

Simulation VIs and Functions

June 2008, 371014E-01

Owning Palette: [Control Design and Simulation VIs and Functions](#)

Installed With: Control Design and Simulation Module. This topic might not match its corresponding palette in LabVIEW depending on your operating system, licensed product(s), and target.

Use the Simulation VIs and functions to create [simulation applications](#) in LabVIEW.

The VIs and functions on this palette can return [general LabVIEW error codes](#) or [specific Simulation error codes](#). If you use the functions on this palette in a [Simulation Loop](#), LabVIEW sends any errors that these functions return to the **Error** output on the Output Node of the Simulation Loop.

Palette Object	Description
Simulation Loop	Executes the simulation diagram until the Simulation Loop reaches the simulation final time or until the Halt Simulation function stops the execution programmatically. You must place all Simulation functions within a Simulation Loop or in a simulation subsystem. You also can place simulation subsystems within a Simulation Loop or another simulation subsystem , or you can place simulation subsystems on a block diagram outside a Simulation Loop or run the simulation subsystems as stand-alone VIs .

Subpalette	Description
Continuous Linear Systems Functions	Use the Continuous Linear Systems functions to represent continuous linear systems of differential equations on the simulation diagram.
Discrete Linear Systems Functions	Use the Discrete Linear Systems functions to represent discrete linear systems of difference equations on the simulation diagram.

<u>Graph Utilities Functions</u>	Use the Graph Utilities functions to display the outputs of the simulation on a chart or graph.
<u>Lookup Tables Functions</u>	Use the Lookup Tables functions to locate a value from an existing data set. You specify this data set in the LUT Data parameter of each Lookup Tables function.
<u>Nonlinear Systems Functions</u>	Use the Nonlinear Systems functions to simulate nonlinear effects that are present in real-world systems.
<u>Optimal Design VIs</u>	Use the Optimal Design VIs to determine the optimal parameters for a system model given a cost function and set of constraints.
<u>Signal Arithmetic Functions</u>	Use the Signal Arithmetic functions to perform basic arithmetic operations on signals in a simulation system.
<u>Signal Generation Functions</u>	Use the Signal Generation functions to generate specific waveform patterns and to index signals into an array.
<u>Trim & Linearize VIs</u>	Use the Trim & Linearize VIs to trim and linearize a simulation subsystem.
<u>Utilities Functions</u>	Use the Utilities functions to perform various tasks such as creating a history of signal values and the times at which they were recorded, indexing a signal by simulation time, reporting simulation time, reporting simulation errors, stopping the simulation programmatically, and returning the simulation parameters.

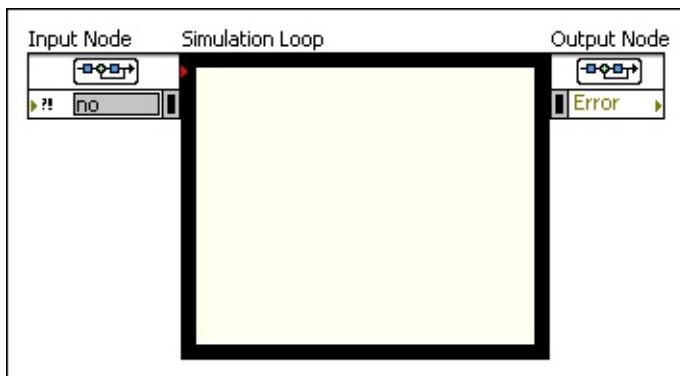
© 2004–2008 National Instruments Corporation. All rights reserved.

Simulation Loop

Owning Palette: [Simulation VIs and Functions](#)

Installed With: Control Design and Simulation Module

Executes the simulation diagram until the Simulation Loop reaches the simulation final time or until the [Halt Simulation](#) function stops the execution programmatically. You must place all [Simulation](#) functions within a Simulation Loop or in a simulation subsystem. You also can place simulation subsystems [within a Simulation Loop or another simulation subsystem](#), or you can place simulation subsystems [on a block diagram outside a Simulation Loop](#) or run the simulation subsystems as [stand-alone VIs](#).



- Place on the block diagram ■ Find on the **Functions** palette

The Simulation Loop has an Input Node and an Output Node. Use the Input Node to configure simulation parameters programmatically. You also can configure these parameters interactively using the [Configure Simulation Parameters](#) dialog box. Access this dialog box by double-clicking the Input Node or by right-clicking the border and selecting **Configure Simulation Parameters** from the shortcut menu.

-  **Note** Parameters that you programmatically configure using the Input Node override any settings that you make in the [Configure Simulation Parameters](#) dialog box.

The Simulation Loop has an **Error** input on the Input Node and an **Error** output on the Output Node. These error terminals send error information through the simulation diagram. If the **Error** input detects an error, the simulation diagram returns the error information in the **Error** output and does not execute the simulation. If an error occurs while the Simulation

Loop is executing, the simulation stops running and returns the error information in the **Error** output.

Continuous Linear Systems Functions

Owning Palette: [Simulation VIs and Functions](#)

Installed With: Control Design and Simulation Module. This topic might not match its corresponding palette in LabVIEW depending on your operating system, licensed product(s), and target.

Use the Continuous Linear Systems functions to represent continuous linear systems of differential equations on the simulation diagram.

The functions on this palette can return [general LabVIEW error codes](#) or [specific Simulation error codes](#). If you use the functions on this palette in a [Simulation Loop](#), LabVIEW sends any errors that these functions return to the **Error** output on the Output Node of the Simulation Loop.

Palette Object	Description
CD Continuous Observer	Implements an observer for a continuous linear time-invariant (LTI) state-space model .
CD Continuous Recursive Kalman Filter	Implements a Kalman filter for a continuous linear time-invariant (LTI) or linear time-variant (LTV) stochastic state-space model. This function calculates the Kalman filtered state estimates and outputs at time t .
Derivative	Calculates the derivative of a continuous signal.
Integrator	Integrates a continuous input signal using the ordinary differential equation (ODE) solver you specify for the simulation.
State-Space	Implements a system model in state-space form. You define the system model by specifying the input, output, state, and direct transmission matrices.
Transfer Function	Implements a system model in transfer function form. You define the system model by specifying the Numerator and Denominator of the transfer function equation.
Transport Delay	Delays the input signal by the amount of time you specify.
Zero-Pole-	Implements a system model in zero-pole-gain form. You

[Gain](#)

[define the system model](#) by specifying the **Zeros**, **Poles**, and **Gain** of the zero-pole-gain equation.

CD Continuous Observer Function

Owning Palette: [Continuous Linear Systems Functions](#)

Installed With: Control Design and Simulation Module

Implements an observer for a continuous [linear time-invariant \(LTI\) state-space model](#).

Refer to Chapter 15, *Estimating Model States*, of the [LabVIEW Control Design User Manual](#) for information about using a continuous observer.

[Details](#)

[Dialog Box Options](#)

[Block Diagram Inputs](#)

[Block Diagram Outputs](#)

■ Place on the block diagram ■ Find on the **Functions** palette

Dialog Box Options

Parameter	Description
Parameters	Lists all the parameters associated with this function. Select a parameter from this list to configure the parameter. When you select a parameter, the parameter and its associated Parameter source control appear in the Parameter Information section of the configuration dialog box.
Preview	Displays a graphical preview, if available, of the function output or configuration.
Parameter Information	Contains the parameters you can configure for this function. You must select a parameter from the Parameters list to make that parameter and its associated Parameter source control visible in the Parameter Information section of the configuration dialog box.
Parameter source	Specifies whether you configure this parameter using the Configuration Dialog Box or a Terminal on the simulation diagram. The default value is Configuration Dialog Box . If you select Terminal , LabVIEW displays an input for that parameter on the simulation diagram, and you can wire values to that input to configure this function programmatically. If you select Configuration Dialog Box , LabVIEW removes that input from the simulation diagram. You then must set the value for this parameter inside the configuration dialog box.
State-Space Model	Specifies the mathematical representation of and information about the system for which this function implements the observer. <ul style="list-style-type: none">• Model name—Specifies the name of the state-space model.• Sampling Time—Specifies the sampling time of the system model and determines whether the model represents a continuous-time or discrete-time system. If the model represents a continuous-time system, Sampling Time must equal zero. If the model represents a discrete-time system,

Sampling Time must be greater than zero and equal to the sampling rate, in seconds, of the discrete system. The default is 0.



Note If you use the inputs to create a continuous-time system, setting the **Sampling Time** to a value greater than zero does not yield the discrete-time equivalent of the system. You must use the [CD Convert Continuous Stochastic to Discrete](#) VI to convert the continuous-time system to the discrete-time equivalent of the system.

- **A**—Specifies the $n \times n$ state matrix of the given system.
- **B**—Specifies the $n \times m$ input matrix of the given system.
- **C**—Specifies the $r \times n$ output matrix of the given system.
- **D**—Specifies the $r \times m$ direct transmission matrix of the given system.
where m is the number of inputs
 n is the number of states
 r is the number of outputs

Observer Gain	Specifies the estimator gain matrix this function applies to the difference between the observed output and the estimated output, which is $y(t) - \hat{y}(t)$. You can use the CD Pole Placement VI or the CD Ackermann VI to calculate Observer Gain .
Initial state estimate $\hat{x}(t_0)$	Specifies the initial states from which this function begins estimating the model states. If you do not specify a value for this parameter, Initial state estimate $\hat{x}(t_0)$ is a vector of zeros.
Output $y(t)$	Specifies the measurements made on the State-Space Model .
Input $u(t)$	Specifies the control action this function applies to the model. If you specify a vector of zeros for Input $u(t)$, this

function does not apply any control action.

Block Diagram Inputs

Parameter	Description
Observer Gain	Specifies the estimator gain matrix this function applies to the difference between the observed output and the estimated output, which is $y(t) - \hat{y}(t)$. You can use the CD Pole Placement VI or the CD Ackermann VI to calculate Observer Gain .
Initial state estimate $\hat{x}(t_0)$	Specifies the initial states from which this function begins estimating the model states. If you do not specify a value for this parameter, Initial state estimate $\hat{x}(t_0)$ is a vector of zeros.
Output $y(t)$	Specifies the measurements made on the State-Space Model .
Input $u(t)$	Specifies the control action this function applies to the model. If you specify a vector of zeros for Input $u(t)$, this function does not apply any control action.
State-Space Model	Specifies the mathematical representation of and information about the system for which this function calculates the estimated states.

Block Diagram Outputs

Parameter	Description
Estimated Output $y\hat{a}(t)$	Returns the estimated model output at time t .
Estimated State $x\hat{a}(t)$	Returns the estimated model states at time t .

CD Continuous Observer Details

This function solves the following equations:

$$\dot{\hat{x}}(t) = A\hat{x}(t) + Bu(t) + L[y(t) - \hat{y}(t)]$$

$$\hat{y}(t) = C\hat{x}(t) + Du(t)$$

The output **Estimated State xhat(t)** is the result of integrating $\dot{\hat{x}}(t)$ by using the ordinary differential equation (ODE) solver and time step information you specify for the [Simulation Loop](#).

CD Continuous Recursive Kalman Filter Function

Owning Palette: [Continuous Linear Systems Functions](#)

Installed With: Control Design and Simulation Module

Implements a Kalman filter for a continuous linear time-invariant (LTI) or linear time-variant (LTV) stochastic state-space model. This function calculates the Kalman filtered state estimates and outputs at time t .

[Details](#)

[Dialog Box Options](#)

[Block Diagram Inputs](#)

[Block Diagram Outputs](#)

■ Place on the block diagram ■ Find on the **Functions** palette

Dialog Box Options

Parameter	Description
Parameters	Lists all the parameters associated with this function. Select a parameter from this list to configure the parameter. When you select a parameter, the parameter and its associated Parameter source control appear in the Parameter Information section of the configuration dialog box.
Preview	Displays a graphical preview, if available, of the function output or configuration.
Parameter Information	Contains the parameters you can configure for this function. You must select a parameter from the Parameters list to make that parameter and its associated Parameter source control visible in the Parameter Information section of the configuration dialog box.
Parameter source	Specifies whether you configure this parameter using the Configuration Dialog Box or a Terminal on the simulation diagram. The default value is Configuration Dialog Box . If you select Terminal , LabVIEW displays an input for that parameter on the simulation diagram, and you can wire values to that input to configure this function programmatically. If you select Configuration Dialog Box , LabVIEW removes that input from the simulation diagram. You then must set the value for this parameter inside the configuration dialog box.
Stochastic State-Space Model	Specifies a mathematical representation of a stochastic system. You can construct a stochastic state-space model using the CD Construct Stochastic Model VI . <ul style="list-style-type: none">• A—Specifies the system matrix that describes the dynamics of the states of the system.• B—Specifies the input matrix that relates the inputs to the states.• G—Specifies the matrix that relates the process noise vector to the model states.• C—Specifies the output matrix that relates the

	<p>outputs to the states.</p> <ul style="list-style-type: none"> • D—Specifies the transmission matrix that relates the inputs to the outputs. • H—Specifies the matrix that relates the process noise vector to the model outputs. • Sampling Time (s)—Specifies the sampling time of the system model and determines whether the model represents a continuous-time or discrete-time system. If the model represents a continuous-time system, Sampling Time (s) must equal zero. If the model represents a discrete-time system, Sampling Time (s) must be greater than zero and equal to the sampling rate, in seconds, of the discrete system. The default is 0. <p> Note If you use the inputs to create a continuous-time system, setting the Sampling Time (s) to a value greater than zero does not yield the discrete-time equivalent of the system. You must use the CD Convert Continuous Stochastic to Discrete VI to convert the continuous-time system to the discrete-time equivalent of the system.</p>
Second-Order Statistics Noise Model	<p>Specifies a mathematical representation of the noise model of a stochastic state-space model. You can create a noise model using the CD Construct Noise Model VI.</p> <ul style="list-style-type: none"> • E{w}—Specifies the expected value or mean of the process noise vector. The default is 0. • Q—Specifies the covariance matrix of the process noise vector. The default is 0.01. • E{v}—Specifies the expected value or mean of the measurement noise vector. The default is 0. • R—Specifies the covariance matrix of the measurement noise vector. The default is 0.1. • N—Specifies the cross-covariance matrix between the process noise vector and the measurement

	noise vector. If these noise vectors are uncorrelated, \mathbf{N} is a matrix of zeros. The default is 0.
Initial State Estimate $\mathbf{xhat(t0)}$	Specifies the initial states from which this function begins estimating the model states. If you do not specify a value for this parameter, Initial State Estimate $\mathbf{xhat(t0)}$ is a vector of zeros.
Input $\mathbf{u(t)}$	Specifies the control action this function applies to the model. If you specify a vector of zeros for Input $\mathbf{u(t)}$, this function does not apply any control action.
Output $\mathbf{y(t)}$	Specifies the measurement made on the stochastic state-space model.
Initial Estimation Error Covariance $\mathbf{P(t0)}$	Specifies the initial covariance matrix of the estimation error. If you do not specify a value for this parameter, Initial Estimation Error Covariance $\mathbf{P(t0)}$ is a matrix of zeros.

Block Diagram Inputs

Parameter	Description
Stochastic State-Space Model	<p>Specifies a mathematical representation of a stochastic system. You can construct a stochastic state-space model using the CD Construct Stochastic Model VI.</p> <ul style="list-style-type: none">• A—Specifies the system matrix that describes the dynamics of the states of the system.• B—Specifies the input matrix that relates the inputs to the states.• G—Specifies the matrix that relates the process noise vector to the model states.• C—Specifies the output matrix that relates the outputs to the states.• D—Specifies the transmission matrix that relates the inputs to the outputs.• H—Specifies the matrix that relates the process noise vector to the model outputs.• Sampling Time (s)—Specifies the sampling time of the system model and determines whether the model represents a continuous-time or discrete-time system. If the model represents a continuous-time system, Sampling Time (s) must equal zero. If the model represents a discrete-time system, Sampling Time (s) must be greater than zero and equal to the sampling rate, in seconds, of the discrete system. The default is 0. <p> Note If you use the inputs to create a continuous-time system, setting the Sampling Time (s) to a value greater than zero does not yield the discrete-time equivalent of the system. You must use the CD Convert Continuous Stochastic to Discrete VI to convert the continuous-time system to the discrete-time equivalent of the system.</p>

Second-Order Statistics Noise Model	<p>Specifies a mathematical representation of the noise model of a stochastic state-space model. You can create a noise model using the CD Construct Noise Model VI.</p> <ul style="list-style-type: none"> • E{w}—Specifies the expected value or mean of the process noise vector. The default is 0. • Q—Specifies the covariance matrix of the process noise vector. The default is 0.01. • E{v}—Specifies the expected value or mean of the measurement noise vector. The default is 0. • R—Specifies the covariance matrix of the measurement noise vector. The default is 0.1. • N—Specifies the cross-covariance matrix between the process noise vector and the measurement noise vector. If these noise vectors are uncorrelated, N is a matrix of zeros. The default is 0.
Initial State Estimate $\hat{x}(t_0)$	Specifies the initial states from which this function begins estimating the model states. If you do not specify a value for this parameter, Initial State Estimate $\hat{x}(t_0)$ is a vector of zeros.
Input $u(t)$	Specifies the control action this function applies to the model. If you specify a vector of zeros for Input $u(t)$, this function does not apply any control action.
Output $y(t)$	Specifies the measurement made on the stochastic state-space model.
Initial Estimation Error Covariance $P(t_0)$	Specifies the initial covariance matrix of the estimation error. If you do not specify a value for this parameter, Initial Estimation Error Covariance $P(t_0)$ is a matrix of zeros.

Block Diagram Outputs

Parameter	Description
Estimated Output $y\hat{(t)}$	Returns the estimated values of the model outputs at time t .
Estimated State $x\hat{(t)}$	Returns the estimated model states at time t .
Estimation Error Covariance $P(t)$	Returns the covariance matrix of the estimation error associated with the estimated model states $x\hat{(t)}$.
Kalman Filter Gain $L(t)$	Returns the estimator gain matrix this function uses to estimate the model states $x\hat{(t)}$ at time t .

CD Continuous Recursive Kalman Filter Details

The following equations define the outputs this function calculates:

$$\text{Estimated Output } \mathbf{yhat(t)} = \mathbf{C(t)xhat(t)} + \mathbf{D(t)u(t)}$$

$$\text{Kalman Filter Gain } \mathbf{L(t)} = [\mathbf{P(t)C}^T(t) + \mathbf{G(t)Q(t)H}^T(t) + \mathbf{G(t)N(t)}] .$$

$$[\mathbf{H(t)Q(t)H}^T(t) + \mathbf{H(t)N(t)} + \mathbf{N}^T(t)\mathbf{H(t)}^T + \mathbf{R(t)}]^{-1}$$

$$\text{Estimation Error Covariance } \mathbf{P(t)} = \mathbf{A(t)P(t)} + \mathbf{P(t)A}^T(t) + \mathbf{G(t)Q(t)G}^T(t) -$$

$$\mathbf{P(t)C(t)}^T[\mathbf{H(t)Q(t)H}^T(t) + \mathbf{H(t)N(t)} + \mathbf{N}^T(t)\mathbf{H}^T(t) + \mathbf{R}]^{-1}\mathbf{C(t)P(t)} -$$

$$[\mathbf{G(t)Q(t)H}^T(t) + \mathbf{G(t)N(t)}][\mathbf{H(t)Q(t)H}^T(t) + \mathbf{H(t)N(t)} + \mathbf{N}^T(t)\mathbf{H}^T(t) + \mathbf{R}]^{-1} .$$

$$[\mathbf{G(t)Q(t)H}^T(t) + \mathbf{G(t)N(t)}]^T$$

$$\text{Estimated State } \mathbf{xhat(t)} = \mathbf{A(t)xhat(t)} + \mathbf{B(t)u(t)} + \mathbf{L(t)[y(t) - yhat(t)]}$$

Derivative Function

Owning Palette: [Continuous Linear Systems Functions](#)

Installed With: Control Design and Simulation Module

Calculates the derivative of a continuous signal.

[Details](#) [Example](#)

[Dialog Box Options](#)

[Block Diagram Inputs](#)

[Block Diagram Outputs](#)

■ Place on the block diagram ■ Find on the **Functions** palette

Dialog Box Options

Parameter	Description
Polymorphic instance	Specifies whether this function is Scalar or Vector . The default value is Scalar .
Preview	Displays a graphical preview, if available, of the function output or configuration.

Block Diagram Inputs

Parameter	Description
input	Specifies the input to the function.

Block Diagram Outputs

Parameter	Description
output	Returns the output of the function.

Derivative Details

This function uses the following first-order backward difference formula to calculate the output.

$$y(t) = \frac{u(t) - u(t-1)}{dt}$$

where u is the input

y is the output

t is the current simulation time



Note If the input signal you use has noise, this function might return unwanted results. Consider filtering noisy input signals to avoid this problem. For example, the following transfer function can provide a filtered derivative for a noisy signal.

$$\frac{50 \cdot s}{s + 50}$$

where s is the Laplace transform variable.

Derivatives of Discontinuous Signals

If you take the derivative of a discontinuous signal, such as the [Step Signal](#) function, and integrate that derivative, the LabVIEW Control Design and Simulation Module returns an error. Refer to the labview\examples\Control and Simulation\Simulation\PID\PID.llb for an example of how to address this scenario without causing an error.

Feedthrough Behavior

All input/output pairs of this function have [direct feedthrough behavior](#).

Example

Refer to the SimEx derivative VI in the labview\examples\Control and Simulation\Simulation\Continuous Linear directory for an example of using the Derivative VI.

