Datasheet Battery Block”
“Generate Parameter Data for Equivalent Circuit Battery Block”

**Introduced in R2016b**

# getLoadData

Retrieve experimental data during load phase of pulse

## Syntax

```
LoadData = plot(pulseObj,Buffer)
```

## Description

`LoadData = plot(pulseObj,Buffer)` retrieves the experimental data from a `Battery.Pulse` object during the load phase of a pulse.

## Input Arguments

### `pulseObj` — Instance of `Battery.Pulse` class

`Battery.Pulse` object

`Battery.Pulse` object for the pulse event that you want to analyze.

### `Buffer` — Number of samples

vector

Number of buffer samples before and after the load data, in the form `[BeforeBufferSize,AfterBufferSize]`. Use the buffer to ensure that the estimation has sufficient data before and after a transition.

## Output Arguments

### `LoadData` — Load data

array

Load data during pulse event.

Data Types: `double`

## See Also

`Battery.Pulse`

### Topics

“Generate Parameter Data for Datasheet Battery Block”
“Generate Parameter Data for Equivalent Circuit Battery Block”

### Introduced in R2016b

# getRelaxationData

Retrieve experimental data during relaxation phase of pulse

## Syntax

```
RelaxationData = plot(pulseObj,Buffer)
```

## Description

`RelaxationData = plot(pulseObj,Buffer)` retrieves the experimental data from a `Battery.Pulse` object during the relaxation phase of a pulse.

## Input Arguments

### `pulseObj` — Instance of `Battery.Pulse` class

`Battery.Pulse` object

`Battery.Pulse` object for the pulse event that you want to analyze.

### `Buffer` — Number of samples

vector

Number of buffer samples before and after the load data, in the form `[BeforeBufferSize,AfterBufferSize]`. Use the buffer to ensure that the estimation has sufficient data before and after a transition.

## Output Arguments

### `RelaxationData` — Relaxation data

array

Relaxation data during pulse event.

Data Types: `double`

## See Also

`Battery.Pulse`

### Topics

“Generate Parameter Data for Datasheet Battery Block”
“Generate Parameter Data for Equivalent Circuit Battery Block”

### Introduced in R2016b

# getTransitionData

Retrieve experimental data during transition phase of pulse

## Syntax

```
[TransitionDataBefore,TransitionDataAfter]=plot(pulseObj,idx)
[TransitionDataBefore,TransitionDataAfter]=plot(pulseObj,idx,Buffer)
```

## Description

`[TransitionDataBefore,TransitionDataAfter]=plot(pulseObj,idx)` retrieves the transition data from a `Battery.Pulse` object during the transition phase of a pulse.

`[TransitionDataBefore,TransitionDataAfter]=plot(pulseObj,idx,Buffer)` retrieves buffered experimental data from a `Battery.Pulse` object during the transition phase of a pulse.

## Input Arguments

### `pulseObj` — Instance of `Battery.Pulse` class

`Battery.Pulse` object

`Battery.Pulse` object for the pulse event that you want to analyze.

### `idx` — Transition data index

scalar

Index of transition data.

Data Types: `int16`

### `Buffer` — Number of samples

vector

Number of buffer samples before and after the load data, in the form `[BeforeBufferSize,AfterBufferSize]`. Use the buffer to ensure that the estimation has sufficient data before and after a transition.

## Output Arguments

### `TransitionDataBefore` — Data before transition

array

Data before transition during pulse event.

Data Types: `double`

### `TransitionDataAfter` — Data after transition

array

Data after transition during pulse event.

Data Types: `double`

## See Also

`Battery.Pulse`

**Topics**

“Generate Parameter Data for Datasheet Battery Block”
“Generate Parameter Data for Equivalent Circuit Battery Block”

**Introduced in R2016b**

# Battery.Parameters

Define battery equivalent circuit parameters

## Description

Use the `Battery.Parameters` object to define the battery equivalent circuit parameters. `Battery.Parameters` objects are contained in the `Battery.PulseSequence` and `Battery.Pulse` objects. The pulse sequence estimation sets some of the `Battery.Parameters` properties. You can override the properties by manually setting the properties. The number of pulses, *N*, in the dataset determines the length of each array.

## Creation

### Syntax

```
paramObj = Battery.Parameters
```

#### Description

MATLAB creates a `paramObj = Battery.Parameters` object that defines the battery equivalent circuit parameters.

### Properties

**`SOC` — State of charge breakpoints**
1-by-11 array (default)

A 1-by-`N` array of the state of charge (SOC) breakpoints.

Data Types: `double`

**`Em` — Open circuit voltage**
1-by-11 array (default)

A 1-by-`N` array of the open circuit voltage, in V.

Data Types: `double`

**`EmMin` — Minimum open circuit voltage**
1-by-11 array (default)

A 1-by-`N` array of the minimum open circuit voltage, in V.

Data Types: `double`

**`EmMax` — Maximum open circuit voltage**
1-by-11 array (default)

A 1-by-`N` array of the maximum open circuit voltage, in V.

Data Types: `double`

**`R0` — Terminal resistance**
1-by-11 array (default)

A 1-by-N array of the terminal resistance, in Ohms.

Data Types: `double`

**`R0Min` — Minimum terminal resistance**
1-by-11 array (default)

A 1-by-N array of the minimum terminal resistance, in Ohms.

Data Types: `double`

**`R0Max` — Maximum terminal resistance**
1-by-11 array (default)

A 1-by-N array of the maximum terminal resistance, in Ohms.

