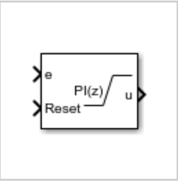
Discrete PI Controller with Integral Anti-Windup

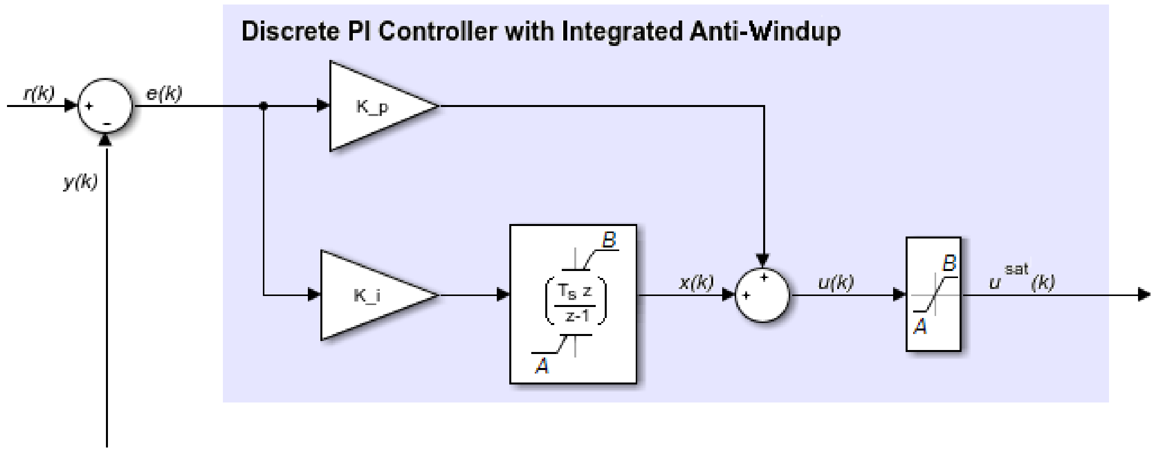
Discrete-time PI control with integral anti-windup

Library: Simscape / Electrical / Control / General Control



Description

The Discrete PI Controller with Integral Anti-Windup block implements discrete PI control with internal anti-windup. The figure shows the equivalent circuit for the controller with internal anti-windup.



Equations

The block calculates the control signal using the backward Euler discretization method:

$$u(k) = \mathrm{sat}\left(K_p\mathrm{e(k)} + \mathrm{sat}\left(K_i\frac{T_s z}{z-1}\mathrm{e(k), A, B}\right), A, B\right),$$

$$sat(x, A, B) = \min(\max(x, \mathrm{A}), B),$$

where:

- u is the control signal.
- K_p is the proportional gain coefficient.

- e is the error signal.
- K_i is the integral gain coefficient.
- T_s is the sampling period.
- A is the lower limit for saturation.
- B is the upper limit for saturation.

Ports

Input

collapse all

e — Error signal
scalar

Error signal, $e(k)$, obtained as the difference between the reference, $r(k)$, and measurement $y(k)$ signals.

Data Types: `single` | `double`

`Reset` — Integrator gain reset
scalar

External reset (rising edge) signal for the integrator.

Data Types: `single` | `double`

Output

collapse all

u — Control signal
scalar

Control signal, $u(k)$.

Data Types: `single` | `double`

Parameters

collapse all

`Proportional gain` — K_p
1 (default) | positive scalar

Proportional gain, K_p, of the PI controller.

`Integral gain` — K_i
`1` (default) | positive scalar

Integral gain, K_i, of the PI controller.

`Upper saturation limit` — B
`5` (default) | scalar greater than the value of the **Lower saturation limit** parameter

Upper limit, B, of the output for the PI controller.

`Lower saturation limit` — A
`-5` (default) | scalar

Upper limit, A, of the output for the PI controller.

`Integrator initial condition` — Initial integrator value
`0` (default) | scalar

Value of the integrator at simulation start time.

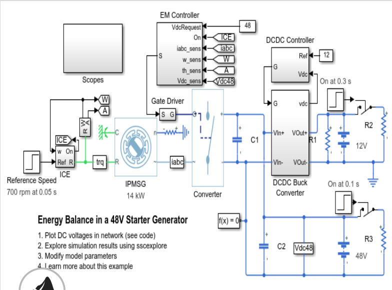
`Sample time (-1 for inherited)` — Block sample time
`-1` (default) | 0 | positive scalar

Time between consecutive block executions. During execution, the block produces outputs and, if appropriate, updates its internal state. For more information, see What Is Sample Time? (Simulink) and Specify Sample Time (Simulink).

For inherited discrete-time operation, specify `-1`. For discrete-time operation, specify a positive integer. For continuous-time operation, specify `0`.

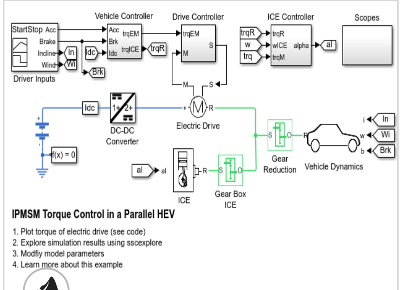
If this block is in a masked subsystem, or other variant subsystem that allows you to switch between continuous operation and discrete operation, promote the sample time parameter. Promoting the sample time parameter ensures correct switching between the continuous and discrete implementations of the block. For more information, see Promote Parameter to Mask (Simulink).

Model Examples



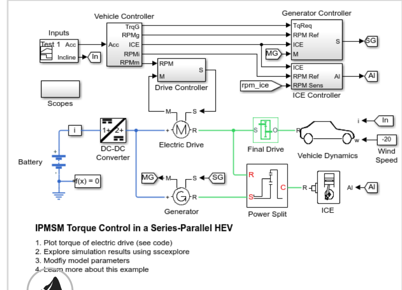
Energy Balance in a 48V Starter Generator

An interior permanent magnet synchronous machine (IPMSM) used as a starter/generator in a simplified 48V automotive system.



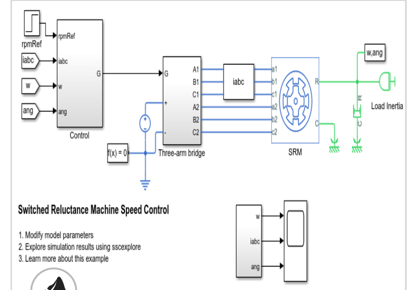
IPMSM Torque Control in a Parallel HEV

A simplified parallel hybrid electric vehicle (HEV). An interior permanent magnet synchronous machine (IPMSM) and an internal combustion



IPMSM Torque Control in a Series-Parallel HEV

A simplified series-parallel hybrid electric vehicle (HEV). An interior permanent magnet synchronous machine (IPMSM) and an internal



Switched Reluctance Machine Speed Control

Control the rotor speed in a switched reluctance machine (SRM) based electrical drive. A DC voltage source feeds the SRM through a controlled

References

[1] *IEEE Recommended Practice for Excitation System Models for Power System Stability Studies.* IEEE Std 421.5/D39. Piscataway, NJ: IEEE-SA, 2015.

Extended Capabilities

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

See Also

Blocks

Discrete PI Controller

Topics

Promote Parameter to Mask (Simulink)

Introduced in R2017b