- Open example ■ Browse related examples

Integrator Function

Owning Palette: [Continuous Linear Systems Functions](#)

Installed With: Control Design and Simulation Module

Integrates a continuous input signal using the ordinary differential equation (ODE) solver you specify for the simulation.

Use the [Configure Simulation Parameters](#) dialog box to specify the ODE solver.

[Details](#) [Examples](#)

[Dialog Box Options](#)

[Block Diagram Inputs](#)

[Block Diagram Outputs](#)

■ Place on the block diagram ■ Find on the **Functions** palette

Dialog Box Options

Parameter	Description
Polymorphic instance	Specifies whether this function is Scalar or Vector . The default value is Scalar .
Parameters	Lists all the parameters associated with this function. Select a parameter from this list to configure the parameter. When you select a parameter, the parameter and its associated Parameter source control appear in the Parameter Information section of the configuration dialog box.
Preview	Displays a graphical preview, if available, of the function output or configuration.
Parameter Information	Contains the parameters you can configure for this function. You must select a parameter from the Parameters list to make that parameter and its associated Parameter source control visible in the Parameter Information section of the configuration dialog box.
Parameter source	Specifies whether you configure this parameter using the Configuration Dialog Box or a Terminal on the simulation diagram. The default value is Configuration Dialog Box . If you select Terminal , LabVIEW displays an input for that parameter on the simulation diagram, and you can wire values to that input to configure this function programmatically. If you select Configuration Dialog Box , LabVIEW removes that input from the simulation diagram. You then must set the value for this parameter inside the configuration dialog box.
initial condition	Specifies the output of the integrator at the simulation initial time. The default value is 0.
limit type	Specifies the limit behavior of the integral. You can choose from the following options: <ul style="list-style-type: none">- upper—The integrator output saturates at the upper limit. The output can drop below the lower limit.

	<ul style="list-style-type: none"> - lower—The integrator output saturates at the lower limit. The output can exceed the upper limit. - both—The integrator output saturates at the lower limit and the upper limit. - none—(Default) The function does not limit the integrator output.
upper limit	Specifies the upper limit of the output. The function ignores this value unless you set limit type to upper or both . The default value of upper limit is 0.
lower limit	Specifies the lower limit of the output. The function ignores this value unless you set limit type to lower or both . The default value of lower limit is 0.
initial condition for reset	Specifies the initial condition when the integrator resets. The LabVIEW Control Design and Simulation Module does not use this parameter if you specify a value of none for the reset type parameter.
reset type	<p>Specifies the condition at which the integrator resets. You can select from the following options:</p> <ul style="list-style-type: none"> - rising—The integrator resets when the reset signal crosses the x-axis with a positive slope. - falling—The integrator resets when the reset signal crosses the x-axis with a negative slope. - either—The integrator resets when the reset signal crosses the x-axis with either a positive or negative slope. - none—(Default) The integrator does not reset.
reset	Specifies the signal used to trigger the integrator to reset. The condition you set in reset type determines when the Control Design and Simulation Module triggers the reset.
reset offset	<p>Specifies the value around which this function detects a crossing for the reset type condition. This function detects a crossing if the following equation is true:</p> $\text{sgn}(\text{reset}(t-1)-\text{reset offset}(t)) \neq \text{sgn}(\text{reset}(t)-\text{reset offset}(t))$

Block Diagram Inputs

Parameter	Description
initial condition	Specifies the output of the integrator at the simulation initial time. The default value is 0.
limit type	Specifies the limit behavior of the integral. You can choose from the following options: <ul style="list-style-type: none">- upper—The integrator output saturates at the upper limit. The output can drop below the lower limit.- lower—The integrator output saturates at the lower limit. The output can exceed the upper limit.- both—The integrator output saturates at the lower limit and the upper limit.- none—(Default) The function does not limit the integrator output.
upper limit	Specifies the upper limit of the output. The function ignores this value unless you set limit type to upper or both . The default value of upper limit is 0.
lower limit	Specifies the lower limit of the output. The function ignores this value unless you set limit type to lower or both . The default value of lower limit is 0.
initial condition for reset	Specifies the initial condition when the integrator resets. The LabVIEW Control Design and Simulation Module does not use this parameter if you specify a value of none for the reset type parameter.
reset type	Specifies the condition at which the integrator resets. You can select from the following options: <ul style="list-style-type: none">- rising—The integrator resets when the reset signal crosses the x-axis with a positive slope.- falling—The integrator resets when the reset signal crosses the x-axis with a negative slope.- either—The integrator resets when the reset signal crosses the x-axis with either a positive or negative slope.- none—(Default) The integrator does not reset.

reset	Specifies the signal used to trigger the integrator to reset. The condition you set in reset type determines when the Control Design and Simulation Module triggers the reset.
reset offset	Specifies the value around which this function detects a crossing for the reset type condition. This function detects a crossing if the following equation is true: $\text{sgn}(\text{reset}(t-1) - \text{reset offset}(t)) \neq \text{sgn}(\text{reset}(t) - \text{reset offset}(t))$
input	Specifies the signal to which you want to apply the function.

Block Diagram Outputs

Parameter	Description
output	Returns the integral of the input signal over time. This integral also is the state value.
limited?	Returns TRUE if the output of this function saturates at the upper limit or lower limit . limited? returns FALSE if you specified a value of none for the limit type parameter or if the value of the output is between the upper limit and the lower limit .
reset?	Returns TRUE if the integrator states were reset during the last time step.

Integrator Details

This function uses the following equation to calculate the output:

$$y(t) = \int_{t_0}^t u(\tau) d\tau + y(t_0)$$

where u is the input

y is the output

t_0 is the simulation initial time

t is the current simulation time

If you select the **Vector** instance of this function, the **input** and **output** parameters must be the same size as the **initial condition** parameter. If you set **reset type** to **rising**, **falling**, or **either**, the **initial condition for reset** parameter also must be the same size as the **initial condition** parameter.

All other Integrator function parameters must be the same size as the **initial condition** parameter or be of size one. If the parameter array is of size one, this function assumes that array[i] equals array[0] for all values of i .

Feedthrough Behavior

For this function, the following input/output pairs have indirect feedthrough behavior.

- **input – output**
- **input – limited?**
- **reset – output**
- **reset – limited?**
- **initial condition for reset – output**
- **initial condition for reset – limited?**

All other input/output pairs have direct feedthrough behavior.

Examples

Refer to the following VIs for examples of using the Integrator VI:

- SimEx integrator VI: labview\examples\Control and Simulation\Simulation\Continuous Linear
 - Open example ■ Browse related examples
- SimEx integrator reset VI: labview\examples\Control and Simulation\Simulation\Continuous Linear
 - Open example ■ Browse related examples
- SimEx limited integrator VI: labview\examples\Control and Simulation\Simulation\Continuous Linear
 - Open example ■ Browse related examples

State-Space Function

Owning Palette: [Continuous Linear Systems Functions](#)

Installed With: Control Design and Simulation Module

Implements a system model in state-space form. You [define the system model](#) by specifying the input, output, state, and direct transmission matrices.

[Details](#) [Example](#)

[Dialog Box Options](#)

[Block Diagram Inputs](#)

[Block Diagram Outputs](#)

■ Place on the block diagram ■ Find on the **Functions** palette

Dialog Box Options

Parameter	Description
Polymorphic instance	Specifies whether this function is single-input single-output (SISO) or multiple-input multiple-output (MIMO). The default value is SISO .
Feedthrough	Configures the function to be either a Direct or Indirect feedthrough function. Enable this control by selecting a parameter from the Parameters list and then selecting Terminal from the Parameter source pull-down menu. If you select Configuration Dialog Box from the Parameter source pull-down menu, LabVIEW disables this control and calculates the feedthrough behavior automatically. Refer to the Details section for a description of the parameters that determine the feedthrough behavior of this function.
Parameters	Lists all the parameters associated with this function. Select a parameter from this list to configure the parameter. When you select a parameter, the parameter and its associated Parameter source control appear in the Parameter Information section of the configuration dialog box.
Preview	Displays a graphical preview, if available, of the function output or configuration.
Parameter Information	Contains the parameters you can configure for this function. You must select a parameter from the Parameters list to make that parameter and its associated Parameter source control visible in the Parameter Information section of the configuration dialog box.
Parameter source	Specifies whether you configure this parameter using the Configuration Dialog Box or a Terminal on the simulation diagram. The default value is Configuration Dialog Box . If you select Terminal , LabVIEW displays an input for that parameter on the simulation diagram, and

	<p>you can wire values to that input to configure this function programmatically. If you select Configuration Dialog Box, LabVIEW removes that input from the simulation diagram. You then must set the value for this parameter inside the configuration dialog box.</p>
State-Space	<p>Specifies the state-space model.</p> <ul style="list-style-type: none"> • Load Model—Loads model information from a data file. • Save Model—Saves model information to a data file. This file is compatible with the Control Design VIs and functions. • Copy to Clipboard—Copies the current model definition to the clipboard. From the clipboard, you can paste the model on the block diagram or into another configuration dialog box of the same model form. • Paste from Clipboard—Pastes model information from the clipboard to the configuration dialog box. • Model Dimensions—Use this section to specify the number of inputs, states, and outputs of the system model. <ul style="list-style-type: none"> - Inputs—Specifies the number of model inputs. - States—Specifies the number of model states. - Outputs—Specifies the number of model outputs. • A—Specifies the $n \times n$ state matrix of the given system. • B—Specifies the $n \times m$ input matrix of the given system. • C—Specifies the $r \times n$ output matrix of the given system. • D—Specifies the $r \times m$ direct transmission matrix of the given system.

	<p>where m is the number of inputs</p> <p>n is the number of states</p> <p>r is the number of outputs</p>
initial state (x0)	Specifies a vector of initial states for the system. This vector must be of length n , where n is the number of states.
reset?	Sets the model state(s) to the values you specify in reset state (xr) , when TRUE.
reset state (xr)	Specifies the values to which to set the model state(s) when reset? is TRUE. The default is 0.

Block Diagram Inputs

Parameter	Description
initial state (x0)	Specifies a vector of initial states for the system. This vector must be of length n , where n is the number of states.
reset?	Sets the model state(s) to the values you specify in reset state (xr) , when TRUE.
reset state (xr)	Specifies the values to which to set the model state(s) when reset? is TRUE. The default is 0.
input u(k)	Specifies the input to the system. input u(k) must be a vector of length m , where m is the number of inputs.
State-Space	Specifies a state-space model. This input accepts either a block diagram constant or a model you created using the Control Design VIs and functions.

Block Diagram Outputs

Parameter	Description
output $y(k)$	Returns the current output of the system. This vector must be of length r , where r is the number of outputs.
state $x(k)$	Returns the set of current internal states of the system. This vector must be of length n , where n is the number of states.

State-Space Details

The following equations define the state-space system.

$$\dot{x} = Ax + Bu$$

$$y = Cx + Du$$

where u is the input vector

y is the output vector

x is the state vector

A , B , C , D are the state-space matrices. Refer to the **State-Space** parameter help for more information about these matrices.

Feedthrough Behavior

The value you specify for the **D** matrix of the **State-Space** parameter determines the feedthrough behavior of this function.

- If the **D** matrix is nonzero, all input/output pairs have direct feedthrough behavior.
- If the **D** matrix is zero, the following input/output pairs have indirect feedthrough behavior.
 - **input — output**
 - **input — states**

All other input/output pairs have direct feedthrough behavior.

When you use the configuration dialog box to configure the **D** matrix, LabVIEW verifies that the feedthrough behavior is correct. For example, if you set the **Feedthrough** parameter to **Indirect**, and you set the **D** matrix to nonzero, LabVIEW changes the **Feedthrough** parameter to **Direct**.

If you specify the value of the **D** matrix programmatically by wiring a value to the parameter terminal, LabVIEW does not adjust the feedthrough behavior for you. You must ensure that you specify the proper feedthrough behavior for the value of the **D** matrix that you specify.

Example

Refer to the SimEx state space VI in the labview\examples\Control and Simulation\Simulation\Continuous Linear directory for an example of using the State-Space VI.

- [Open example](#)
- [Browse related examples](#)

Transfer Function Function

Owning Palette: [Continuous Linear Systems Functions](#)

Installed With: Control Design and Simulation Module

Implements a system model in transfer function form. You [define the system model](#) by specifying the **Numerator** and **Denominator** of the transfer function equation.

[Details](#) [Example](#)

[Dialog Box Options](#)

[Block Diagram Inputs](#)

[Block Diagram Outputs](#)

■ Place on the block diagram ■ Find on the **Functions** palette

Dialog Box Options

Parameter	Description
Polymorphic instance	Specifies whether this function is single-input single-output (SISO) or multiple-input multiple-output (MIMO). The default value is SISO .
Feedthrough	Configures the function to be either a Direct or Indirect feedthrough function. Enable this control by selecting a parameter from the Parameters list and then selecting Terminal from the Parameter source pull-down menu. If you select Configuration Dialog Box from the Parameter source pull-down menu, LabVIEW disables this control and calculates the feedthrough behavior automatically. Refer to the Details section for a description of the parameters that determine the feedthrough behavior of this function.
Parameters	Lists all the parameters associated with this function. Select a parameter from this list to configure the parameter. When you select a parameter, the parameter and its associated Parameter source control appear in the Parameter Information section of the configuration dialog box.
Preview	Displays a graphical preview, if available, of the function output or configuration.
Parameter Information	Contains the parameters you can configure for this function. You must select a parameter from the Parameters list to make that parameter and its associated Parameter source control visible in the Parameter Information section of the configuration dialog box.
Parameter source	Specifies whether you configure this parameter using the Configuration Dialog Box or a Terminal on the simulation diagram. The default value is Configuration Dialog Box . If you select Terminal , LabVIEW displays an input for that parameter on the simulation diagram, and

	<p>you can wire values to that input to configure this function programmatically. If you select Configuration Dialog Box, LabVIEW removes that input from the simulation diagram. You then must set the value for this parameter inside the configuration dialog box.</p>
Transfer Function	<p>Specifies the transfer function in terms of numerator and denominator polynomial functions.</p> <ul style="list-style-type: none"> • Load Model—Loads model information from a data file. • Save Model—Saves model information to a data file. This file is compatible with the Control Design VIs and functions. • Copy to Clipboard—Copies the current model definition to the clipboard. From the clipboard, you can paste the model on the block diagram or into another configuration dialog box of the same model form. • Paste from Clipboard—Pastes model information from the clipboard to the configuration dialog box. • Model Dimensions—Use this section to specify the number of inputs and outputs of the system model. You also use this section to specify the input-output location of the equation you want to edit. This section is available only if you select MIMO from the Polymorphic instance pull-down menu. <ul style="list-style-type: none"> - Inputs—Specifies the number of model inputs. - Outputs—Specifies the number of model outputs. • Current Input—Specifies the current column of the Input-Output Model. • Current Output—Specifies the current row of the Input-Output Model. • Input-Output Model—Displays a graphical representation of the MIMO model. Click a cell to

	<p>edit the equation at that input-output location. You also can use the Current Input and Current Output controls to specify the location of the equation you want to edit.</p> <ul style="list-style-type: none"> • Numerator—Specifies the coefficients of the numerator polynomial function in ascending powers of s, where s is the Laplace transform variable. For MIMO models, Numerator applies to the equation that the Current Input and Current Output parameters specify. • Denominator—Specifies the coefficients of the denominator polynomial function in ascending powers of s, where s is the Laplace transform variable. For MIMO models, Denominator applies to the equation that the Current Input and Current Output parameters specify. For each equation, the order of the Denominator must be greater than or equal to the order of the Numerator.
reset?	Sets the model state(s) to 0, when TRUE.

Block Diagram Inputs

Parameter	Description
reset?	Sets the model state(s) to 0, when TRUE.
input u(k)	Specifies the input to the system. input u(k) must be a vector of length m , where m is the number of inputs.
Transfer Function	Specifies a transfer function model. This input accepts either a block diagram constant or a model you created using Control Design VIs and functions.

Block Diagram Outputs

Parameter	Description
output $y(k)$	Returns the current output of the system. This vector must be of length r , where r is the number of outputs.
state $x(k)$	Returns the states of the system.

Transfer Function Details

For SISO models, this function uses the following equation to calculate the output:

$$H(s) = \frac{b_0 + b_1 s + \dots + b_{m-1} s^{m-1} + b_m s^m}{a_0 + a_1 s + \dots + a_{n-1} s^{n-1} + a_n s^n}$$

For MIMO models, this function calculates the output as $H = [H_{ij}]$

where $b_0 \dots b_m$ are the coefficients of the numerator polynomial

$a_0 \dots a_n$ are the coefficients of the denominator polynomial

m is the order of the numerator

n is the order of the denominator

s is the Laplace transform variable

i is the index number of the input

j is the index number of the output

Feedthrough Behavior

The values you specify for the **Numerator** and **Denominator** subparameters determine the [feedthrough behavior](#) of this function. Given m as the order of the **Numerator** and n as the order of the **Denominator**:

- If $n = m$, all input/output pairs have direct feedthrough behavior.
 - If $n > m$, the following input/output pairs have indirect feedthrough behavior.
 - **input — output**
 - **input — states**
- All other input/output pairs have direct feedthrough behavior.
- If $n < m$, LabVIEW returns an error.

When you use the configuration dialog box to configure n and m , LabVIEW verifies that the feedthrough behavior is correct. For example, if you set the **Feedthrough** parameter to **Indirect**, and you set n equal to m , LabVIEW changes the **Feedthrough** parameter to **Direct**.

If you define the transfer function programmatically, LabVIEW does not adjust the feedthrough behavior for you. You must ensure that you specify the proper feedthrough behavior for the orders of n and m you specify.

Example

Refer to the SimEx transfer function VI in the labview\examples\Control and Simulation\Simulation\Continuous Linear directory for an example of using the Transfer Function VI.

- [Open example](#)
- [Browse related examples](#)

Transport Delay Function

Owning Palette: [Continuous Linear Systems Functions](#)

Installed With: Control Design and Simulation Module

Delays the **input** signal by the amount of time you specify.

[Details](#) [Example](#)

[Dialog Box Options](#)

[Block Diagram Inputs](#)

[Block Diagram Outputs](#)

■ Place on the block diagram ■ Find on the **Functions** palette

Dialog Box Options

Parameter	Description
Polymorphic instance	Specifies whether this function is Scalar , Vector , or vector with a Vector of Delays . The default value is Scalar .
Feedthrough	Configures the function to be either a Direct or Indirect feedthrough function.
Parameters	Lists all the parameters associated with this function. Select a parameter from this list to configure the parameter. When you select a parameter, the parameter and its associated Parameter source control appear in the Parameter Information section of the configuration dialog box.
Preview	Displays a graphical preview, if available, of the function output or configuration.
Parameter Information	Contains the parameters you can configure for this function. You must select a parameter from the Parameters list to make that parameter and its associated Parameter source control visible in the Parameter Information section of the configuration dialog box.
Parameter source	Specifies whether you configure this parameter using the Configuration Dialog Box or a Terminal on the simulation diagram. The default value is Configuration Dialog Box . If you select Terminal , LabVIEW displays an input for that parameter on the simulation diagram, and you can wire values to that input to configure this function programmatically. If you select Configuration Dialog Box , LabVIEW removes that input from the simulation diagram. You then must set the value for this parameter inside the configuration dialog box.
initial condition	Specifies the output when the simulation time is less than the delay. The default value is 0.
delay (s)	Specifies the length of time to delay the input signal. The default value is 1.

max delay (s)	Specifies the maximum possible delay when using a fixed step-size ordinary differential equation (ODE) solver. The default value is 1.
--------------------------	--

Block Diagram Inputs

Parameter	Description
initial condition	Specifies the output when the simulation time is less than the delay. The default value is 0.
delay (s)	Specifies the length of time to delay the input signal. The default value is 1.
max delay (s)	Specifies the maximum possible delay when using a fixed step-size ordinary differential equation (ODE) solver. The default value is 1.
input	Specifies the input to the system.

Block Diagram Outputs

Parameter	Description
output	Returns the output of the function.

Transport Delay Details

This function uses the following equation to calculate the output:

$$y(t) = u(t - \tau)$$

where u is the input

y is the output

t is the current simulation time

τ is the **delay (s)**

When the simulation time is less than the **delay (s)**, the function outputs the **initial condition**. If the **delay (s)** is greater than the simulation time step size, the function linearly interpolates to produce an output. If the **delay (s)** is less than the simulation time step size, the function extrapolates to produce an output.

Feedthrough Behavior

Use the **Feedthrough** option to specify the [feedthrough behavior](#) of the following input/output pair:

- **input – output**

All other input/output pairs have direct feedthrough behavior.

Example

Refer to the SimEx transport delay VI in the labview\examples\Control and Simulation\Simulation\Continuous Linear directory for an example of using the Transport Delay VI.

- Open example ■ Browse related examples

Zero-Pole-Gain Function

Owning Palette: [Continuous Linear Systems Functions](#)

Installed With: Control Design and Simulation Module

Implements a system model in zero-pole-gain form. You [define the system model](#) by specifying the **Zeros**, **Poles**, and **Gain** of the zero-pole-gain equation.