Data Types: `double`

**`Rx` — RC pair resistance**
3-by-11 array (default)

A 3-by-N array of the RC pair resistance, in Ohms.

Data Types: `double`

**`RxMin` — Minimum RC pair resistance**
3-by-11 array (default)

A 3-by-N array of the minimum RC pair resistance, in Ohms.

Data Types: `double`

**`RxMax` — Maximum RC pair resistance**
3-by-11 array (default)

A 3-by-N array of the maximum RC pair resistance, in Ohms.

Data Types: `double`

**`Tx` — RC pair time constant**
3-by-11 array (default)

A 3-by-N array of the RC pair time constant, in s.

Data Types: `double`

**`TxMin` — Minimum RC pair time constant**
3-by-11 array (default)

A 3-by-N array of the minimum RC pair time constant, in s.

Data Types: `double`

**`TxMax` — Maximum RC pair time constant**
3-by-11 array (default)

A 3-by-N array of the maximum RC pair time constant, in s.

Data Types: double

## Object Functions

| | |
|---|---|
| lookupSocFromVoltage | Determine SOC from voltage |
| plot | Plot battery parameter data |

## Examples

### Create Battery.Parameters Object

This example shows how to create a Battery.Parameters object.

Create a Battery.Parameters object.

```
paramObj=Battery.Parameters;
```

## See Also

Battery.MetaData | Battery.Pulse | Battery.PulseSequence | sdo.OptimizeOptions

### Topics

“Generate Parameter Data for Datasheet Battery Block”
“Generate Parameter Data for Equivalent Circuit Battery Block”

### Introduced in R2016b

# lookupSocFromVoltage

Determine SOC from voltage

## Syntax

```
SOC=lookupSocFromVoltage(paramObj,Voltage)
```

## Description

`SOC=lookupSocFromVoltage(paramObj,Voltage)` calculates the state of charge (SOC) from the voltage for a given open-circuit voltage. Use `lookupSocFromVoltage` after you know the open-circuit voltage, *Em*, value.

## Input Arguments

### `paramObj` — Instance of `Battery.Parameters` class

`Battery.Parameters` object

`Battery.Parameters` object for the battery that you want to analyze.

### `Voltage` — Open circuit voltage

scalar

Open circuit voltage, in V.

Data Types: `char`

## Output Arguments

### `SOC` — State of charge

scalar

State of charge.

Data Types: `double`

## See Also

`Battery.Parameters`

**Topics**

“Generate Parameter Data for Datasheet Battery Block”
“Generate Parameter Data for Equivalent Circuit Battery Block”

**Introduced in R2016b**

# plot

Plot battery parameter data

## Syntax

```
plot_handle=plot(paramObj)
plot_handle=plot(paramObj,LegendNames)
```

## Description

`plot_handle=plot(paramObj)` plots the data from a `Battery.Parameters` object.

`plot_handle=plot(paramObj,LegendNames)` plots the data from a `Battery.Parameters` object with the legend names.

## Input Arguments

### `paramObj` — Instance of `Battery.Pulse` class

`Battery.Parameters` object

`Battery.Parameters` object for the battery that you want to analyze.

### `LegendNames` — Plot legends

character vector

Name of plot legends.

Data Types: `char`

## Output Arguments

### `plot_handle` — Plot handle

object handle

Handles to plot objects.

Data Types: `function_handle`

## See Also

`Battery.Parameters`

**Topics**

“Generate Parameter Data for Datasheet Battery Block”
“Generate Parameter Data for Equivalent Circuit Battery Block”

**Introduced in R2016b**

# Battery.MetaData

Define battery metadata

## Description

Use the `Battery.MetaData` object to define the battery metadata. A `Battery.PulseSequence` object contains the `Battery.MetaData` object. You must specify the metadata values.

## Creation

### Syntax

```
batmetaObj = Battery.MetaData
```

#### Description

MATLAB creates a `batmetaObj = Battery.MetaData` object that defines the battery metadata.

### Properties

**`BatteryId` — Battery identification**
character vector

Battery identification name.

Data Types: `double`

**`RatingAh` — Battery rating**
character vector

Battery rating.

Data Types: `char`

**`Name` — Dataset name**
character vector

Dataset name.

Data Types: `char`

**`Date` — Dataset date**
character vector

Dataset date.

Data Types: `char`

**`Source` — Dataset source**
character vector

Dataset source.

Data Types: `char`

**`TestType` — Experimental data type**
character vector

Test time, for example charge or discharge.

Data Types: `char`

**`TestCurrent` — Test current**
scalar

Test current, in A.

Data Types: `double`

**`TestTemperature` — Test temperature**
scalar

Test temperature, in C.

Data Types: `double`

## Examples

### Create `Battery.MetaData` Object and Set Properties

This example shows how to create a `Battery.MetaData` object and set properties.

Create a `Battery.MetaData` object.

```
batmetaObj=Battery.MetaData;
```

Set `Battery.MetaData` properties.

```
batmetaObj.BatteryId='myBatteryId';
batmetaObj.RatingAh='myRatingAh';
batmetaObj.Name='myName';
batmetaObj.Date='myDate';
batmetaObj.Source='mySource';
batmetaObj.TestType='Charge';
batmetaObj.TestCurrent=300;
batmetaObj.TestCurrent=120;
```

Display `Battery.MetaData` properties.

```
disp(batmetaObj)
```

## See Also

`Battery.Parameters` | `Battery.Pulse` | `Battery.PulseSequence`

**Topics**
“Generate Parameter Data for Datasheet Battery Block”
“Generate Parameter Data for Equivalent Circuit Battery Block”

**Introduced in R2016b**