[Details](#) [Example](#)

[Dialog Box Options](#)

[Block Diagram Inputs](#)

[Block Diagram Outputs](#)

■ Place on the block diagram ■ Find on the **Functions** palette

Dialog Box Options

Parameter	Description
Polymorphic instance	Specifies whether this function is single-input single-output (SISO) or multiple-input multiple-output (MIMO). The default value is SISO .
Feedthrough	Configures the function to be either a Direct or Indirect feedthrough function. Enable this control by selecting a parameter from the Parameters list and then selecting Terminal from the Parameter source pull-down menu. If you select Configuration Dialog Box from the Parameter source pull-down menu, LabVIEW disables this control and calculates the feedthrough behavior automatically. Refer to the Details section for a description of the parameters that determine the feedthrough behavior of this function.
Parameters	Lists all the parameters associated with this function. Select a parameter from this list to configure the parameter. When you select a parameter, the parameter and its associated Parameter source control appear in the Parameter Information section of the configuration dialog box.
Preview	Displays a graphical preview, if available, of the function output or configuration.
Parameter Information	Contains the parameters you can configure for this function. You must select a parameter from the Parameters list to make that parameter and its associated Parameter source control visible in the Parameter Information section of the configuration dialog box.
Parameter source	Specifies whether you configure this parameter using the Configuration Dialog Box or a Terminal on the simulation diagram. The default value is Configuration Dialog Box . If you select Terminal , LabVIEW displays an input for that parameter on the simulation diagram, and

	<p>you can wire values to that input to configure this function programmatically. If you select Configuration Dialog Box, LabVIEW removes that input from the simulation diagram. You then must set the value for this parameter inside the configuration dialog box.</p>
Zeros-Poles-Gain	<p>Specifies the zero-pole-gain model.</p> <ul style="list-style-type: none"> • Load Model—Loads model information from a data file. • Save Model—Saves model information to a data file. This file is compatible with the Control Design VIs and functions. • Copy to Clipboard—Copies the current model definition to the clipboard. From the clipboard, you can paste the model on the block diagram or into another configuration dialog box of the same model form. • Paste from Clipboard—Pastes model information from the clipboard to the configuration dialog box. • Model Dimensions—Use this section to specify the number of inputs and outputs of the system model. You also use this section to specify the input-output location of the equation you want to edit. This section is available only if you select MIMO from the Polymorphic instance pull-down menu. <ul style="list-style-type: none"> - Inputs—Specifies the number of model inputs. - Outputs—Specifies the number of model outputs. • Current Input—Specifies the current column of the Input-Output Model. • Current Output—Specifies the current row of the Input-Output Model. • Input-Output Model—Displays a graphical representation of the MIMO model. Click a cell to edit the equation at that input-output location. You

	<p>also can use the Current Input and Current Output controls to specify the location of the equation you want to edit.</p> <ul style="list-style-type: none"> • Gain—Specifies a real value representing the common gain of the zero-pole-gain model. For MIMO models, Gain applies to the equation that the Current Input and Current Output parameters specify. • Zeros—Specifies zeros of the zero-pole-gain model. For all zeros with an imaginary part, the conjugate complex number also must belong to this array. For MIMO models, Zeros applies to the equation that the Current Input and Current Output parameters specify. • Poles—Specifies the poles of the zero-pole-gain model. For all poles with an imaginary part, the conjugate complex number also must belong to this array. For MIMO models, Poles applies to the equation that the Current Input and Current Output parameters specify.
reset?	Sets the model state(s) to 0, when TRUE.

Block Diagram Inputs

Parameter	Description
reset?	Sets the model state(s) to 0, when TRUE.
input u(k)	Specifies the input to the system. input u(k) must be a vector of length m , where m is the number of inputs.
Zeros-Poles-Gain	Specifies a zero-pole-gain model. This input accepts either a block diagram constant or a model you created using the Control Design VIs and functions.

Block Diagram Outputs

Parameter	Description
output $y(k)$	Returns the current output of the system. This vector must be of length r , where r is the number of outputs.
state $x(k)$	Returns a vector containing the current internal states of the system.

Zero-Pole-Gain Details

For SISO models, this function uses the following equation to calculate the output:

$$H(s) = \frac{k(s - Z[1])(s - Z[2])\dots(s - Z[m])}{(s - P[1])(s - P[2])\dots(s - P[n])}$$

For MIMO models, this function calculates the output as $H = [H_{ij}]$

where k is the gain

$Z[m]$ is the array of zeros

$P[n]$ is the array of poles

s is the Laplace transform variable

i is the index number of the input

j is the index number of the output

Feedthrough Behavior

The values you specify for the **Zeros** and **Poles** subparameters determine the feedthrough behavior of this function. Given Z as the **Zeros** subparameter and P as the **Poles** subparameter:

- If the order of Z = the order of P , all input/output pairs have direct feedthrough behavior.
- If the order of Z < the order of P , the following input/output pairs have indirect feedthrough behavior.
 - **input – output**
 - **input – states**

All other input/output pairs have direct feedthrough behavior.

- If the order of Z > the order of P , LabVIEW returns an error.

When you use the configuration dialog box to configure Z and P , LabVIEW verifies that the feedthrough behavior is correct. For example, if you set the **Feedthrough** parameter to **Indirect**, and you set the order of Z equal to the order of P , LabVIEW changes the **Feedthrough** parameter to **Direct**.

If you define the zero-pole-gain equation programmatically, LabVIEW does not adjust the feedthrough behavior for you. You must ensure that you specify the proper feedthrough behavior for the orders of Z and P you specify.

The direct or indirect feedthrough behavior of a function determines how you can use that function in a feedback cycle.

Example

Refer to the SimEx zero-pole-gain VI in the labview\examples\Control and Simulation\Simulation\Continuous Linear directory for an example of using the Zero-Pole-Gain VI.

- [Open example](#)
- [Browse related examples](#)

Discrete Linear Systems Functions

Owning Palette: [Simulation VIs and Functions](#)

Installed With: Control Design and Simulation Module. This topic might not match its corresponding palette in LabVIEW depending on your operating system, licensed product(s), and target.

Use the Discrete Linear Systems functions to represent discrete linear systems of difference equations on the simulation diagram.

The functions on this palette can return [general LabVIEW error codes](#) or [specific Simulation error codes](#). If you use the functions on this palette in a [Simulation Loop](#), LabVIEW sends any errors that these functions return to the **Error** output on the Output Node of the Simulation Loop.

Palette Object	Description
Discrete Filter	Computes an infinite impulse response (IIR) filter or finite impulse response (FIR) filter with forward and backward coefficients. The Numerator and Denominator parameters define these coefficients.
Discrete First-Order Hold	Extrapolates the value of the output signal based on the first derivative of the input signal.
Discrete Integrator	Integrates the input using forward rectangular (Euler) integration, backward rectangular (Euler) integration, or trapezoidal integration.
Discrete Kalman Filter	Implements a discrete-time, linear time-variant, recursive Kalman filter. You define the system by specifying the stochastic state-space model and noise model as well as the inputs and outputs to the system. The Discrete Kalman Filter function calculates the predicted state estimates $\hat{x}(k+1 k)$, the corrected state estimates $\hat{x}(k k)$, the corresponding gains used to calculate these estimates, and the associated estimation error covariances corresponding to these estimates. This function also calculates the estimated output $\hat{y}(k)$.
Discrete	Implements a discrete-time observer for a linear state-space

<u>Observer</u>	system model.
<u>Discrete State-Space</u>	Implements a system model in discrete state-space form. You define the system model by specifying the input, output, state, and direct transmission matrices.
<u>Discrete Stochastic State-Space</u>	Implements a discrete-time, linear, stochastic state-space system. You define the system model by specifying the input, output, state, and direct transmission matrices. You also specify the matrices relating the process noise to the system states and outputs.
<u>Discrete Transfer Function</u>	Implements a system model in discrete transfer function form. You define the system model by specifying the Numerator and Denominator of the transfer function equation.
<u>Discrete Unit Delay</u>	Delays the input by the value that you specify for the sample period (s) of this function.
<u>Discrete Zero-Order Hold</u>	Holds the input signal for a definite period of time equal to the value you specify for the sample period (s) parameter of this function. Use this function to discretize or resample an input signal.
<u>Discrete Zero-Pole-Gain</u>	Implements a system model in discrete zero-pole-gain form. You define the system model by specifying the Zeros , Poles , and Gain of the zero-pole-gain equation.

Discrete Filter Function

Owning Palette: [Discrete Linear Systems Functions](#)

Installed With: Control Design and Simulation Module

Computes an infinite impulse response (IIR) filter or finite impulse response (FIR) filter with forward and backward coefficients. The **Numerator** and **Denominator** parameters define these coefficients.

[Details](#) [Example](#)

[Dialog Box Options](#)

[Block Diagram Inputs](#)

[Block Diagram Outputs](#)

■ Place on the block diagram ■ Find on the **Functions** palette

Dialog Box Options

Parameter	Description
Polymorphic instance	Specifies whether this function is single-input single-output (SISO) or multiple-input multiple-output (MIMO). The default value is SISO .
Feedthrough	Configures the function to be either a Direct or Indirect feedthrough function. Enable this control by selecting a parameter from the Parameters list and then selecting Terminal from the Parameter source pull-down menu. If you select Configuration Dialog Box from the Parameter source pull-down menu, LabVIEW disables this control and calculates the feedthrough behavior automatically. Refer to the Details section for a description of the parameters that determine the feedthrough behavior of this function.
Parameters	Lists all the parameters associated with this function. Select a parameter from this list to configure the parameter. When you select a parameter, the parameter and its associated Parameter source control appear in the Parameter Information section of the configuration dialog box.
Preview	Displays a graphical preview, if available, of the function output or configuration.
Parameter Information	Contains the parameters you can configure for this function. You must select a parameter from the Parameters list to make that parameter and its associated Parameter source control visible in the Parameter Information section of the configuration dialog box.
Parameter source	Specifies whether you configure this parameter using the Configuration Dialog Box or a Terminal on the simulation diagram. The default value is Configuration Dialog Box . If you select Terminal , LabVIEW displays an input for that parameter on the simulation diagram, and

	<p>you can wire values to that input to configure this function programmatically. If you select Configuration Dialog Box, LabVIEW removes that input from the simulation diagram. You then must set the value for this parameter inside the configuration dialog box.</p>
Filter	<p>Specifies the filter in terms of numerator and denominator polynomial functions.</p> <ul style="list-style-type: none"> • Numerator—Specifies the coefficients of the numerator polynomial function in ascending powers of z, where z is the z-transform variable. • Denominator—Specifies the coefficients of the denominator polynomial function in ascending powers of z, where z is the z-transform variable. The order of the Denominator must be greater than or equal to the order of the Numerator.
sample period (s)	<p>Specifies the length of the discrete time step, in seconds, of this function. If you enter a value of –1, this function inherits the discrete time step you specify for the simulation diagram. Otherwise, the value of sample period (s) must be a multiple of the discrete time step you specify for the simulation diagram. The default value of sample period (s) is 1.</p>
sample skew (s)	<p>Specifies the length of time by which you want to delay the execution of this function. The default value is 0. The value of this parameter must satisfy the following relationship:</p> $0 \leq \text{sample skew (s)} \leq \text{sample period (s)}$
reset?	Sets the model state(s) to 0, when TRUE.

Block Diagram Inputs

Parameter	Description
Filter	<p>Specifies the filter in terms of numerator and denominator polynomial functions.</p> <ul style="list-style-type: none"> • Numerator—Specifies the coefficients of the numerator polynomial function in ascending powers of z, where z is the z-transform variable. • Denominator—Specifies the coefficients of the denominator polynomial function in ascending powers of z, where z is the z-transform variable. The order of the Denominator must be greater than or equal to the order of the Numerator.
sample period (s)	<p>Specifies the length of the discrete time step, in seconds, of this function. If you enter a value of -1, this function inherits the discrete time step you specify for the simulation diagram. Otherwise, the value of sample period (s) must be a multiple of the discrete time step you specify for the simulation diagram. The default value of sample period (s) is 1.</p>
sample skew (s)	<p>Specifies the length of time by which you want to delay the execution of this function. The default value is 0. The value of this parameter must satisfy the following relationship:</p> $0 \leq \text{sample skew (s)} \leq \text{sample period (s)}$
reset?	Sets the model state(s) to 0 , when TRUE.
input u(k)	Specifies the input to this function.

Block Diagram Outputs

Parameter	Description
output y(k)	Returns the output of this function.

Discrete Filter Details

This function uses the following equation to calculate the output:

$$H(z) = \frac{b_0 + b_1 z^{-1} + \dots + b_m z^{-m}}{a_0 + a_1 z^{-1} + \dots + a_n z^{-n}}$$

where $b_0 \dots b_m$ are the coefficients of the numerator polynomial function

$a_0 \dots a_n$ are the coefficients of the denominator polynomial function

m is the order of the numerator

n is the order of the denominator

z is the z -transform variable

The Discrete Filter function is based completely on the [Discrete Transfer Function](#) function.

Feedthrough Behavior

The number of zeros that begin the **Numerator** and **Denominator** subparameters determine the **feedthrough behavior** of this function. Given c as the index of the first non-zero coefficient in the **Numerator** and d as the index of the first non-zero coefficient in the **Denominator**:

- If $c = d$, all input/output pairs have direct feedthrough behavior.
- If $c > d$, the following input/output pair has indirect feedthrough behavior.
 - **input $u(k)$ – output $y(k)$**All other input/output pairs have direct feedthrough behavior.
- If $c < d$, LabVIEW returns an error.

When you use the configuration dialog box to configure c and d , LabVIEW verifies that the feedthrough behavior is correct. For example, if you set the **Feedthrough** parameter to **Indirect**, and you set c equal to d , LabVIEW changes the **Feedthrough** parameter to **Direct**.

If you set **Numerator** and **Denominator** programmatically, LabVIEW does not adjust the feedthrough behavior for you. You must ensure that you specify the proper feedthrough behavior for the **Numerator** and **Denominator** values you specify.

Example

Refer to the SimEx discrete filter VI in the labview\examples\Control and Simulation\Simulation\Discrete Linear directory for an example of using the Discrete Filter VI.

- Open example ■ Browse related examples

Discrete First-Order Hold Function

Owning Palette: [Discrete Linear Systems Functions](#)

Installed With: Control Design and Simulation Module

Extrapolates the value of the **output** signal based on the first derivative of the **input** signal.

[Details](#) [Example](#)

[Dialog Box Options](#)

[Block Diagram Inputs](#)

[Block Diagram Outputs](#)

■ Place on the block diagram ■ Find on the **Functions** palette

Dialog Box Options

Parameter	Description
Polymorphic instance	Specifies whether this function is Scalar or Vector . The default value is Scalar .
Parameters	Lists all the parameters associated with this function. Select a parameter from this list to configure the parameter. When you select a parameter, the parameter and its associated Parameter source control appear in the Parameter Information section of the configuration dialog box.
Preview	Displays a graphical preview, if available, of the function output or configuration.
Parameter Information	Contains the parameters you can configure for this function. You must select a parameter from the Parameters list to make that parameter and its associated Parameter source control visible in the Parameter Information section of the configuration dialog box.
Parameter source	Specifies whether you configure this parameter using the Configuration Dialog Box or a Terminal on the simulation diagram. The default value is Configuration Dialog Box . If you select Terminal , LabVIEW displays an input for that parameter on the simulation diagram, and you can wire values to that input to configure this function programmatically. If you select Configuration Dialog Box , LabVIEW removes that input from the simulation diagram. You then must set the value for this parameter inside the configuration dialog box.
sample period (s)	Specifies the length of the discrete time step, in seconds, of this function. If you enter a value of –1, this function inherits the discrete time step you specify for the simulation diagram. Otherwise, the value of sample period (s) must be a multiple of the discrete time step you specify for the simulation diagram. The default value of sample period (s) is 1.

sample skew (s)	Specifies the length of time by which you want to delay the execution of this function. The default value is 0. The value of this parameter must satisfy the following relationship: $0 \leq \text{sample skew (s)} \leq \text{sample period (s)}$
------------------------	---

Block Diagram Inputs

Parameter	Description
sample period (s)	Specifies the length of the discrete time step, in seconds, of this function. If you enter a value of –1, this function inherits the discrete time step you specify for the simulation diagram. Otherwise, the value of sample period (s) must be a multiple of the discrete time step you specify for the simulation diagram. The default value of sample period (s) is 1.
sample skew (s)	Specifies the length of time by which you want to delay the execution of this function. The default value is 0. The value of this parameter must satisfy the following relationship: $0 \leq \text{sample skew (s)} \leq \text{sample period (s)}$
input	Specifies the input to this function.

Block Diagram Outputs

Parameter	Description
output	Specifies the output of the system.

Discrete First-Order Hold Details

The following equation defines the output of this function.

$$y(t) = \begin{cases} u(t_i), & \text{if } \mathbf{sample\ skew\ (s)} > 0 \text{ and } t_i \leq t < t_i + \mathbf{sample\ skew\ (s)} \\ u(t_i) \text{ or } \mathbf{sample\ skew\ (s)} = 0 \text{ and } t_i \leq t < t_i + \mathbf{sample\ period\ (s)} \\ u(t_k) + m * (t - t_{k-1}), & t_k \leq t < t_{k+1} \end{cases}$$

where y is the output

u is the input

t is the current simulation time

t_i is the initial simulation time

$t_k = t_i + \mathbf{sample\ skew\ (s)} + k * \mathbf{sample\ period\ (s)}$, for $k = 1, 2, 3,$

...

$$m = \frac{u(t_k) - u(t_{k-1})}{t_k - t_{k-1}}$$

Feedthrough Behavior

All input/output pairs of this function have [direct feedthrough behavior](#).

Example

Refer to the SimEx first-order-hold VI in the labview\examples\Control and Simulation\Simulation\Discrete Linear directory for an example of using the Discrete First-Order Hold VI.

- [Open example](#)
- [Browse related examples](#)

Discrete Integrator Function

Owning Palette: [Discrete Linear Systems Functions](#)

Installed With: Control Design and Simulation Module

Integrates the input using forward rectangular (Euler) integration, backward rectangular (Euler) integration, or trapezoidal integration.

[Details](#) [Example](#)

[Dialog Box Options](#)

[Block Diagram Inputs](#)

[Block Diagram Outputs](#)

■ Place on the block diagram ■ Find on the **Functions** palette

Dialog Box Options

Parameter	Description
Polymorphic instance	Specifies whether this function is Scalar or Vector . The default value is Scalar .
Feedthrough	Configures the function to be either a Direct or Indirect feedthrough function. Enable this control by selecting a parameter from the Parameters list and then selecting Terminal from the Parameter source pull-down menu. If you select Configuration Dialog Box from the Parameter source pull-down menu, LabVIEW disables this control and calculates the feedthrough behavior automatically. Refer to the Details section for a description of the parameters that determine the feedthrough behavior of this function.
Parameters	Lists all the parameters associated with this function. Select a parameter from this list to configure the parameter. When you select a parameter, the parameter and its associated Parameter source control appear in the Parameter Information section of the configuration dialog box.
Preview	Displays a graphical preview, if available, of the function output or configuration.
Parameter Information	Contains the parameters you can configure for this function. You must select a parameter from the Parameters list to make that parameter and its associated Parameter source control visible in the Parameter Information section of the configuration dialog box.
Parameter source	Specifies whether you configure this parameter using the Configuration Dialog Box or a Terminal on the simulation diagram. The default value is Configuration Dialog Box . If you select Terminal , LabVIEW displays an input for that parameter on the simulation diagram, and you can wire values to that input to configure this function

	<p>programmatically. If you select Configuration Dialog Box, LabVIEW removes that input from the simulation diagram. You then must set the value for this parameter inside the configuration dialog box.</p>
sample period (s)	<p>Specifies the length of the discrete time step, in seconds, of this function. If you enter a value of –1, this function inherits the discrete time step you specify for the simulation diagram. Otherwise, the value of sample period (s) must be a multiple of the discrete time step you specify for the simulation diagram. The default value of sample period (s) is 1.</p>
sample skew (s)	<p>Specifies the length of time by which you want to delay the execution of this function. The default value is 0. The value of this parameter must satisfy the following relationship:</p> $0 \leq \text{sample skew (s)} \leq \text{sample period (s)}$
discrete integrator	<p>Specifies the discrete integration method this function uses. You can choose from the following options:</p> <ul style="list-style-type: none"> - backward—Backward rectangular (Euler) integration method. - forward—(Default) Forward rectangular (Euler) integration method. - trapezoidal—Trapezoidal integration method. <p>Refer to the Details section for information about how these options affect the feedthrough behavior of this function.</p>
initial condition	<p>Specifies the initial output of the integrator on the first time step. The default value is 0.</p>
limit type	<p>Specifies the limit behavior of the integral. You can choose from the following options:</p> <ul style="list-style-type: none"> - upper—The integrator output saturates at the upper limit. The output can drop below the lower limit. - lower—The integrator output saturates at the lower limit. The output can exceed the upper

	<p>limit.</p> <ul style="list-style-type: none"> - both—The integrator output saturates at the lower limit and the upper limit. - none—(Default) The function does not limit the integrator output.
lower limit	Specifies the lower limit of the output. The LabVIEW Control Design and Simulation Module uses this parameter only if you set limit type to lower or both . The default value of lower limit is 0.
upper limit	Specifies the upper limit of the output. The Control Design and Simulation Module uses this parameter only if you set limit type to upper or both . The default value of upper limit is 0.
reset?	Sets the discrete states to the reset condition , when TRUE.
reset condition	Specifies the values to which to reset the discrete states when reset? is TRUE.

Block Diagram Inputs

Parameter	Description
sample period (s)	Specifies the length of the discrete time step, in seconds, of this function. If you enter a value of –1, this function inherits the discrete time step you specify for the simulation diagram. Otherwise, the value of sample period (s) must be a multiple of the discrete time step you specify for the simulation diagram. The default value of sample period (s) is 1.
sample skew (s)	Specifies the length of time by which you want to delay the execution of this function. The default value is 0. The value of this parameter must satisfy the following relationship: $0 \leq \text{sample skew (s)} \leq \text{sample period (s)}$
discrete integrator	Specifies the discrete integration method this function uses. You can choose from the following options: <ul style="list-style-type: none"> - backward—Backward rectangular (Euler) integration method. - forward—(Default) Forward rectangular (Euler) integration method. - trapezoidal—Trapezoidal integration method. Refer to the Details section for information about how these options affect the feedthrough behavior of this function.
initial condition	Specifies the initial output of the integrator on the first time step. The default value is 0.
limit type	Specifies the limit behavior of the integral. You can choose from the following options: <ul style="list-style-type: none"> - upper—The integrator output saturates at the upper limit. The output can drop below the lower limit. - lower—The integrator output saturates at the lower limit. The output can exceed the upper limit. - both—The integrator output saturates at the lower limit and the upper limit. - none—(Default) The function does not limit the integrator output.

lower limit	Specifies the lower limit of the output. The LabVIEW Control Design and Simulation Module uses this parameter only if you set limit type to lower or both . The default value of lower limit is 0.
upper limit	Specifies the upper limit of the output. The Control Design and Simulation Module uses this parameter only if you set limit type to upper or both . The default value of upper limit is 0.
reset?	Sets the discrete states to the reset condition , when TRUE.
reset condition	Specifies the values to which to reset the discrete states when reset? is TRUE.
input	Specifies the input to the system.

Block Diagram Outputs

Parameter	Description
output	Returns the integral of the input signal over time. This integral also is the state value.
limited?	Returns TRUE if the output of this function saturates at the upper limit or lower limit . limited? returns FALSE if you specified a value of none for the limit type parameter or if the value of the output is between the upper limit and the lower limit .

Discrete Integrator Details

This function uses the following equations to calculate the output:

Forward Rectangular (Euler) Integration:

$$y(t_k) = u(t_{k-1}) * T + y(t_{k-1})$$

Backward Rectangular (Euler) Integration:

$$y(t_k) = u(t_k) * T + y(t_{k-1})$$

Trapezoidal Integration:

$$y(t_k) = \frac{(u(t_k) + u(t_{k-1})) * T}{2} + y(t_{k-1})$$

where u is the input

$$t_k = \text{initial time} + \text{sample skew (s)} + k * \text{sample period (s)}, \text{ for } k = 0, 1, 2, \dots$$

t is the current simulation time

y is the output

T is the **sample period (s)** of this function

When you select the **Vector** instance of this function, the **input** and **output** parameters must be the same size as the **initial condition** parameter.

All other Discrete Integrator function parameters must be either the same size as the **initial condition** parameter or of size one. If the parameter array is of size one, the function assumes that array[i] equals array[0] for all values of i .

Feedthrough Behavior

The value you specify for the **discrete integrator** parameter determines the [feedthrough behavior](#) of this function.

- If you select **backward** or **trapezoidal**, the all input/output pairs have direct feedthrough behavior.
- If you select **forward**, the **input** input has indirect feedthrough to the **output** output. All other input/output pairs have direct feedthrough behavior.

When you use the configuration dialog box to specify a value for the **discrete integrator** parameter, LabVIEW verifies that the feedthrough behavior is correct. For example, if you set the **Feedthrough** parameter to **Indirect**, and you set the value of the **discrete integrator** parameter to **trapezoidal**, LabVIEW changes the **Feedthrough** parameter to **Direct**.

If you specify a value for the **discrete integrator** parameter programmatically by wiring values to the parameter terminals, LabVIEW does not adjust the feedthrough behavior for you. You must ensure that you specify the proper feedthrough behavior for the value of the **discrete integrator** parameter that you specify.

Example

Refer to the SimEx discrete integrator VI in the labview\examples\Control and Simulation\Simulation\Discrete Linear directory for an example of using the Discrete Integrator VI.

- [Open example](#)
- [Browse related examples](#)

Discrete Kalman Filter Function

Owning Palette: [Discrete Linear Systems Functions](#)

Installed With: Control Design and Simulation Module

Implements a discrete-time, linear time-variant, recursive Kalman filter. You define the system by specifying the stochastic state-space model and noise model as well as the inputs and outputs to the system. The Discrete Kalman Filter function calculates the predicted state estimates $\hat{x}(k+1|k)$, the corrected state estimates $\hat{x}(k|k)$, the corresponding gains used to calculate these estimates, and the associated estimation error covariances corresponding to these estimates. This function also calculates the estimated output $\hat{y}(k)$.

Refer to Chapter 16, *Using Stochastic System Models*, of the [LabVIEW Control Design User Manual](#) for information about using this function.

Details

[Dialog Box Options](#)

[Block Diagram Inputs](#)

[Block Diagram Outputs](#)

■ Place on the block diagram ■ Find on the **Functions** palette

Dialog Box Options

Parameter	Description
Polymorphic instance	<p>Specifies the behavior of this function:</p> <ul style="list-style-type: none"> • Pred gain, w/ check—(Default) Performs prediction and correction with prediction gain; checks the model parameters. • Pred gain, w/o check—Performs prediction and correction with prediction gain; does not check the model parameters. • No pred gain, w/ check—Performs prediction and correction without prediction gain; checks the model parameters. • No pred gain, w/o check—Performs prediction and correction without prediction gain; does not check the model parameters. • No corr, w/ check—Performs only prediction with prediction gain; checks the model parameters. • No corr, w/o check—Performs only prediction with prediction gain; does not check the model parameters. <p>Refer to the Details section for a description of the equations this function uses to calculate the outputs for each of these polymorphic instances.</p>
Feedthrough	<p>Configures the function to be either a Direct or Indirect feedthrough function. Enable this control by selecting Stochastic State-Space Model from the Parameters list and then selecting Terminal from the Parameter source pull-down menu. If you select Configuration Dialog Box from the Parameter source pull-down menu, LabVIEW disables this control and calculates the feedthrough behavior automatically.</p> <p>If you specify Direct feedthrough, the transmission matrix D in the Stochastic State-Space Model is non-zero. If you specify Indirect feedthrough, the transmission matrix D in the Stochastic State-Space Model is zero. You can</p>

	<p>use the Corrected State Estimate $\hat{x}(k k)$ to design the input $u(k)$ only if you specify Indirect feedthrough.</p> <p>Refer to the Details section for a description of the parameters that determine the feedthrough behavior of this function.</p>
Parameters	<p>Lists all the parameters associated with this function. Select a parameter from this list to configure the parameter. When you select a parameter, the parameter and its associated Parameter source control appear in the Parameter Information section of the configuration dialog box.</p>
Preview	<p>Displays a graphical preview, if available, of the function output or configuration.</p>
Parameter Information	<p>Contains the parameters you can configure for this function. You must select a parameter from the Parameters list to make that parameter and its associated Parameter source control visible in the Parameter Information section of the configuration dialog box.</p>
Parameter source	<p>Specifies whether you configure this parameter using the Configuration Dialog Box or a Terminal on the simulation diagram. The default value is Configuration Dialog Box. If you select Terminal, LabVIEW displays an input for that parameter on the simulation diagram, and you can wire values to that input to configure this function programmatically. If you select Configuration Dialog Box, LabVIEW removes that input from the simulation diagram. You then must set the value for this parameter inside the configuration dialog box.</p>
skew	<p>Specifies the length of time by which you want to delay the execution of this function. The default value is 0.</p>
Initial State Estimate xhat(0 -1)	<p>Specifies the initial states from which this function begins estimating the model states. If you do not specify a value for this parameter and you select one of the w/ check instances of this function, Initial State Estimate xhat(0 -1) is a vector of zeros.</p>

Initial Estimation Error Covariance $P(0 1)$	Specifies the initial covariance matrix of the estimation error. If you do not specify a value for this parameter and you select one of the w/ check instances of this function, Initial Estimation Error Covariance $P(0 1)$ is the identity matrix. Initial Estimation Error Covariance $P(0 1)$ must be symmetric and positive semi-definite such that $P(0 1) = P^T(0 1) \geq 0$.
Initialize?	Specifies whether to restart the calculation from any initial values you provide. The default is FALSE.
Second-Order Statistics Noise Model	Specifies a mathematical representation of the noise model of a stochastic state-space model. You can create a noise model using the CD Construct Noise Model VI . <ul style="list-style-type: none"> • E{w}—Specifies the expected value or mean of the process noise vector. The default is 0. • Q—Specifies the covariance matrix of the process noise vector. The default is 0.01. • E{v}—Specifies the expected value or mean of the measurement noise vector. The default is 0. • R—Specifies the covariance matrix of the measurement noise vector. The default is 0.1. • N—Specifies the cross-covariance matrix between the process noise vector and the measurement noise vector. If these noise vectors are uncorrelated, N is a matrix of zeros. The default is 0.
Stochastic State-Space Model	Specifies a mathematical representation of a stochastic system. You can construct a stochastic state-space model using the CD Construct Stochastic Model VI . <ul style="list-style-type: none"> • A—Specifies the system matrix that describes the dynamics of the states of the system. • B—Specifies the input matrix that relates the inputs to the states. • G—Specifies the matrix that relates the process noise vector to the model states. • C—Specifies the output matrix that relates the

outputs to the states.

- **D**—Specifies the transmission matrix that relates the inputs to the outputs.
- **H**—Specifies the matrix that relates the process noise vector to the model outputs.
- **Sampling Time (s)**—Specifies the sampling time of the system model and determines whether the model represents a continuous-time or discrete-time system. If the model represents a continuous-time system, **Sampling Time (s)** must equal zero. If the model represents a discrete-time system, **Sampling Time (s)** must be greater than zero and equal to the sampling rate, in seconds, of the discrete system. The default is –1. If you specify a value of –1, this function inherits the discrete time step you specify for the simulation diagram. Otherwise, the value of **Sampling Time (s)** must be a multiple of the discrete time step you specify for the simulation diagram. This limitation applies only if you place this function inside a [Simulation Loop](#).



Note If you use the inputs to create a continuous-time system, setting the **Sampling Time (s)** to a value greater than zero does not yield the discrete-time equivalent of the system. You must use the [CD Convert Continuous Stochastic to Discrete](#) VI to convert the continuous-time system to the discrete-time equivalent of the system.

Block Diagram Inputs

Parameter	Description
skew	Specifies the length of time by which you want to delay the execution of this function. The default value is 0.
Initial State Estimate $\hat{x}(0 1)$	Specifies the initial states from which this function begins estimating the model states. If you do not specify a value for this parameter and you select one of the wl check instances of this function, Initial State Estimate $\hat{x}(0 1)$ is a vector of zeros.
Initial Estimation Error Covariance $P(0 1)$	Specifies the initial covariance matrix of the estimation error. If you do not specify a value for this parameter and you select one of the wl check instances of this function, Initial Estimation Error Covariance $P(0 1)$ is the identity matrix. Initial Estimation Error Covariance $P(0 1)$ must be symmetric and positive semi-definite such that $P(0 1) = P^T(0 1) \geq 0$.
Initialize?	Specifies whether to restart the calculation from any initial values you provide. The default is FALSE.
Second-Order Statistics Noise Model	Specifies a mathematical representation of the noise model of a stochastic state-space model. You can create a noise model using the CD Construct Noise Model VI . <ul style="list-style-type: none"> • E{w}—Specifies the expected value or mean of the process noise vector. The default is 0. • Q—Specifies the covariance matrix of the process noise vector. The default is 0.01. • E{v}—Specifies the expected value or mean of the measurement noise vector. The default is 0. • R—Specifies the covariance matrix of the measurement noise vector. The default is 0.1. • N—Specifies the cross-covariance matrix between the process noise vector and the measurement noise vector. If these noise vectors are uncorrelated, N is a matrix of zeros. The default is 0.
Stochastic	Specifies a mathematical representation of a stochastic

State-Space Model

system. You can construct a stochastic state-space model using the [CD Construct Stochastic Model VI](#).

- **A**—Specifies the system matrix that describes the dynamics of the states of the system.
- **B**—Specifies the input matrix that relates the inputs to the states.
- **G**—Specifies the matrix that relates the process noise vector to the model states.
- **C**—Specifies the output matrix that relates the outputs to the states.
- **D**—Specifies the transmission matrix that relates the inputs to the outputs.
- **H**—Specifies the matrix that relates the process noise vector to the model outputs.
- **Sampling Time (s)**—Specifies the sampling time of the system model and determines whether the model represents a continuous-time or discrete-time system. If the model represents a continuous-time system, **Sampling Time (s)** must equal zero. If the model represents a discrete-time system, **Sampling Time (s)** must be greater than zero and equal to the sampling rate, in seconds, of the discrete system. The default is –1. If you specify a value of –1, this function inherits the discrete time step you specify for the simulation diagram. Otherwise, the value of **Sampling Time (s)** must be a multiple of the discrete time step you specify for the simulation diagram. This limitation applies only if you place this function inside a [Simulation Loop](#).



Note If you use the inputs to create a continuous-time system, setting the **Sampling Time (s)** to a value greater than zero does not yield the discrete-time equivalent of the system. You must use the [CD Convert Continuous Stochastic to Discrete VI](#) to convert the continuous-time

	system to the discrete-time equivalent of the system.
Output $y(k)$	Specifies measurements made on the Stochastic State-Space Model . You can use the Discrete Stochastic State-Space function to simulate the model and obtain this parameter. You also can wire in measurements made from a hardware sensor.
input $u(k)$	Specifies the control action this function applies to the model.

Block Diagram Outputs

Parameter	Description
Estimated Output $yhat(k)$	Returns the estimated model output at time k .
Corrected State Estimate $xhat(k k)$	Returns the corrected Kalman state estimate at time k , given all measurements up to and including time k . The length of this vector is equal to the number of model states. This parameter is not available if you select one of the No corr instances of this function.
Predicted State Estimate $xhat(k+1 k)$	Returns the predicted state estimate for the next time step $k + 1$, given all measurements up to and including time k . The length of this vector is equal to the number of model states.
Kalman Filter Gain $M(k)$	Returns the Kalman filtered gain matrix this function uses to calculate the Corrected State Estimate $xhat(k k)$. This parameter is not available if you select one of the No corr instances of this function.
Kalman Predictor Gain $L(k)$	Returns the gain matrix this function uses to calculate the Predicted State Estimate $xhat(k+1 k)$. This parameter is not available if you select one of the No pred gain instances of this function.
Filter Error Covariance $P(k k)$	Returns the covariance matrix of the estimation error associated with the Corrected State Estimate $xhat(k k)$. This parameter is not available if you select one of the No corr instances of this function.
Prediction Error Covariance $P(k+1 k)$	Returns the covariance matrix of the estimation error associated with the Predicted State Estimate $xhat(k+1 k)$, given all measurements up to and including time k .

Discrete Kalman Filter Details

This function adapts to changes in the **Stochastic State-Space Model** and the **Second-Order Statistics Noise Model** as long as the model dimensions do not change. Therefore, you can use this function with linear time-variant (LTV) models.

This function uses the **Second-Order Statistics Noise Model** to obtain the values of $E\{\mathbf{w}(k)\}$, $E\{\mathbf{v}(k)\}$, $\mathbf{Q}(k)$, $\mathbf{R}(k)$, and $\mathbf{N}(k)$. The following equations define these terms:

$$\mathbf{Q}(k)\delta_{kl} = E\{\mathbf{w}(k) \cdot \mathbf{w}^T(l)\} - E\{\mathbf{w}(k)\} \cdot E^T\{\mathbf{w}(l)\}$$

$$\mathbf{R}(k)\delta_{kl} = E\{\mathbf{v}(k) \cdot \mathbf{v}^T(l)\} - E\{\mathbf{v}(k)\} \cdot E^T\{\mathbf{v}(l)\}$$

$$\mathbf{N}(k)\delta_{kl} = E\{\mathbf{w}(k) \cdot \mathbf{v}^T(l)\} - E\{\mathbf{w}(k)\} \cdot E^T\{\mathbf{v}(l)\}$$

where δ_{kl} is the Kronecker delta function. The following equations defines this function: $\delta_{kl} = 1$ when $k = l$; $\delta_{kl} = 0$ when $k \neq l$.

$\mathbf{w}(k)$ is the process noise vector.

$\mathbf{v}(k)$ is the measurement noise vector.

$\mathbf{Q}(k)$ is the covariance matrix of $\mathbf{w}(k)$.

$\mathbf{R}(k)$ is the covariance matrix of $\mathbf{v}(k)$.

$\mathbf{N}(k)$ is the cross-covariance matrix between $\mathbf{w}(k)$ and $\mathbf{v}(k)$. If these noise vectors are uncorrelated, $\mathbf{N}(k)$ is a matrix of zeros.

$E\{\}$ denotes the expected value or the mean of the enclosed term(s).

The noise covariance matrices must satisfy the following conditions:

$$\mathbf{Q}(k) = \mathbf{Q}^T(k) \geq 0$$

$$\mathbf{R}(k) = \mathbf{R}^T(k) > 0$$

$$\begin{bmatrix} \mathbf{Q} & \mathbf{N} \\ \mathbf{N}^T & \mathbf{R} \end{bmatrix} \geq 0$$

$$\begin{bmatrix} \bar{\mathbf{Q}} & \bar{\mathbf{N}} \\ \bar{\mathbf{N}}^T & \bar{\mathbf{R}} \end{bmatrix} \geq 0$$

$$\bar{\mathbf{Q}} = \mathbf{G}(k)\mathbf{Q}(k)\mathbf{G}^T(k)$$

$$\bar{\mathbf{N}} = \mathbf{G}(k)\mathbf{N}(k) + \mathbf{G}(k)\mathbf{Q}(k)\mathbf{H}^T(k)$$

$$\bar{R}(k) = \mathbf{C}(k)\mathbf{P}(k|k-1)\mathbf{C}^T(k) + \mathbf{R}(k) + \mathbf{H}(k)\mathbf{Q}(k)\mathbf{H}(k)^T + \mathbf{H}(k)\mathbf{N}(k) + \mathbf{N}^T(k)\mathbf{H}^T(k) > 0$$

This function assumes the process noise and measurement noise vectors to be temporally uncorrelated between time steps. This function also assumes the **Initial State Estimate xhat(0|−1)** to be uncorrelated with the noise vectors. If the noise vectors are Gaussian-distributed, this function is an optimal minimum mean square error (MMSE) estimator. However, if these vectors are not Gaussian-distributed, this function is an optimal affine MMSE estimator.

Feedthrough Behavior

The value you specify for the **D** matrix of the **Stochastic State-Space Model** parameter determines the feedthrough behavior of this function.

- If the **D** matrix is nonzero, all input/output pairs have direct feedthrough behavior.
- If the **D** matrix is zero, the following input/output pairs have indirect feedthrough behavior.
 - **Input u(k) — Corrected State Estimate $\hat{x}(k|k)$**
 - **Input u(k) — Predicted State Estimate $\hat{x}(k|k-1)$**

The remaining input/output pairs have direct feedthrough behavior.

When you use the configuration dialog box to configure the **D** matrix, LabVIEW verifies that the feedthrough behavior is correct. For example, if you set the **Feedthrough** parameter to **Indirect**, and you set the **D** matrix to nonzero, LabVIEW changes the **Feedthrough** parameter to **Direct**.

If you specify the value of the **D** matrix programmatically by wiring a value to the parameter terminal, LabVIEW does not adjust the feedthrough behavior for you. You must ensure that you specify the proper feedthrough behavior for the value of the **D** matrix that you specify.

Kalman Filter Structures

This function uses the following equations to calculate the outputs for each of the polymorphic instances.

Pred Gain Instances—Correction Stage

$$xhat(k|k) = xhat(k|k-1) + M(k) \cdot [y(k) - yhat(k)]$$

$$yhat(k) = C(k) \cdot xhat(k|k-1) + D(k) \cdot u(k)$$

$$M(k) = P(k|k-1) \cdot C^T(k) \cdot [C(k) \cdot P(k|k-1) \cdot C^T(k) + H(k)Q(k)H^T(k) + H(k)N(k) + N^T(k)H^T(k) + R(k)]^{-1}$$

$$P(k|k) = P(k|k-1) - M(k) \cdot C(k) \cdot P(k|k-1)$$

Pred Gain Instances—Prediction Stage

$$xhat(k+1|k) = A(k) \cdot xhat(k|k-1) + B(k) \cdot u(k) + L(k) \cdot [y(k) - yhat(k)]$$

$$yhat(k) = C(k) \cdot xhat(k|k-1) + D(k) \cdot u(k)$$

$$L(k) = [A(k) \cdot P(k|k-1) \cdot C^T(k) + G(k)Q(k)H^T(k) + G(k)N(k)] \cdot [C(k) \cdot P(k|k-1) \cdot C^T(k) + H(k)Q(k)H^T(k) + H(k)N(k) + N^T(k)H^T(k) + R(k)]^{-1}$$

$$P(k+1|k) = [A(k) \cdot P(k|k-1) \cdot A^T(k) + G(k)Q(k)G^T(k)] - L(k) \cdot [A(k) \cdot P(k|k-1) \cdot C^T(k) + G(k)Q(k)H^T(k) + G(k)N(k)]^T$$

No Pred Gain Instances—Correction Stage

$$xhat(k|k) = xhat(k|k-1) + M(k) \cdot [y(k) - yhat(k)]$$

$$yhat(k) = C(k) \cdot xhat(k|k-1) + D(k) \cdot u(k)$$

$$M(k) = P(k|k-1) \cdot C^T(k) \cdot [C(k) \cdot P(k|k-1) \cdot C^T(k) + H(k)Q(k)H^T(k) + H(k)N(k) + N^T(k)H^T(k) + R(k)]^{-1}$$

$$P(k|k) = P(k|k-1) - M(k) \cdot C(k) \cdot P(k|k-1)$$

No Pred Gain Instances—Prediction Stage

$$xhat(k+1|k) = A(k)xhat(k|k-1) + B(k)u(k)$$

No Corr Instances—Prediction Stage

$$xhat(k+1|k) = A(k) \cdot xhat(k|k-1) + B(k) \cdot u(k) + L(k) \cdot [y(k) - yhat(k)]$$

$$yhat(k) = C(k) \cdot xhat(k|k-1) + D(k) \cdot u(k)$$

$$L(k) = [A(k) \cdot P(k|k-1) \cdot C^T(k) + G(k)Q(k)H^T(k) + G(k)N(k)] \cdot [C(k) \cdot P(k|k-1) \cdot C^T(k) + H(k)Q(k)H^T(k) + H(k)N(k) + N^T(k)H^T(k) + R(k)]^{-1}$$

$$\mathsf{P}(k\!+\!1|k) = [\mathbf{A}(k)\cdot \mathsf{P}(k|k\!-\!1)\cdot \mathbf{A}^\mathsf{T}(k) + \mathbf{G}(k)\mathbf{Q}(k)\mathbf{G}^\mathsf{T}(k)] - \mathsf{L}(k)\cdot[\mathbf{A}(k)\cdot \mathsf{P}(k|k\!-\!1)\cdot \mathbf{C}^\mathsf{T}(k) + \mathbf{G}(k)\mathbf{Q}(k)\mathbf{H}^\mathsf{T}(k) + \mathbf{G}(k)\mathbf{N}(k)]^\mathsf{T}$$

Discrete Observer Function

Owning Palette: [Discrete Linear Systems Functions](#)

Installed With: Control Design and Simulation Module

Implements a discrete-time observer for a linear state-space system model.

[Details](#)

[Dialog Box Options](#)

[Block Diagram Inputs](#)

[Block Diagram Outputs](#)

■ Place on the block diagram ■ Find on the **Functions** palette

Dialog Box Options

Parameter	Description
Polymorphic instance	Specifies whether this function implements a current observer and corrects state estimates made at a previous time step, or whether this function implements a predictive observer and calculates the estimated states for the next time step $k + 1$. This option also specifies whether the observer checks the model parameters. The default is Current, w/ check . Refer to Chapter 15, <i>Estimating Model States</i> , of the LabVIEW Control Design User Manual for information about using current and predictive observers.
Feedthrough	Configures the function to be either a Direct or Indirect feedthrough function. Enable this control by selecting State-Space Model from the Parameters list and then selecting Terminal from the Parameter source pull-down menu. If you select Configuration Dialog Box from the Parameter source pull-down menu, LabVIEW disables this control and calculates the feedthrough behavior automatically. You can use the Corrected State Estimate $xhat(k k)$ to design the Input $u(k)$ only if you specify Indirect feedthrough. Refer to the Details section for a description of the parameters that determine the feedthrough behavior of this function.
Parameters	Lists all the parameters associated with this function. Select a parameter from this list to configure the parameter. When you select a parameter, the parameter and its associated Parameter source control appear in the Parameter Information section of the configuration dialog box.
Preview	Displays a graphical preview, if available, of the function output or configuration.

Parameter Information	Contains the parameters you can configure for this function. You must select a parameter from the Parameters list to make that parameter and its associated Parameter source control visible in the Parameter Information section of the configuration dialog box.
Parameter source	Specifies whether you configure this parameter using the Configuration Dialog Box or a Terminal on the simulation diagram. The default value is Configuration Dialog Box . If you select Terminal , LabVIEW displays an input for that parameter on the simulation diagram, and you can wire values to that input to configure this function programmatically. If you select Configuration Dialog Box , LabVIEW removes that input from the simulation diagram. You then must set the value for this parameter inside the configuration dialog box.
reset?	Specifies whether to restart the calculation from any initial values you provide. The default is FALSE.
sample skew (s)	Specifies the length of time by which you want to delay the execution of this function. The default value is 0. The value of this parameter must satisfy the following relationship: $0 \leq \text{sample skew (s)} \leq \text{Sampling Time}$
Initial State Estimate xhat(0 -1)	Specifies the initial states from which this function begins estimating the model states. If you do not specify a value for this parameter and you select one of the w/ check instances of this function, Initial State Estimate xhat(0 -1) is a vector of zeros.
Observer Gain	Specifies the estimator gain matrix this VI applies to the difference between the observed output and the estimated output, which is Output y(k) – Estimated Output yhat(k) . You can use the CD Pole Placement VI or the CD Ackermann VI to calculate the Observer Gain . The Observer Gain for a current observer relates to the Observer Gain for a predictive observer through the

	<p>following relationship:</p> <p>Observer Gain (Current) = $A^{-1} \cdot \text{Observer Gain (Predictive)}$, where A is the state matrix of the State-Space Model.</p>
State-Space Model	<p>Specifies the mathematical representation of and information about the system for which this function implements the observer.</p> <ul style="list-style-type: none"> • Model name—Specifies the name of the state-space model. • Sampling Time—Specifies the sampling time of the system model and determines whether the model represents a continuous-time or discrete-time system. If the model represents a continuous-time system, Sampling Time must equal zero. If the model represents a discrete-time system, Sampling Time must be greater than zero and equal to the sampling rate, in seconds, of the discrete system. The default is –1. If you enter a value of –1, this function inherits the discrete time step you specify for the simulation diagram. Otherwise, the value of Sampling Time must be a multiple of the discrete time step you specify for the simulation diagram. This limitation applies only when you place this function inside a Simulation Loop. • A—Specifies the $n \times n$ state matrix of the given system. • B—Specifies the $n \times m$ input matrix of the given system. • C—Specifies the $r \times n$ output matrix of the given system. • D—Specifies the $r \times m$ direct transmission matrix of the given system. <p style="margin-left: 20px;">where m is the number of inputs n is the number of states</p>

r is the number of outputs

Block Diagram Inputs

Parameter	Description
reset?	Specifies whether to restart the calculation from any initial values you provide. The default is FALSE.
sample skew (s)	Specifies the length of time by which you want to delay the execution of this function. The default value is 0. The value of this parameter must satisfy the following relationship: $0 \leq \text{sample skew (s)} \leq \text{Sampling Time}$
Initial State Estimate $xhat(0 1)$	Specifies the initial states from which this function begins estimating the model states. If you do not specify a value for this parameter and you select one of the wl check instances of this function, Initial State Estimate $xhat(0 1)$ is a vector of zeros.
Observer Gain	Specifies the estimator gain matrix this VI applies to the difference between the observed output and the estimated output, which is Output $y(k)$ – Estimated Output $yhat(k)$. You can use the CD Pole Placement VI or the CD Ackermann VI to calculate the Observer Gain . The Observer Gain for a current observer relates to the Observer Gain for a predictive observer through the following relationship: $\text{Observer Gain (Current)} = A^{-1} \cdot \text{Observer Gain (Predictive)}$, where A is the state matrix of the State-Space Model .
State-Space Model	Specifies the mathematical representation of and information about the system for which this function calculates the estimated states.
Output $y(k)$	Specifies the measurements a sensor makes on the State-Space Model . You also can use the Discrete State-Space function to simulate the behavior of a state-space model.
Input $u(k)$	Specifies the control action this VI applies to the model.

Block Diagram Outputs

Parameter	Description
Estimated Output $yhat(k)$	Returns the estimated output of the State-Space Model .
Predicted State Estimate $xhat(k+1 k)$	Returns the predicted state estimates of the State-Space Model for the next time step $k + 1$. The length of this vector is equal to the number of model states.
Corrected State Estimate $xhat(k k)$	Returns the corrected state estimates at time k . The length of this vector is equal to the number of model states. This output appears only if you select Current, w/ check or Current, w/o check from the Polymorphic instance pull-down menu.
Predicted State Estimate $xhat(k k-1)$	Returns the estimated states of the State-Space Model estimated at the previous time step. This output appears only if you select Predictive, w/ check or Predictive, w/o check from the Polymorphic instance pull-down menu.

Discrete Observer Details

Current Observer

At each time step k , this function calculates the **Corrected State Estimate** $\hat{x}(k|k)$ using the **Estimated Output** $y(k)$, any control action **Input** $u(k)$, and the **Predicted State Estimate** $\hat{x}(k|k-1)$ calculated at the previous time step. This function also calculates the estimated states for the next time step $k + 1$, **Predicted State Estimate** $\hat{x}(k+1|k)$.

This VI uses the following equations to calculate the outputs.

$$y(k) = \mathbf{C}(k) \cdot \hat{x}(k|k-1) + \mathbf{D}(k) \cdot u(k)$$

where \mathbf{C} and \mathbf{D} are the output and direct feedthrough matrices, respectively, of the **State-Space Model**.

$$\hat{x}(k|k) = \hat{x}(k|k-1) + \mathbf{Lc} [y(k) - y(k)]$$

where \mathbf{Lc} is the **Observer Gain**.

$$\hat{x}(k+1|k) = \mathbf{A}(k) \cdot \hat{x}(k|k) + \mathbf{B}(k) \cdot u(k)$$

where \mathbf{A} and \mathbf{B} are the state and input matrices, respectively, of the **State-Space Model**.

Predictive Observer

At each time step k , this function calculates the **Predicted State Estimate** $\hat{x}(k+1|k)$ using the **Estimated Output** $\hat{y}(k)$ and the **Predicted State Estimate** $\hat{x}(k|k-1)$.

The following equations define the outputs of this function:

$$\hat{y}(k) = \mathbf{C}(k) \cdot \hat{x}(k|k-1) + \mathbf{D}(k) \cdot u(k)$$

$$\hat{x}(k+1|k) = \mathbf{A}(k) \cdot \hat{x}(k|k-1) + \mathbf{B}(k) \cdot u(k) + \mathbf{Lp}[\hat{y}(k) - \hat{y}(k)]$$

where \mathbf{Lp} is the **Observer Gain**.

Feedthrough Behavior

The value you specify for the **D** matrix of the **Stochastic State-Space Model** parameter determines the feedthrough behavior of this function.

- If the **D** matrix is nonzero, all input/output pairs have direct feedthrough behavior.
- If the **D** matrix is zero, the following input/output pairs have indirect feedthrough behavior.
 - **Input u(k) — Corrected State Estimate $\hat{x}(k|k)$**
 - **Input u(k) — Predicted State Estimate $\hat{x}(k|k-1)$**

The remaining input/output pairs have direct feedthrough behavior.

When you use the configuration dialog box to configure the **D** matrix, LabVIEW verifies that the feedthrough behavior is correct. For example, if you set the **Feedthrough** parameter to **Indirect**, and you set the **D** matrix to nonzero, LabVIEW changes the **Feedthrough** parameter to **Direct**.

If you specify the value of the **D** matrix programmatically by wiring a value to the parameter terminal, LabVIEW does not adjust the feedthrough behavior for you. You must ensure that you specify the proper feedthrough behavior for the value of the **D** matrix that you specify.

Discrete State-Space Function

Owning Palette: [Discrete Linear Systems Functions](#)

Installed With: Control Design and Simulation Module

Implements a system model in discrete state-space form. You define the system model by specifying the input, output, state, and direct transmission matrices.

[Details](#) [Example](#)

[Dialog Box Options](#)

[Block Diagram Inputs](#)

[Block Diagram Outputs](#)

■ Place on the block diagram ■ Find on the **Functions** palette

Dialog Box Options

Parameter	Description
Polymorphic instance	Specifies whether this function is single-input single-output (SISO) or multiple-input multiple-output (MIMO). The default value is SISO .
Feedthrough	Configures the function to be either a Direct or Indirect feedthrough function. Enable this control by selecting a parameter from the Parameters list and then selecting Terminal from the Parameter source pull-down menu. If you select Configuration Dialog Box from the Parameter source pull-down menu, LabVIEW disables this control and calculates the feedthrough behavior automatically. Refer to the Details section for a description of the parameters that determine the feedthrough behavior of this function.
Parameters	Lists all the parameters associated with this function. Select a parameter from this list to configure the parameter. When you select a parameter, the parameter and its associated Parameter source control appear in the Parameter Information section of the configuration dialog box.
Preview	Displays a graphical preview, if available, of the function output or configuration.
Parameter Information	Contains the parameters you can configure for this function. You must select a parameter from the Parameters list to make that parameter and its associated Parameter source control visible in the Parameter Information section of the configuration dialog box.
Parameter source	Specifies whether you configure this parameter using the Configuration Dialog Box or a Terminal on the simulation diagram. The default value is Configuration Dialog Box . If you select Terminal , LabVIEW displays an input for that parameter on the simulation diagram, and

	<p>you can wire values to that input to configure this function programmatically. If you select Configuration Dialog Box, LabVIEW removes that input from the simulation diagram. You then must set the value for this parameter inside the configuration dialog box.</p>
State-Space	<p>Specifies the state-space model.</p> <ul style="list-style-type: none"> • Load Model—Loads model information from a data file. • Save Model—Saves model information to a data file. This file is compatible with the Control Design VIs and functions. • Copy to Clipboard—Copies the current model definition to the clipboard. From the clipboard, you can paste the model on the block diagram or into another configuration dialog box of the same model form. • Paste from Clipboard—Pastes model information from the clipboard to the configuration dialog box. • Model Dimensions—Use this section to specify the number of inputs, states, and outputs of the system model. <ul style="list-style-type: none"> - Inputs—Specifies the number of model inputs. - States—Specifies the number of model states. - Outputs—Specifies the number of model outputs. • A—Specifies the $n \times n$ state matrix of the given system. • B—Specifies the $n \times m$ input matrix of the given system. • C—Specifies the $r \times n$ output matrix of the given system. • D—Specifies the $r \times m$ direct transmission matrix of the given system.

	<p>where m is the number of inputs n is the number of states r is the number of outputs</p>
sample period (s)	<p>Specifies the length of the discrete time step, in seconds, of this function. The default value is –1.</p> <p>If you enter a value of –1, this function inherits the discrete time step you specify for the simulation diagram. Otherwise, the value of sample period (s) must be a multiple of the discrete time step you specify for the simulation diagram.</p> <p>This parameter is valid only when you place this function inside a Simulation Loop.</p>
sample skew (s)	<p>Specifies the length of time by which you want to delay the execution of this function. The default value is 0.</p> <p>The value of this parameter must satisfy the following relationship:</p> $0 \leq \text{sample skew (s)} \leq \text{sample period (s)}$ <p>This parameter is valid only when you place this function inside a Simulation Loop.</p>
initial state (x_0)	Specifies a vector of initial states for the system. This vector must be of length n , where n is the number of states.
reset?	Sets the model state(s) to the values you specify in reset state (xr) , when TRUE.
reset state (xr)	Specifies the values to which to set the model state(s) when reset? is TRUE. The default is 0.

Block Diagram Inputs

Parameter	Description
sample period (s)	<p>Specifies the length of the discrete time step, in seconds, of this function. The default value is –1.</p> <p>If you enter a value of –1, this function inherits the discrete time step you specify for the simulation diagram. Otherwise, the value of sample period (s) must be a multiple of the discrete time step you specify for the simulation diagram.</p> <p>This parameter is valid only when you place this function inside a Simulation Loop.</p>
sample skew (s)	<p>Specifies the length of time by which you want to delay the execution of this function. The default value is 0.</p> <p>The value of this parameter must satisfy the following relationship:</p> $0 \leq \text{sample skew (s)} \leq \text{sample period (s)}$ <p>This parameter is valid only when you place this function inside a Simulation Loop.</p>
initial state (x_0)	Specifies a vector of initial states for the system. This vector must be of length n , where n is the number of states.
reset?	Sets the model state(s) to the values you specify in reset state (x_r) , when TRUE.
reset state (x_r)	Specifies the values to which to set the model state(s) when reset? is TRUE. The default is 0.
input $u(k)$	Specifies the input to the system. input $u(k)$ must be a vector of length m , where m is the number of inputs.
State-Space	Specifies a state-space model. This input accepts either a block diagram constant or a model you created using the Control Design VIs and functions.

Block Diagram Outputs

Parameter	Description
state $x(k+1)$	Returns the values of the model state(s) at time $k + 1$. The length of this vector is equal to the number of model states.
output $y(k)$	Returns the current output of the system. This vector must be of length r , where r is the number of outputs.
state $x(k)$	Returns a one-dimensional array that specifies the internal states. This array must be of length n , where n is the number of states. Each element of the array represents $x[n]$.

Discrete State-Space Details

The following equations define the state-space system.

$$\mathbf{x}[t_{k+1}] = \Phi \mathbf{x}[t_k] + \Gamma \mathbf{u}[t_k]$$

$$\mathbf{y}[t_k] = \mathbf{C} \mathbf{x}[t_k] + \mathbf{D} \mathbf{u}[t_k]$$

where \mathbf{u} is the input vector

\mathbf{y} is the output vector

\mathbf{x} is the state vector

Φ , Γ , \mathbf{C} , \mathbf{D} are the state-space matrices. Refer to the **State-Space** parameter help for more information about these matrices.

If you place this function outside a Simulation Loop, $t_k = k * \text{Sampling Time (s)}$, for $k = 0, 1, 2, \dots$.

If you place this function inside a Simulation Loop, $t_k = \text{initial time} + \text{sample skew (s)} + k * \text{sample period (s)}$, for $k = 0, 1, 2, \dots$

Feedthrough Behavior

The value you specify for the **D** matrix of the **State-Space** parameter determines the feedthrough behavior of this function.

- If the **D** matrix is nonzero, all input/output pairs have direct feedthrough behavior.
- If the **D** matrix is zero, the following input/output pairs have indirect feedthrough behavior.
 - **input u(k) — output y(k)**
 - **input u(k) — state x(k)**

The remaining input/output pairs have direct feedthrough behavior.

When you use the configuration dialog box to configure the **D** matrix, LabVIEW verifies that the feedthrough behavior is correct. For example, if you set the **Feedthrough** parameter to **Indirect**, and you set the **D** matrix to nonzero, LabVIEW changes the **Feedthrough** parameter to **Direct**.

If you specify the value of the **D** matrix programmatically by wiring a value to the parameter terminal, LabVIEW does not adjust the feedthrough behavior for you. You must ensure that you specify the proper feedthrough behavior for the value of the **D** matrix that you specify.

Example

Refer to the SimEx discrete state space VI in the labview\examples\Control and Simulation\Simulation\Discrete Linear directory for an example of using the Discrete State-Space VI.

- Open example ■ Browse related examples

Discrete Stochastic State-Space Function

Owning Palette: [Discrete Linear Systems Functions](#)

Installed With: Control Design and Simulation Module

Implements a discrete-time, linear, stochastic state-space system. You define the system model by specifying the input, output, state, and direct transmission matrices. You also specify the matrices relating the process noise to the system states and outputs.

If you use the **Internal Noise** instance of this function, this function generates samples of the noise vectors using the model you wire to the **Second-Order Statistics Noise Model** input. If you use the **External Noise** instance of this function, you can use the [CD Correlated Gaussian Random Noise VI](#) to generate samples of the noise vectors.

[Details](#)

[Dialog Box Options](#)

[Block Diagram Inputs](#)

[Block Diagram Outputs](#)

- Place on the block diagram ■ Find on the **Functions** palette

Dialog Box Options

Parameter	Description
Polymorphic instance	Specifies whether this function uses internally-generated random samples of Gaussian-distributed noise vectors or externally-generated noise or disturbances. The default is Internal Noise .
Feedthrough	Configures the function to be either a Direct or Indirect feedthrough function. Enable this control by selecting Stochastic State-Space Model from the Parameters list and then selecting Terminal from the Parameter source pull-down menu. If you select Configuration Dialog Box from the Parameter source pull-down menu, LabVIEW disables this control and calculates the feedthrough behavior automatically. Refer to the Details section for a description of the parameters that determine the feedthrough behavior of this function.
Parameters	Lists all the parameters associated with this function. Select a parameter from this list to configure the parameter. When you select a parameter, the parameter and its associated Parameter source control appear in the Parameter Information section of the configuration dialog box.
Preview	Displays a graphical preview, if available, of the function output or configuration.
Parameter Information	Contains the parameters you can configure for this function. You must select a parameter from the Parameters list to make that parameter and its associated Parameter source control visible in the Parameter Information section of the configuration dialog box.
Parameter source	Specifies whether you configure this parameter using the Configuration Dialog Box or a Terminal on the simulation diagram. The default value is Configuration Dialog Box . If you select Terminal , LabVIEW displays an

	<p>input for that parameter on the simulation diagram, and you can wire values to that input to configure this function programmatically. If you select Configuration Dialog Box, LabVIEW removes that input from the simulation diagram. You then must set the value for this parameter inside the configuration dialog box.</p>
reset?	Specifies whether to restart the calculation from any initial values you provide. The default is FALSE.
Stochastic State-Space Model	<p>Specifies a mathematical representation of a stochastic system. You can construct a stochastic state-space model using the CD Construct Stochastic Model VI.</p> <ul style="list-style-type: none"> • A—Specifies the system matrix that describes the dynamics of the states of the system. • B—Specifies the input matrix that relates the inputs to the states. • G—Specifies the matrix that relates the process noise vector to the model states. • C—Specifies the output matrix that relates the outputs to the states. • D—Specifies the transmission matrix that relates the inputs to the outputs. • H—Specifies the matrix that relates the process noise vector to the model outputs. • Sampling Time (s)—Specifies the sampling time of the system model and determines whether the model represents a continuous-time or discrete-time system. If the model represents a continuous-time system, Sampling Time (s) must equal zero. If the model represents a discrete-time system, Sampling Time (s) must be greater than zero and equal to the sampling rate, in seconds, of the discrete system. The default is –1. If you specify a value of –1, this function inherits the discrete time step you specify for the simulation diagram. Otherwise, the value of Sampling Time (s) must be a multiple of the discrete time step you specify for the simulation

	<p>diagram. This limitation applies only if you place this function inside a Simulation Loop.</p> <p> Note If you use the inputs to create a continuous-time system, setting the Sampling Time (s) to a value greater than zero does not yield the discrete-time equivalent of the system. You must use the CD Convert Continuous Stochastic to Discrete VI to convert the continuous-time system to the discrete-time equivalent of the system.</p>
Second-Order Statistics Noise Model	<p>Specifies a mathematical representation of the noise model of a stochastic state-space model. You can create a noise model using the CD Construct Noise Model VI. This option is available only if you select Internal Noise from the Polymorphic instance pull-down menu.</p> <ul style="list-style-type: none"> • E{w}—Specifies the expected value or mean of the process noise vector. The default is 0. • Q—Specifies the covariance matrix of the process noise vector. The default is 0.01. • E{v}—Specifies the expected value or mean of the measurement noise vector. The default is 0. • R—Specifies the covariance matrix of the measurement noise vector. The default is 0.1. • N—Specifies the cross-covariance matrix between the process noise vector and the measurement noise vector. If these noise vectors are uncorrelated, N is a matrix of zeros. The default is 0.
sample skew (s)	<p>Specifies the length of time by which you want to delay the execution of this function. The default value is 0. The value of this parameter must satisfy the following relationship:</p> $0 \leq \text{sample skew (s)} \leq \text{Sampling Time}$
Initial State	Specifies the initial state of the model. If you do not

x(0)

specify a value for this parameter, this function assumes an initial state of 0.

Block Diagram Inputs

Parameter	Description
reset?	Specifies whether to restart the calculation from any initial values you provide. The default is FALSE.
Stochastic State-Space Model	<p>Specifies a mathematical representation of a stochastic system. You can construct a stochastic state-space model using the CD Construct Stochastic Model VI.</p> <ul style="list-style-type: none">• A—Specifies the system matrix that describes the dynamics of the states of the system.• B—Specifies the input matrix that relates the inputs to the states.• G—Specifies the matrix that relates the process noise vector to the model states.• C—Specifies the output matrix that relates the outputs to the states.• D—Specifies the transmission matrix that relates the inputs to the outputs.• H—Specifies the matrix that relates the process noise vector to the model outputs.• Sampling Time (s)—Specifies the sampling time of the system model and determines whether the model represents a continuous-time or discrete-time system. If the model represents a continuous-time system, Sampling Time (s) must equal zero. If the model represents a discrete-time system, Sampling Time (s) must be greater than zero and equal to the sampling rate, in seconds, of the discrete system. The default is –1. If you specify a value of –1, this function inherits the discrete time step you specify for the simulation diagram. Otherwise, the value of Sampling Time (s) must be a multiple of the discrete time step you specify for the simulation diagram. This limitation applies only if you place this function inside a Simulation Loop.

	 Note If you use the inputs to create a continuous-time system, setting the Sampling Time (s) to a value greater than zero does not yield the discrete-time equivalent of the system. You must use the CD Convert Continuous Stochastic to Discrete VI to convert the continuous-time system to the discrete-time equivalent of the system.
Second-Order Statistics Noise Model	<p>Specifies a mathematical representation of the noise model of a stochastic state-space model. You can create a noise model using the CD Construct Noise Model VI. This option is available only if you select Internal Noise from the Polymorphic instance pull-down menu.</p> <ul style="list-style-type: none"> • E{w}—Specifies the expected value or mean of the process noise vector. The default is 0. • Q—Specifies the covariance matrix of the process noise vector. The default is 0.01. • E{v}—Specifies the expected value or mean of the measurement noise vector. The default is 0. • R—Specifies the covariance matrix of the measurement noise vector. The default is 0.1. • N—Specifies the cross-covariance matrix between the process noise vector and the measurement noise vector. If these noise vectors are uncorrelated, N is a matrix of zeros. The default is 0.
sample skew (s)	<p>Specifies the length of time by which you want to delay the execution of this function. The default value is 0. The value of this parameter must satisfy the following relationship:</p> $0 \leq \text{sample skew (s)} \leq \text{Sampling Time}$
Initial State $x(0)$	Specifies the initial state of the model. If you do not specify a value for this parameter, this function assumes an initial state of 0.

Input $u(k)$	Specifies the control action this function applies to the model. If you specify a vector of zeros for Input $u(k)$, this function does not apply a control action.
Process Noise $w(k)$	Specifies the process noise vector. This input is available only if you select External Noise from the Polymorphic instance pull-down menu.
Measurement Noise $v(k)$	Specifies the measurement noise vector. This input is available only if you select External Noise from the Polymorphic instance pull-down menu.

Block Diagram Outputs

Parameter	Description
Output $y(k)$	Returns the values of the model output(s) at time k . The length of this vector is equal to the number of model outputs.
State $x(k+1)$	Returns the values of the model state(s) at time $k + 1$. The length of this vector is equal to the number of model states.
State $x(k)$	Returns the values of the model state(s) at time k . The length of this vector is equal to the number of model states.

Discrete Stochastic State-Space Details

Internal Noise

This function adapts to changes in the **Stochastic State-Space Model** and the **Second-Order Statistics Noise Model**, as long as the model dimensions do not change. Therefore, you can use this function to simulate linear time-variant (LTV) stochastic state-space models.

If you set the **Parameter source** to **Terminal** for the **Second-Order Statistics Noise Model** input, this function uses the **Second-Order Statistics Noise Model** to obtain the values of $E\{\mathbf{w}(k)\}$, $E\{\mathbf{v}(k)\}$, $\mathbf{Q}(k)$, $\mathbf{R}(k)$, and $\mathbf{N}(k)$. The following equations define these terms:

$$Q(k)\delta_{kl} = E\{\mathbf{w}(k) \cdot \mathbf{w}^T(l)\} - E\{\mathbf{w}(k)\} \cdot E^T\{\mathbf{w}(l)\}$$

$$R(k)\delta_{kl} = E\{\mathbf{v}(k) \cdot \mathbf{v}^T(l)\} - E\{\mathbf{v}(k)\} \cdot E^T\{\mathbf{v}(l)\}$$

$$N(k)\delta_{kl} = E\{\mathbf{w}(k) \cdot \mathbf{v}^T(l)\} - E\{\mathbf{w}(k)\} \cdot E^T\{\mathbf{v}(l)\}$$

where δ_{kl} is the Kronecker delta function. The following equations defines this function: $\delta_{kl} = 1$ when $k = l$; $\delta_{kl} = 0$ when $k \neq l$.

$\mathbf{w}(k)$ is the process noise vector.

$\mathbf{v}(k)$ is the measurement noise vector.

$\mathbf{Q}(k)$ is the covariance matrix of $\mathbf{w}(k)$.

$\mathbf{R}(k)$ is the covariance matrix of $\mathbf{v}(k)$.

$\mathbf{N}(k)$ is the cross-covariance matrix of $\mathbf{w}(k)$ and $\mathbf{v}(k)$. If these noise vectors are uncorrelated, $\mathbf{N}(k)$ is a matrix of zeros.

$E\{\}$ denotes the expected mean of the enclosed term(s).

This function also uses the **Second-Order Statistics Noise Model** to generate Gaussian-distributed temporally-uncorrelated random samples of $\mathbf{w}(k)$ and $\mathbf{v}(k)$ at each time step.

External Noise

This function adapts to changes in the **Stochastic State-Space Model**, as long as the model dimensions do not change. Therefore, you can use this function to simulate linear time-variant (LTV) stochastic state-space models.

Use the CD Correlated Gaussian Random Noise VI to generate external random samples of Gaussian-distributed noise vectors. You can use this VI and the **Process Noise $w(k)$** and **Measurement Noise $v(k)$** inputs of the Discrete Stochastic State-Space function to simulate the following conditions:

- Deterministic systems excited by deterministic disturbance vectors $w(k)$ and $v(k)$.
- Stochastic systems excited by stochastic noise vectors $w(k)$ and $v(k)$. These stochastic noise vectors can be zero-mean or nonzero-mean, stationary or nonstationary, and can have any density distribution.
- Deterministic or stochastic systems excited by $w(k)$ and $v(k)$, where one of these vectors is a deterministic disturbance and the other is a stochastic noise.

Feedthrough Behavior

The value you specify for the **D** matrix of the **Stochastic State-Space Model** parameter determines the feedthrough behavior of this function.

- If the **D** matrix is nonzero, all input/output pairs have direct feedthrough behavior.
- If the **D** matrix is zero, the following input/output pairs have indirect feedthrough behavior.
 - **Input u(k) — Output y(k)**
 - **Input u(k) — State x(k)**

The remaining input/output pairs have direct feedthrough behavior.

When you use the configuration dialog box to configure the **D** matrix, LabVIEW verifies that the feedthrough behavior is correct. For example, if you set the **Feedthrough** parameter to **Indirect**, and you set the **D** matrix to nonzero, LabVIEW changes the **Feedthrough** parameter to **Direct**.

If you specify the value of the **D** matrix programmatically by wiring a value to the parameter terminal, LabVIEW does not adjust the feedthrough behavior for you. You must ensure that you specify the proper feedthrough behavior for the value of the **D** matrix that you specify.

Discrete Transfer Function Function

Owning Palette: [Discrete Linear Systems Functions](#)

Installed With: Control Design and Simulation Module

Implements a system model in discrete transfer function form. You define the system model by specifying the **Numerator** and **Denominator** of the transfer function equation.

[Details](#) [Example](#)

[Dialog Box Options](#)

[Block Diagram Inputs](#)

[Block Diagram Outputs](#)

■ Place on the block diagram ■ Find on the **Functions** palette

Dialog Box Options

Parameter	Description
Polymorphic instance	Specifies whether this function is single-input single-output (SISO) or multiple-input multiple-output (MIMO). The default value is SISO .
Feedthrough	Configures the function to be either a Direct or Indirect feedthrough function. Enable this control by selecting a parameter from the Parameters list and then selecting Terminal from the Parameter source pull-down menu. If you select Configuration Dialog Box from the Parameter source pull-down menu, LabVIEW disables this control and calculates the feedthrough behavior automatically. Refer to the Details section for a description of the parameters that determine the feedthrough behavior of this function.
Parameters	Lists all the parameters associated with this function. Select a parameter from this list to configure the parameter. When you select a parameter, the parameter and its associated Parameter source control appear in the Parameter Information section of the configuration dialog box.
Preview	Displays a graphical preview, if available, of the function output or configuration.
Parameter Information	Contains the parameters you can configure for this function. You must select a parameter from the Parameters list to make that parameter and its associated Parameter source control visible in the Parameter Information section of the configuration dialog box.
Parameter source	Specifies whether you configure this parameter using the Configuration Dialog Box or a Terminal on the simulation diagram. The default value is Configuration Dialog Box . If you select Terminal , LabVIEW displays an input for that parameter on the simulation diagram, and

	<p>you can wire values to that input to configure this function programmatically. If you select Configuration Dialog Box, LabVIEW removes that input from the simulation diagram. You then must set the value for this parameter inside the configuration dialog box.</p>
Transfer Function	<p>Specifies the transfer function in terms of numerator and denominator polynomial functions.</p> <ul style="list-style-type: none"> • Load Model—Loads model information from a data file. • Save Model—Saves model information to a data file. This file is compatible with the Control Design VIs and functions. • Copy to Clipboard—Copies the current model definition to the clipboard. From the clipboard, you can paste the model on the block diagram or into another configuration dialog box of the same model form. • Paste from Clipboard—Pastes model information from the clipboard to the configuration dialog box. • Model Dimensions—Use this section to specify the number of inputs and outputs of the system model. You also use this section to specify the input-output location of the equation you want to edit. This section is available only if you select MIMO from the Polymorphic instance pull-down menu. <ul style="list-style-type: none"> - Inputs—Specifies the number of model inputs. - Outputs—Specifies the number of model outputs. • Current Input—Specifies the current column of the Input-Output Model. • Current Output—Specifies the current row of the Input-Output Model. • Input-Output Model—Displays a graphical representation of the MIMO model. Click a cell to

	<p>edit the equation at that input-output location. You also can use the Current Input and Current Output controls to specify the location of the equation you want to edit.</p> <ul style="list-style-type: none"> • Numerator—Specifies the coefficients of the numerator polynomial function in ascending powers of z, where z is the z-transform variable. For MIMO models, Numerator applies to the equation that the Current Input and Current Output parameters specify. • Denominator—Specifies the coefficients of the denominator polynomial function in ascending powers of z, where z is the z-transform variable. For MIMO models, Denominator applies to the equation that the Current Input and Current Output parameters specify. For each equation, the order of the Denominator must be greater than or equal to the order of the Numerator.
sample period (s)	<p>Specifies the length of the discrete time step, in seconds, of this function. The default value is -1.</p> <p>If you enter a value of -1, this function inherits the discrete time step you specify for the simulation diagram. Otherwise, the value of sample period (s) must be a multiple of the discrete time step you specify for the simulation diagram.</p> <p>This parameter is valid only when you place this function inside a Simulation Loop.</p>
sample skew (s)	<p>Specifies the length of time by which you want to delay the execution of this function. The default value is 0.</p> <p>The value of this parameter must satisfy the following relationship:</p> $0 \leq \text{sample skew (s)} \leq \text{sample period (s)}$

	This parameter is valid only when you place this function inside a Simulation Loop.
reset?	Sets the model state(s) to 0, when TRUE.

Block Diagram Inputs

Parameter	Description
sample period (s)	<p>Specifies the length of the discrete time step, in seconds, of this function. The default value is –1.</p> <p>If you enter a value of –1, this function inherits the discrete time step you specify for the simulation diagram. Otherwise, the value of sample period (s) must be a multiple of the discrete time step you specify for the simulation diagram.</p> <p>This parameter is valid only when you place this function inside a Simulation Loop.</p>
sample skew (s)	<p>Specifies the length of time by which you want to delay the execution of this function. The default value is 0.</p> <p>The value of this parameter must satisfy the following relationship:</p> $0 \leq \text{sample skew (s)} \leq \text{sample period (s)}$ <p>This parameter is valid only when you place this function inside a Simulation Loop.</p>
reset?	Sets the model state(s) to 0, when TRUE.
input u(k)	Specifies the input to the system. input u(k) must be a vector of length m , where m is the number of inputs.
Transfer Function	Specifies a transfer function model. This input accepts either a block diagram constant or a model you created using Control Design VIs and functions.

Block Diagram Outputs

Parameter	Description
output $y(k)$	Returns the current output of the system. This vector must be of length r , where r is the number of outputs.

Discrete Transfer Function Details

For SISO models, this function uses the following equation to calculate the output:

$$H(z) = \frac{b_0 + b_1 z + \dots + b_{m-1} z^{m-1} + b_m z^m}{a_0 + a_1 z + \dots + a_{n-1} z^{n-1} + a_n z^n}$$

For MIMO models, this function calculates the output as $H = [H_{ij}]$.

where $b_0 \dots b_m$ are the coefficients of the numerator polynomial

$a_0 \dots a_n$ are the coefficients of the denominator polynomial

m is the order of the numerator

n is the order of the denominator

z is the z-transform variable

i is the index number of the input

j is the index number of the output

Feedthrough Behavior

The values you specify for the **Numerator** and **Denominator** subparameters determine the feedthrough behavior of this function. Given m as the order of the **Numerator** and n as the order of the **Denominator**:

- If $n = m$, all input/output pairs have direct feedthrough behavior.
- If $n > m$, the **input u(k)** input has indirect feedthrough to the **output y(k)** output.
- If $n < m$, LabVIEW returns an error.

When you use the configuration dialog box to configure n and m , LabVIEW verifies that the feedthrough behavior is correct. For example, if you set the **Execution Mode** parameter to **Indirect**, and you set n equal to m , LabVIEW changes the **Execution Mode** parameter to **Direct**.

If you define the transfer function programmatically, LabVIEW does not adjust the feedthrough behavior for you. You must ensure that you specify the proper feedthrough behavior for the orders of n and m you specify.

Example

Refer to the SimEx discrete transfer function VI in the labview\examples\Control and Simulation\Simulation\Discrete Linear directory for an example of using the Discrete Transfer Function VI.

- Open example ■ Browse related examples

Discrete Unit Delay Function

Owning Palette: [Discrete Linear Systems Functions](#)

Installed With: Control Design and Simulation Module

Delays the input by the value that you specify for the **sample period (s)** of this function.

[Details](#) [Example](#)

[Dialog Box Options](#)

[Block Diagram Inputs](#)

[Block Diagram Outputs](#)

■ Place on the block diagram ■ Find on the **Functions** palette

Dialog Box Options

Parameter	Description
Polymorphic instance	Specifies whether this function is Scalar or Vector . The default value is Scalar .
Parameters	Lists all the parameters associated with this function. Select a parameter from this list to configure the parameter. When you select a parameter, the parameter and its associated Parameter source control appear in the Parameter Information section of the configuration dialog box.
Preview	Displays a graphical preview, if available, of the function output or configuration.
Parameter Information	Contains the parameters you can configure for this function. You must select a parameter from the Parameters list to make that parameter and its associated Parameter source control visible in the Parameter Information section of the configuration dialog box.
Parameter source	Specifies whether you configure this parameter using the Configuration Dialog Box or a Terminal on the simulation diagram. The default value is Configuration Dialog Box . If you select Terminal , LabVIEW displays an input for that parameter on the simulation diagram, and you can wire values to that input to configure this function programmatically. If you select Configuration Dialog Box , LabVIEW removes that input from the simulation diagram. You then must set the value for this parameter inside the configuration dialog box.
sample period (s)	Specifies the length of the discrete time step, in seconds, of this function. If you enter a value of –1, this function inherits the discrete time step you specify for the simulation diagram. Otherwise, the value of sample period (s) must be a multiple of the discrete time step you specify for the simulation diagram. The default value of sample period (s) is 1.

sample skew (s)	Specifies the length of time by which you want to delay the execution of this function. The default value is 0. The value of this parameter must satisfy the following relationship: $0 \leq \text{sample skew (s)} \leq \text{sample period (s)}$
initial condition	Specifies the output value at the first time step. The default value is 0.

Block Diagram Inputs

Parameter	Description
sample period (s)	Specifies the length of the discrete time step, in seconds, of this function. If you enter a value of –1, this function inherits the discrete time step you specify for the simulation diagram. Otherwise, the value of sample period (s) must be a multiple of the discrete time step you specify for the simulation diagram. The default value of sample period (s) is 1.
sample skew (s)	Specifies the length of time by which you want to delay the execution of this function. The default value is 0. The value of this parameter must satisfy the following relationship: $0 \leq \text{sample skew (s)} \leq \text{sample period (s)}$
initial condition	Specifies the output value at the first time step. The default value is 0.
input	Specifies the input to the function.

Block Diagram Outputs

Parameter	Description
output	Returns the output of the function.

Discrete Unit Delay Details

The following equations define the output of this function.

$$y(0) = \text{initial condition}$$

$$y(t_k) = u(t_{k-1})$$

where u is the input

y is the output

$$t_k = \text{initial time} + \text{sample skew (s)} + k * \text{sample period (s)}, \text{ for } k = 0, 1, 2, \dots$$

Feedthrough Behavior

For this function, the **input** input has [indirect feedthrough](#) to the **output** output. All other input/output pairs have direct feedthrough behavior.

Example

Refer to the SimEx discrete unit delay VI in the labview\examples\Control and Simulation\Simulation\Discrete Linear directory for an example of using the Discrete Unit Delay VI.

- Open example ■ Browse related examples

Discrete Zero-Order Hold Function

Owning Palette: [Discrete Linear Systems Functions](#)

Installed With: Control Design and Simulation Module

Holds the input signal for a definite period of time equal to the value you specify for the **sample period (s)** parameter of this function. Use this function to discretize or resample an input signal.

[Details](#) [Example](#)

[Dialog Box Options](#)

[Block Diagram Inputs](#)

[Block Diagram Outputs](#)

■ Place on the block diagram ■ Find on the **Functions** palette

Dialog Box Options

Parameter	Description
Polymorphic instance	Specifies whether this function is Scalar or Vector . The default value is Scalar .
Parameters	Lists all the parameters associated with this function. Select a parameter from this list to configure the parameter. When you select a parameter, the parameter and its associated Parameter source control appear in the Parameter Information section of the configuration dialog box.
Preview	Displays a graphical preview, if available, of the function output or configuration.
Parameter Information	Contains the parameters you can configure for this function. You must select a parameter from the Parameters list to make that parameter and its associated Parameter source control visible in the Parameter Information section of the configuration dialog box.
Parameter source	Specifies whether you configure this parameter using the Configuration Dialog Box or a Terminal on the simulation diagram. The default value is Configuration Dialog Box . If you select Terminal , LabVIEW displays an input for that parameter on the simulation diagram, and you can wire values to that input to configure this function programmatically. If you select Configuration Dialog Box , LabVIEW removes that input from the simulation diagram. You then must set the value for this parameter inside the configuration dialog box.
sample period (s)	Specifies the length of the discrete time step, in seconds, of this function. If you enter a value of –1, this function inherits the discrete time step you specify for the simulation diagram. Otherwise, the value of sample period (s) must be a multiple of the discrete time step you specify for the simulation diagram. The default value of sample period (s) is 1.

sample skew (s)	Specifies the length of time by which you want to delay the execution of this function. The default value is 0. The value of this parameter must satisfy the following relationship: $0 \leq \text{sample skew (s)} \leq \text{sample period (s)}$
------------------------	---

Block Diagram Inputs

Parameter	Description
sample period (s)	Specifies the length of the discrete time step, in seconds, of this function. If you enter a value of –1, this function inherits the discrete time step you specify for the simulation diagram. Otherwise, the value of sample period (s) must be a multiple of the discrete time step you specify for the simulation diagram. The default value of sample period (s) is 1.
sample skew (s)	Specifies the length of time by which you want to delay the execution of this function. The default value is 0. The value of this parameter must satisfy the following relationship: $0 \leq \text{sample skew (s)} \leq \text{sample period (s)}$
input	Specifies the input to the function.

Block Diagram Outputs

Parameter	Description
output	Returns the output of the function.

Discrete Zero-Order Hold Details

The following equation defines the output of this function.

$$y(t) = \begin{cases} u(t_i), & t_i \leq t < t_i + \text{sample skew (s)} \\ u(t_k), & t_k \leq t < t_{k+1} \end{cases}$$

where y is the output

u is the input

t is the current simulation time

t_i is the initial simulation time

$t_k = t_i + \text{sample skew (s)} + k * \text{sample period (s)}$, for $k = 0, 1, 2,$

...

Feedthrough Behavior

All input/output pairs of this function have [direct feedthrough behavior](#).

Example

Refer to the SimEx zero-order-hold VI in the labview\examples\Control and Simulation\Simulation\Discrete Linear directory for an example of using the Discrete Zero-Order Hold VI.

- Open example ■ Browse related examples

Discrete Zero-Pole-Gain Function

Owning Palette: [Discrete Linear Systems Functions](#)

Installed With: Control Design and Simulation Module

Implements a system model in discrete zero-pole-gain form. You define the system model by specifying the **Zeros**, **Poles**, and **Gain** of the zero-pole-gain equation.

[Details](#) [Example](#)

[Dialog Box Options](#)

[Block Diagram Inputs](#)

[Block Diagram Outputs](#)

■ Place on the block diagram ■ Find on the **Functions** palette

Dialog Box Options

Parameter	Description
Polymorphic instance	Specifies whether this function is single-input single-output (SISO) or multiple-input multiple-output (MIMO). The default value is SISO .
Feedthrough	Configures the function to be either a Direct or Indirect feedthrough function. Enable this control by selecting a parameter from the Parameters list and then selecting Terminal from the Parameter source pull-down menu. If you select Configuration Dialog Box from the Parameter source pull-down menu, LabVIEW disables this control and calculates the feedthrough behavior automatically. Refer to the Details section for a description of the parameters that determine the feedthrough behavior of this function.
Parameters	Lists all the parameters associated with this function. Select a parameter from this list to configure the parameter. When you select a parameter, the parameter and its associated Parameter source control appear in the Parameter Information section of the configuration dialog box.
Preview	Displays a graphical preview, if available, of the function output or configuration.
Parameter Information	Contains the parameters you can configure for this function. You must select a parameter from the Parameters list to make that parameter and its associated Parameter source control visible in the Parameter Information section of the configuration dialog box.
Parameter source	Specifies whether you configure this parameter using the Configuration Dialog Box or a Terminal on the simulation diagram. The default value is Configuration Dialog Box . If you select Terminal , LabVIEW displays an input for that parameter on the simulation diagram, and

	<p>you can wire values to that input to configure this function programmatically. If you select Configuration Dialog Box, LabVIEW removes that input from the simulation diagram. You then must set the value for this parameter inside the configuration dialog box.</p>
Zeros-Poles-Gain	<p>Specifies the zero-pole-gain model.</p> <ul style="list-style-type: none"> • Load Model—Loads model information from a data file. • Save Model—Saves model information to a data file. This file is compatible with the Control Design VIs and functions. • Copy to Clipboard—Copies the current model definition to the clipboard. From the clipboard, you can paste the model on the block diagram or into another configuration dialog box of the same model form. • Paste from Clipboard—Pastes model information from the clipboard to the configuration dialog box. • Model Dimensions—Use this section to specify the number of inputs and outputs of the system model. You also use this section to specify the input-output location of the equation you want to edit. This section is available only if you select MIMO from the Polymorphic instance pull-down menu. <ul style="list-style-type: none"> - Inputs—Specifies the number of model inputs. - Outputs—Specifies the number of model outputs. • Current Input—Specifies the current column of the Input-Output Model. • Current Output—Specifies the current row of the Input-Output Model. • Input-Output Model—Displays a graphical representation of the MIMO model. Click a cell to edit the equation at that input-output location. You

	<p>also can use the Current Input and Current Output controls to specify the location of the equation you want to edit.</p> <ul style="list-style-type: none"> • Gain—Specifies a real value representing the common gain of the zero-pole-gain model. For MIMO models, Gain applies to the equation that the Current Input and Current Output parameters specify. • Zeros—Specifies zeros of the zero-pole-gain model. For all zeros with an imaginary part, the conjugate complex number also must belong to this array. For MIMO models, Zeros applies to the equation that the Current Input and Current Output parameters specify. • Poles—Specifies the poles of the zero-pole-gain model. For all poles with an imaginary part, the conjugate complex number also must belong to this array. For MIMO models, Poles applies to the equation that the Current Input and Current Output parameters specify.
sample period (s)	<p>Specifies the length of the discrete time step, in seconds, of this function. The default value is –1.</p> <p>If you enter a value of –1, this function inherits the discrete time step you specify for the simulation diagram. Otherwise, the value of sample period (s) must be a multiple of the discrete time step you specify for the simulation diagram.</p> <p>This parameter is valid only when you place this function inside a Simulation Loop.</p>
sample skew (s)	<p>Specifies the length of time by which you want to delay the execution of this function. The default value is 0.</p> <p>The value of this parameter must satisfy the following relationship:</p>

	<p>$0 \leq \text{sample skew (s)} \leq \text{sample period (s)}$</p> <p>This parameter is valid only when you place this function inside a Simulation Loop.</p>
reset?	Sets the model state(s) to 0, when TRUE.

Block Diagram Inputs

Parameter	Description
sample period (s)	<p>Specifies the length of the discrete time step, in seconds, of this function. The default value is –1.</p> <p>If you enter a value of –1, this function inherits the discrete time step you specify for the simulation diagram. Otherwise, the value of sample period (s) must be a multiple of the discrete time step you specify for the simulation diagram.</p> <p>This parameter is valid only when you place this function inside a Simulation Loop.</p>
sample skew (s)	<p>Specifies the length of time by which you want to delay the execution of this function. The default value is 0.</p> <p>The value of this parameter must satisfy the following relationship:</p> $0 \leq \text{sample skew (s)} \leq \text{sample period (s)}$ <p>This parameter is valid only when you place this function inside a Simulation Loop.</p>
reset?	Sets the model state(s) to 0, when TRUE.
input u(k)	Specifies the input to the system. input u(k) must be a vector of length m , where m is the number of inputs.
Zeros-Poles-Gain	Specifies a zero-pole-gain model. This input accepts either a block diagram constant or a model you created using the Control Design VIs and functions.

Block Diagram Outputs

Parameter	Description
output $y(k)$	Returns the current output of the system. This vector must be of length r , where r is the number of outputs.

Discrete Zero-Pole-Gain Details

For SISO models, this function uses the following equation to calculate the output:

$$H(z) = \frac{k(z - Z[1])(z - Z[2])\dots(z - Z[m])}{(z - P[1])(z - P[2])\dots(z - P[n])}$$

For MIMO models, this function calculates the output as $H = [H_{ij}]$.

where k is the gain

$Z[m]$ is the array of zeros

$P[n]$ is the array of poles

m is the order of the numerator

n is the order of the denominator

z is the z-transform variable

i is the index number of the input

j is the index number of the output

The function is based on the [Discrete Transfer Function](#), which is represented in zero-pole-gain notation.

Feedthrough Behavior

The values you specify for the **Zeros** and **Poles** subparameters determine the feedthrough behavior of this function. Given Z as the **Zeros** subparameter and P as the **Poles** subparameter:

- If the order of Z = the order of P , the function has direct feedthrough behavior.
- If the order of Z < the order of P , the **input u(k)** input has indirect feedthrough to the **output y(k)** output. All other input/output pairs have direct feedthrough behavior.
- If the order of Z > the order of P , LabVIEW returns an error.

When you use the configuration dialog box to configure Z and P , LabVIEW verifies that the feedthrough behavior is correct. For example, if you set the **Execution Mode** parameter to **Indirect**, and you set the order of Z equal to the order of P , LabVIEW changes the **Execution Mode** parameter to **Direct**.

If you define the zero-pole-gain equation programmatically, LabVIEW does not adjust the feedthrough behavior for you. You must ensure that you specify the proper feedthrough behavior for the orders of Z and P you specify.

Example

Refer to the SimEx discrete zero-pole-gain VI in the labview\examples\Control and Simulation\Simulation\Discrete Linear directory for an example of using the Discrete Zero-Pole-Gain VI.

- Open example ■ Browse related examples

Graph Utilities Functions

Owning Palette: [Simulation VIs and Functions](#)

Installed With: Control Design and Simulation Module. This topic might not match its corresponding palette in LabVIEW depending on your operating system, licensed product(s), and target.

Use the Graph Utilities functions to display the outputs of the simulation on a chart or graph.

The functions on this palette can return [general LabVIEW error codes](#) or [specific Simulation error codes](#). If you use the functions on this palette in a [Simulation Loop](#), LabVIEW sends any errors that these functions return to the **Error** output on the Output Node of the Simulation Loop.

Palette Object	Description
Buffer XY Graph	Collects and plots x, y value pairs on an XY graph .
SimTime Waveform	Plots a value versus the simulation time on a waveform chart .

Buffer XY Graph Function

Owning Palette: [Graph Utilities Functions](#)

Installed With: Control Design and Simulation Module

Collects and plots x, y value pairs on an [XY graph](#).

When you place this function on a simulation diagram, LabVIEW automatically wires an XY graph indicator to the function.

[Details](#)

[Dialog Box Options](#)

[Block Diagram Inputs](#)

[Block Diagram Outputs](#)

■ Place on the block diagram ■ Find on the **Functions** palette

Dialog Box Options

Parameter	Description
Polymorphic instance	Specifies whether this function is Scalar or Vector . The default value is Scalar .
Parameters	Lists all the parameters associated with this function. Select a parameter from this list to configure the parameter. When you select a parameter, the parameter and its associated Parameter source control appear in the Parameter Information section of the configuration dialog box.
Preview	Displays a graphical preview, if available, of the function output or configuration.
Parameter Information	Contains the parameters you can configure for this function. You must select a parameter from the Parameters list to make that parameter and its associated Parameter source control visible in the Parameter Information section of the configuration dialog box.
Parameter source	Specifies whether you configure this parameter using the Configuration Dialog Box or a Terminal on the simulation diagram. The default value is Configuration Dialog Box . If you select Terminal , LabVIEW displays an input for that parameter on the simulation diagram, and you can wire values to that input to configure this function programmatically. If you select Configuration Dialog Box , LabVIEW removes that input from the simulation diagram. You then must set the value for this parameter inside the configuration dialog box.
decimation	Specifies the value the function uses to determine when to plot a value. If you set the value of decimation to 1, the function plots a value every time step. If you set the value of decimation to n , the function plots a value once every n time steps. The default value is 1.
Buffer Size	Specifies the number of values to buffer. You can set the value of this parameter only on the initial iteration of

the simulation. You cannot change the value of this parameter while the simulation is running. The default value is 1024.

Block Diagram Inputs

Parameter	Description
decimation	Specifies the value the function uses to determine when to plot a value. If you set the value of decimation to 1, the function plots a value every time step. If you set the value of decimation to n , the function plots a value once every n time steps. The default value is 1.
Buffer Size	Specifies the number of values to buffer. You can set the value of this parameter only on the initial iteration of the simulation. You cannot change the value of this parameter while the simulation is running. The default value is 1024.
Value	Specifies the x, y pair to plot for the current time step.

Block Diagram Outputs

Parameter	Description
XY Graph	Draws an XY graph that contains a history of points for the previous Buffer Size time steps.

Buffer XY Graph Details

Use this function instead of a LabVIEW XY graph to plot simulation data correctly. The Buffer XY Graph updates only on major time steps of the simulation.

SimTime Waveform Function

Owning Palette: [Graph Utilities Functions](#)

Installed With: Control Design and Simulation Module

Plots a value versus the simulation time on a [waveform chart](#).

When you place this function on a simulation diagram, LabVIEW automatically wires a waveform chart indicator to the function.

[Details](#)

[Dialog Box Options](#)

[Block Diagram Inputs](#)

[Block Diagram Outputs](#)

■ Place on the block diagram ■ Find on the **Functions** palette

Dialog Box Options

Parameter	Description
Polymorphic instance	Specifies whether this function is Scalar or Vector . The default value is Scalar .
Preview	Displays a graphical preview, if available, of the function output or configuration.

Block Diagram Inputs

Parameter	Description
Value	Specifies the value to add to the plot for the current time step.

Block Diagram Outputs

Parameter	Description
Waveform Chart	Draws a waveform consisting of a time-value pair on the waveform chart during the current time step.

SimTime Waveform Details

Use this function in place of a LabVIEW waveform chart to plot simulation data correctly. The SimTime Waveform updates only on major time steps of the simulation.

Lookup Tables Functions

Owning Palette: [Simulation VIs and Functions](#)

Installed With: Control Design and Simulation Module. This topic might not match its corresponding palette in LabVIEW depending on your operating system, licensed product(s), and target.

Use the Lookup Tables functions to locate a value from an existing data set. You specify this data set in the **LUT Data** parameter of each Lookup Tables function.

The functions on this palette can return [general LabVIEW error codes](#) or [specific Simulation error codes](#). If you use the functions on this palette in a [Simulation Loop](#), LabVIEW sends any errors that these functions return to the **Error** output on the Output Node of the Simulation Loop.

Palette Object	Description
Lookup Table 1D	Returns the y value, located in a one-dimensional table, that corresponds to an x value you specify.
Lookup Table 2D	Returns the z value, located in a two-dimensional table, that corresponds to x and y values you specify.
Lookup Table 3D	Returns the w value, located in a three-dimensional table, that corresponds to x, y, and z values you specify.

Lookup Table 1D Function

Owning Palette: [Lookup Tables Functions](#)

Installed With: Control Design and Simulation Module

Returns the y value, located in a one-dimensional table, that corresponds to an x value you specify.

[Details](#) [Example](#)

[Dialog Box Options](#)

[Block Diagram Inputs](#)

[Block Diagram Outputs](#)

■ Place on the block diagram ■ Find on the **Functions** palette

Dialog Box Options

Parameter	Description
Polymorphic instance	Specifies whether this function is Scalar or Vector . The default value is Scalar .
Parameters	Lists all the parameters associated with this function. Select a parameter from this list to configure the parameter. When you select a parameter, the parameter and its associated Parameter source control appear in the Parameter Information section of the configuration dialog box.
Preview	Displays a graphical preview, if available, of the function output or configuration.
Parameter Information	Contains the parameters you can configure for this function. You must select a parameter from the Parameters list to make that parameter and its associated Parameter source control visible in the Parameter Information section of the configuration dialog box.
Parameter source	Specifies whether you configure this parameter using the Configuration Dialog Box or a Terminal on the simulation diagram. The default value is Configuration Dialog Box . If you select Terminal , LabVIEW displays an input for that parameter on the simulation diagram, and you can wire values to that input to configure this function programmatically. If you select Configuration Dialog Box , LabVIEW removes that input from the simulation diagram. You then must set the value for this parameter inside the configuration dialog box.
Method	Specifies the interpolation method the lookup table uses. You can choose from the following options: <ul style="list-style-type: none">- Interpolate/Extrapolate—(Default) If the given X Val lies between two points in the LUT X set, the result is a linear interpolation of the LUT Data values that correspond to the two LUT X points. If the given X Val lies outside the range of the LUT X set, the result is a linear extrapolation of the

LUT Data values that correspond to the nearest two **LUT X** points.

For an interpolation example, given **LUT X** is [0 1 5], **LUT Data** is [4 2 8], and **X Val** is 4, the two **LUT X** values that are nearest to the given **X Val** are 1 and 5. These points represent the *x* coordinates of a set of points on a line. The **LUT Data** values that correspond to those **LUT X** values are 2 and 8, which represent the corresponding *y* coordinates of the points on the line. This function calculates the slope of the line as $(8 - 2) / (5 - 1) = 1.5$. This function then uses the slope to interpolate linearly the *y* value when *x* is 4. Therefore, the **Result** is 6.5.

For an extrapolation example, given **LUT X** is [0 1 5], **LUT Data** is [4 2 8], and **X Val** is 6, the two **LUT X** values that are nearest to the given **X Val** are 1 and 5. These points represent the *x* coordinates of a set of points on a line. The **LUT Data** values that correspond to those **LUT X** values are 2 and 8, which represent the corresponding *y* coordinates of the points on the line. This function calculates the slope of the line as $(8 - 2) / (5 - 1) = 1.5$. This function then uses the slope to extrapolate linearly the *y* value when *x* is 6. Therefore, the **Result** is 9.5.

- **Interpolate**—If the given **X Val** lies between two points in the **LUT X** set, the result is a linear interpolation of the **LUT Data** values that correspond to the two **LUT X** points. If the given **X Val** is out of range, the result is the **LUT Data** value that corresponds to the value of **LUT X** closest to **X Val**.
- **Nearest**—If the given **X Val** lies between two points in the **LUT X** set, the result is the **LUT Data** value that corresponds to the **LUT X** point whose

value is nearest to the given **X Val**.

For example, given **LUT X** is [0 1 5], **LUT Data** is [4 2 8], and **X Val** is 4, the **LUT X** value that is nearest to the given **X Val** is 5. The **LUT Data** value that corresponds to an **LUT X** value of 5 is 8. Therefore, the **Result** is 8.

- **Below**—If the given **X Val** lies between two points in the **LUT X** set, the result is the **LUT Data** value that corresponds to the **LUT X** point whose value is less than the given **X Val**.

For example, given **LUT X** is [0 1 5], **LUT Data** is [4 2 8], and **X Val** is 4, the **LUT X** value that lies directly below the given **X Val** is 1. The **LUT Data** value that corresponds to an **LUT X** value of 1 is 2. Therefore, the **Result** is 2.

- **Above**—If the given **X Val** lies between two points in the **LUT X** set, the result is the **LUT Data** value that corresponds to the **LUT X** point whose value is greater than the given **X Val**.

For example, given **LUT X** is [0 1 5], **LUT Data** is [4 2 8], and **X Val** is 4, the **LUT X** value that lies directly above the given **X Val** is 5. The **LUT Data** value that corresponds to an **LUT X** value of 5 is 8. Therefore, the **Result** is 8.

If you do not select **Interpolate/Extrapolate** for **Method** and the given value is out of range, this function returns the value of the end point that is closest to the given value.

LUT X	Specifies the set of <i>x</i> values that correspond to the <i>y</i> data points in LUT Data . The elements of LUT X must be monotonically increasing or equal such that LUT X[i] ≥ LUT X[j] if <i>i > j</i> .
LUT Data	Specifies a one-dimensional table of <i>y</i> data points that correspond to the values you specify for the LUT X parameter.

Block Diagram Inputs

Parameter	Description
Method	<p>Specifies the interpolation method the lookup table uses. You can choose from the following options:</p> <ul style="list-style-type: none">- Interpolate/Extrapolate—(Default) If the given X Val lies between two points in the LUT X set, the result is a linear interpolation of the LUT Data values that correspond to the two LUT X points. If the given X Val lies outside the range of the LUT X set, the result is a linear extrapolation of the LUT Data values that correspond to the nearest two LUT X points. <p>For an interpolation example, given LUT X is [0 1 5], LUT Data is [4 2 8], and X Val is 4, the two LUT X values that are nearest to the given X Val are 1 and 5. These points represent the x coordinates of a set of points on a line. The LUT Data values that correspond to those LUT X values are 2 and 8, which represent the corresponding y coordinates of the points on the line. This function calculates the slope of the line as $(8 - 2) / (5 - 1) = 1.5$. This function then uses the slope to interpolate linearly the y value when x is 4. Therefore, the Result is 6.5.</p> <p>For an extrapolation example, given LUT X is [0 1 5], LUT Data is [4 2 8], and X Val is 6, the two LUT X values that are nearest to the given X Val are 1 and 5. These points represent the x coordinates of a set of points on a line. The LUT Data values that correspond to those LUT X values are 2 and 8, which represent the corresponding y coordinates of the points on the line. This function calculates the slope of the line as $(8 - 2) / (5 - 1) = 1.5$. This function then uses the slope to extrapolate linearly the y value when x is 6. Therefore, the Result is 9.5.</p> <ul style="list-style-type: none">- Interpolate—If the given X Val lies between two

points in the **LUT X** set, the result is a linear interpolation of the **LUT Data** values that correspond to the two **LUT X** points. If the given **X Val** is out of range, the result is the **LUT Data** value that corresponds to the value of **LUT X** closest to **X Val**.

- **Nearest**—If the given **X Val** lies between two points in the **LUT X** set, the result is the **LUT Data** value that corresponds to the **LUT X** point whose value is nearest to the given **X Val**.

For example, given **LUT X** is [0 1 5], **LUT Data** is [4 2 8], and **X Val** is 4, the **LUT X** value that is nearest to the given **X Val** is 5. The **LUT Data** value that corresponds to an **LUT X** value of 5 is 8. Therefore, the **Result** is 8.

- **Below**—If the given **X Val** lies between two points in the **LUT X** set, the result is the **LUT Data** value that corresponds to the **LUT X** point whose value is less than the given **X Val**.

For example, given **LUT X** is [0 1 5], **LUT Data** is [4 2 8], and **X Val** is 4, the **LUT X** value that lies directly below the given **X Val** is 1. The **LUT Data** value that corresponds to an **LUT X** value of 1 is 2. Therefore, the **Result** is 2.

- **Above**—If the given **X Val** lies between two points in the **LUT X** set, the result is the **LUT Data** value that corresponds to the **LUT X** point whose value is greater than the given **X Val**.

For example, given **LUT X** is [0 1 5], **LUT Data** is [4 2 8], and **X Val** is 4, the **LUT X** value that lies directly above the given **X Val** is 5. The **LUT Data** value that corresponds to an **LUT X** value of 5 is 8. Therefore, the **Result** is 8.

If you do not select **Interpolate/Extrapolate** for **Method** and the given value is out of range, this function returns the value of the end point that is closest to the given value.

LUT X	Specifies the set of x values that correspond to the y data points in LUT Data . The elements of LUT X must be monotonically increasing or equal such that $\text{LUT X}[i] \geq \text{LUT X}[j]$ if $i > j$.
LUT Data	Specifies a one-dimensional table of y data points that correspond to the values you specify for the LUT X parameter.
X Val	Specifies the x value to locate in the LUT Data table.

Block Diagram Outputs

Parameter	Description
Result	Returns the <i>y</i> value in the LUT Data table that corresponds to the value you specify for the X Val parameter. If a direct match does not exist, this function operates according to the value you specify for the Method parameter.

Lookup Table 1D Details

Feedthrough Behavior

All input/output pairs of this function have [direct feedthrough behavior](#).

Example

Refer to the SimEx LUT 1D VI in the labview\examples\Control and Simulation\Simulation\LUT directory for an example of using the Lookup Table 1D VI.

- Open example ■ Browse related examples

Lookup Table 2D Function

Owning Palette: [Lookup Tables Functions](#)

Installed With: Control Design and Simulation Module

Returns the z value, located in a two-dimensional table, that corresponds to x and y values you specify.

[Details](#) [Example](#)

[Dialog Box Options](#)

[Block Diagram Inputs](#)

[Block Diagram Outputs](#)

■ Place on the block diagram ■ Find on the **Functions** palette

Dialog Box Options

Parameter	Description
Polymorphic instance	Specifies whether this function is Scalar or Vector . The default value is Scalar .
Parameters	Lists all the parameters associated with this function. Select a parameter from this list to configure the parameter. When you select a parameter, the parameter and its associated Parameter source control appear in the Parameter Information section of the configuration dialog box.
Preview	Displays a graphical preview, if available, of the function output or configuration.
Parameter Information	Contains the parameters you can configure for this function. You must select a parameter from the Parameters list to make that parameter and its associated Parameter source control visible in the Parameter Information section of the configuration dialog box.
Parameter source	Specifies whether you configure this parameter using the Configuration Dialog Box or a Terminal on the simulation diagram. The default value is Configuration Dialog Box . If you select Terminal , LabVIEW displays an input for that parameter on the simulation diagram, and you can wire values to that input to configure this function programmatically. If you select Configuration Dialog Box , LabVIEW removes that input from the simulation diagram. You then must set the value for this parameter inside the configuration dialog box.
Method	Specifies the interpolation method the lookup table uses. You can choose from the following options: <ul style="list-style-type: none">- Interpolate/Extrapolate (Default)- Interpolate- Nearest- Below- Above

	If you do not select Interpolate/Extrapolate for Method and the given value is out of range, this function returns the value of the end point that is closest to the given value.
LUT X	Specifies the set of x values that correspond to the row index of data points in the LUT Data table. The elements of LUT X must be monotonically increasing or equal such that $\text{LUT X}[i] \geq \text{LUT X}[j]$ if $i > j$.
LUT Y	Specifies the set of y values that correspond to the column index of data points in the LUT Data table. The elements of LUT Y must be monotonically increasing or equal such that $\text{LUT Y}[i] \geq \text{LUT Y}[j]$ if $i > j$.
LUT Data	Specifies a two-dimensional table of x and y data points that correspond to the values you specify for the LUT X and LUT Y parameters.

Block Diagram Inputs

Parameter	Description
Method	<p>Specifies the interpolation method the lookup table uses. You can choose from the following options:</p> <ul style="list-style-type: none"> - Interpolate/Extrapolate (Default) - Interpolate - Nearest - Below - Above <p>If you do not select Interpolate/Extrapolate for Method and the given value is out of range, this function returns the value of the end point that is closest to the given value.</p>
LUT X	Specifies the set of <i>x</i> values that correspond to the row index of data points in the LUT Data table. The elements of LUT X must be monotonically increasing or equal such that LUT X[i] ≥ LUT X[j] if <i>i > j</i> .
LUT Y	Specifies the set of <i>y</i> values that correspond to the column index of data points in the LUT Data table. The elements of LUT Y must be monotonically increasing or equal such that LUT Y[i] ≥ LUT Y[j] if <i>i > j</i> .
LUT Data	Specifies a two-dimensional table of <i>x</i> and <i>y</i> data points that correspond to the values you specify for the LUT X and LUT Y parameters.
X Val	Specifies the <i>x</i> value to locate in the LUT Data table.
Y Val	Specifies the <i>y</i> value to locate in the LUT Data table.

Block Diagram Outputs

Parameter	Description
Result	Returns the z value in the LUT Data table that corresponds to the values you specify for the X Val and Y Val parameters. If a direct match does not exist, this function operates according to the value you specify for the Method parameter.

Lookup Table 2D Details

Feedthrough Behavior

All input/output pairs of this function have [direct feedthrough behavior](#).

Example

Refer to the SimEx LUT 2D VI in the labview\examples\Control and Simulation\Simulation\LUT directory for an example of using the Lookup Table 2D VI.

- Open example ■ Browse related examples

Lookup Table 3D Function

Owning Palette: [Lookup Tables Functions](#)

Installed With: Control Design and Simulation Module

Returns the w value, located in a three-dimensional table, that corresponds to x , y , and z values you specify.

[Details](#) [Example](#)

[Dialog Box Options](#)

[Block Diagram Inputs](#)

[Block Diagram Outputs](#)

■ Place on the block diagram ■ Find on the **Functions** palette

Dialog Box Options

Parameter	Description
Polymorphic instance	Specifies whether this function is Scalar or Vector . The default value is Scalar .
Parameters	Lists all the parameters associated with this function. Select a parameter from this list to configure the parameter. When you select a parameter, the parameter and its associated Parameter source control appear in the Parameter Information section of the configuration dialog box.
Preview	Displays a graphical preview, if available, of the function output or configuration.
Parameter Information	Contains the parameters you can configure for this function. You must select a parameter from the Parameters list to make that parameter and its associated Parameter source control visible in the Parameter Information section of the configuration dialog box.
Parameter source	Specifies whether you configure this parameter using the Configuration Dialog Box or a Terminal on the simulation diagram. The default value is Configuration Dialog Box . If you select Terminal , LabVIEW displays an input for that parameter on the simulation diagram, and you can wire values to that input to configure this function programmatically. If you select Configuration Dialog Box , LabVIEW removes that input from the simulation diagram. You then must set the value for this parameter inside the configuration dialog box.
Method	Specifies the interpolation method the lookup table uses. You can choose from the following options: <ul style="list-style-type: none">- Interpolate/Extrapolate (Default)- Interpolate- Nearest- Below- Above

	If you do not select Interpolate/Extrapolate for Method and the given value is out of range, this function returns the value of the end point that is closest to the given value.
LUT X	Specifies the set of x values that correspond to the row index of data points in the LUT Data table. The elements of LUT X must be monotonically increasing or equal such that $\text{LUT X}[i] \geq \text{LUT X}[j]$ if $i > j$.
LUT Y	Specifies the set of y values that correspond to the column index of data points in the LUT Data table. The elements of LUT Y must be monotonically increasing or equal such that $\text{LUT Y}[i] \geq \text{LUT Y}[j]$ if $i > j$.
LUT Z	Specifies the set of z values that correspond to the page index of data points in the LUT Data table. The elements of LUT Z must be monotonically increasing or equal such that $\text{LUT Z}[i] \geq \text{LUT Z}[j]$ if $i > j$.
LUT Data	Specifies a three-dimensional table of x , y , and z data points that correspond to the values you specify for the LUT X , LUT Y , and LUT Z parameters.

Block Diagram Inputs

Parameter	Description
Method	<p>Specifies the interpolation method the lookup table uses. You can choose from the following options:</p> <ul style="list-style-type: none"> - Interpolate/Extrapolate (Default) - Interpolate - Nearest - Below - Above <p>If you do not select Interpolate/Extrapolate for Method and the given value is out of range, this function returns the value of the end point that is closest to the given value.</p>
LUT X	Specifies the set of <i>x</i> values that correspond to the row index of data points in the LUT Data table. The elements of LUT X must be monotonically increasing or equal such that LUT X[i] ≥ LUT X[j] if <i>i > j</i> .
LUT Y	Specifies the set of <i>y</i> values that correspond to the column index of data points in the LUT Data table. The elements of LUT Y must be monotonically increasing or equal such that LUT Y[i] ≥ LUT Y[j] if <i>i > j</i> .
LUT Z	Specifies the set of <i>z</i> values that correspond to the page index of data points in the LUT Data table. The elements of LUT Z must be monotonically increasing or equal such that LUT Z[i] ≥ LUT Z[j] if <i>i > j</i> .
LUT Data	Specifies a three-dimensional table of <i>x</i> , <i>y</i> , and <i>z</i> data points that correspond to the values you specify for the LUT X , LUT Y , and LUT Z parameters.
X Val	Specifies the <i>x</i> value to locate in the LUT Data table.
Y Val	Specifies the <i>y</i> value to locate in the LUT Data table.
Z Val	Specifies the <i>z</i> value to locate in the LUT Data table.

Block Diagram Outputs

Parameter	Description
Result	Returns the <i>w</i> value in the LUT Data table that corresponds to the values you specify for the X Val , Y Val , and Z Val parameters. If a direct match does not exist, this function operates according to the value you specify for the Method parameter.

Lookup Table 3D Details

Feedthrough Behavior

All input/output pairs of this function have [direct feedthrough behavior](#).

Example

Refer to the SimEx LUT 3D VI in the labview\examples\Control and Simulation\Simulation\LUT directory for an example of using the Lookup Table 3D VI.

- Open example ■ Browse related examples

Nonlinear Systems Functions

Owning Palette: [Simulation VIs and Functions](#)

Installed With: Control Design and Simulation Module. This topic might not match its corresponding palette in LabVIEW depending on your operating system, licensed product(s), and target.

Use the Nonlinear Systems functions to simulate nonlinear effects that are present in real-world systems.

The functions on this palette can return [general LabVIEW error codes](#) or [specific Simulation error codes](#). If you use the functions on this palette in a [Simulation Loop](#), LabVIEW sends any errors that these functions return to the **Error** output on the Output Node of the Simulation Loop.

Palette Object	Description
Backlash	Implements a backlash or deadband function. For example, these functions represent clearance between teeth in a gear train.
Dead Zone	Implements a dead zone function. For example, this function represents a forward-based diode in which you must apply a small forward voltage drop before current begins to flow. You specify the width of the dead zone using the start of dead zone and end of dead zone parameters.
Detect Zero Crossing	Detects a zero crossing according to the trigger type you specify. A zero crossing occurs when the input signal you specify crosses the x-axis.
Friction	Represents friction at zero but guarantees linear behavior for nonzero inputs. You also can use this function to represent Coulomb and viscous friction.
Quantizer	Quantizes a continuous input signal to discrete states.
Rate Limiter	Limits the rate of change of a signal.
Relay	Returns one of two values based on the value of the input signal.
Saturation	Limits the valid range of a signal.

Switch	Switches the value of the output between value above threshold and value below threshold based on the value of the control input .
---------------	--

Backlash Function

Owning Palette: [Nonlinear Systems Functions](#)

Installed With: Control Design and Simulation Module

Implements a backlash or deadband function. For example, these functions represent clearance between teeth in a gear train.

[Details](#) [Example](#)

[Dialog Box Options](#)

[Block Diagram Inputs](#)

[Block Diagram Outputs](#)

- Place on the block diagram ■ Find on the **Functions** palette

Dialog Box Options

Parameter	Description
Polymorphic instance	Specifies whether this function is Scalar or Vector . The default value is Scalar .
Parameters	Lists all the parameters associated with this function. Select a parameter from this list to configure the parameter. When you select a parameter, the parameter and its associated Parameter source control appear in the Parameter Information section of the configuration dialog box.
Preview	Displays a graphical preview, if available, of the function output or configuration.
Parameter Information	Contains the parameters you can configure for this function. You must select a parameter from the Parameters list to make that parameter and its associated Parameter source control visible in the Parameter Information section of the configuration dialog box.
Parameter source	Specifies whether you configure this parameter using the Configuration Dialog Box or a Terminal on the simulation diagram. The default value is Configuration Dialog Box . If you select Terminal , LabVIEW displays an input for that parameter on the simulation diagram, and you can wire values to that input to configure this function programmatically. If you select Configuration Dialog Box , LabVIEW removes that input from the simulation diagram. You then must set the value for this parameter inside the configuration dialog box.
initial output	Specifies the value of the output signal when this function first executes. The default value is 0.
deadband	Specifies the width of the deadband. The default is 1.

Block Diagram Inputs

Parameter	Description
initial output	Specifies the value of the output signal when this function first executes. The default value is 0.
deadband	Specifies the width of the deadband. The default is 1.
input	Specifies the signal to which you want to apply the function.

Block Diagram Outputs

Parameter	Description
output	Returns the signal that results from applying the function to the input signal.

Backlash Details

The following equations define the **output** of this function.

$$y(t) = \begin{cases} y(t-1) & \text{if } |u(t) - y(t-1)| \leq \frac{\text{deadband}}{2} \\ u(t) - \frac{\text{deadband}}{2} & \text{if } |u(t) - y(t-1)| > \frac{\text{deadband}}{2} \text{ and } u(t) > y(t-1) \\ u(t) + \frac{\text{deadband}}{2} & \text{if } |u(t) - y(t-1)| > \frac{\text{deadband}}{2} \text{ and } u(t) \leq y(t-1) \end{cases}$$

where u is the input

y is the output

t is the current simulation time

Feedthrough Behavior

All input/output pairs of this function have [direct feedthrough behavior](#).

Example

Refer to the SimEx backlash VI in the labview\examples\Control and Simulation\Simulation\Nonlinear directory for an example of using the Backlash VI.

- Open example ■ Browse related examples

Dead Zone Function

Owning Palette: [Nonlinear Systems Functions](#)

Installed With: Control Design and Simulation Module

Implements a dead zone function. For example, this function represents a forward-based diode in which you must apply a small forward voltage drop before current begins to flow. You specify the width of the dead zone using the **start of dead zone** and **end of dead zone** parameters.

[Details](#) [Example](#)

[Dialog Box Options](#)

[Block Diagram Inputs](#)

[Block Diagram Outputs](#)

■ Place on the block diagram ■ Find on the **Functions** palette

Dialog Box Options

Parameter	Description
Polymorphic instance	Specifies whether this function is Scalar or Vector . The default value is Scalar .
Parameters	Lists all the parameters associated with this function. Select a parameter from this list to configure the parameter. When you select a parameter, the parameter and its associated Parameter source control appear in the Parameter Information section of the configuration dialog box.
Preview	Displays a graphical preview, if available, of the function output or configuration.
Parameter Information	Contains the parameters you can configure for this function. You must select a parameter from the Parameters list to make that parameter and its associated Parameter source control visible in the Parameter Information section of the configuration dialog box.
Parameter source	Specifies whether you configure this parameter using the Configuration Dialog Box or a Terminal on the simulation diagram. The default value is Configuration Dialog Box . If you select Terminal , LabVIEW displays an input for that parameter on the simulation diagram, and you can wire values to that input to configure this function programmatically. If you select Configuration Dialog Box , LabVIEW removes that input from the simulation diagram. You then must set the value for this parameter inside the configuration dialog box.
start of dead zone	Specifies the beginning of the dead zone. The default is –1.
end of dead zone	Specifies the end of the dead zone. The default is 1.

Block Diagram Inputs

Parameter	Description
start of dead zone	Specifies the beginning of the dead zone. The default is -1.
end of dead zone	Specifies the end of the dead zone. The default is 1.
input	Specifies the signal to which you want to apply the function.

Block Diagram Outputs

Parameter	Description
output	Returns the signal that results from applying the function to the input signal.

Dead Zone Details

The following equations define the **output** of this function.

$$y = \begin{cases} 0 & \text{if } \mathbf{start\ of\ dead\ zone} \leq u \leq \mathbf{end\ of\ dead\ zone} \\ u - \mathbf{start\ of\ dead\ zone} & \text{if } u < \mathbf{start\ of\ dead\ zone} \\ u - \mathbf{end\ of\ dead\ zone} & \text{if } u > \mathbf{end\ of\ dead\ zone} \end{cases}$$

where u is the input

y is the output

Feedthrough Behavior

All input/output pairs of this function have [direct feedthrough behavior](#).

Example

Refer to the SimEx dead zone VI in the labview\examples\Control and Simulation\Simulation\Nonlinear directory for an example of using the Dead Zone VI.

- Open example ■ Browse related examples

Detect Zero Crossing Function

Owning Palette: [Nonlinear Systems Functions](#)

Installed With: Control Design and Simulation Module

Detects a zero crossing according to the **trigger type** you specify. A zero crossing occurs when the **input** signal you specify crosses the x-axis.

This function detects a zero crossing for both fixed and variable step-size ordinary differential equation (ODE) solvers. For variable step-size solvers, this function also forces the ODE solver to evaluate the simulation and update the simulation output when the signal crosses zero. If this function detects a zero crossing in the direction the **trigger type** parameter specifies, this function resets the ODE solver.

[Details](#) [Example](#)

[Dialog Box Options](#)

[Block Diagram Inputs](#)

[Block Diagram Outputs](#)

- Place on the block diagram ■ Find on the **Functions** palette

Dialog Box Options

Parameter	Description
Polymorphic instance	Specifies whether this function is Scalar or Vector . The default value is Scalar .
Parameters	Lists all the parameters associated with this function. Select a parameter from this list to configure the parameter. When you select a parameter, the parameter and its associated Parameter source control appear in the Parameter Information section of the configuration dialog box.
Preview	Displays a graphical preview, if available, of the function output or configuration.
Parameter Information	Contains the parameters you can configure for this function. You must select a parameter from the Parameters list to make that parameter and its associated Parameter source control visible in the Parameter Information section of the configuration dialog box.
Parameter source	Specifies whether you configure this parameter using the Configuration Dialog Box or a Terminal on the simulation diagram. The default value is Configuration Dialog Box . If you select Terminal , LabVIEW displays an input for that parameter on the simulation diagram, and you can wire values to that input to configure this function programmatically. If you select Configuration Dialog Box , LabVIEW removes that input from the simulation diagram. You then must set the value for this parameter inside the configuration dialog box.
trigger type	Specifies when this function resets the ODE solver. You can choose from the following options: <ul style="list-style-type: none">- rising—The ODE solver resets when the input crosses zero with a positive slope.- falling—The ODE solver resets when the input crosses through zero with a negative slope.- either—The ODE solver resets when the input

	<p>crosses zero with either a positive or negative slope.</p> <ul style="list-style-type: none"> - none—(Default) The ODE solver does not reset.
offset	<p>Specifies the value around which this function detects a crossing. This function detects a crossing if the following equation is true:</p> $\text{sgn}(u(t-1)-\text{offset}(t)) \neq \text{sgn}(u(t)-\text{offset}(t))$

Block Diagram Inputs

Parameter	Description
trigger type	Specifies when this function resets the ODE solver. You can choose from the following options: <ul style="list-style-type: none">- rising—The ODE solver resets when the input crosses zero with a positive slope.- falling—The ODE solver resets when the input crosses through zero with a negative slope.- either—The ODE solver resets when the input crosses zero with either a positive or negative slope.- none—(Default) The ODE solver does not reset.
input	Specifies the signal to which you want to apply the function.
offset	Specifies the value around which this function detects a crossing. This function detects a crossing if the following equation is true: $\text{sgn}(u(t-1)-\text{offset}(t)) \neq \text{sgn}(u(t)-\text{offset}(t))$

Block Diagram Outputs

Parameter	Description
crossed?	Returns TRUE if the input signal crossed zero in the direction that the trigger type parameter specifies, depending on the type of ODE solver. For fixed step-size ODE solvers, crossed? returns TRUE if the input signal crossed zero during the last time step. For variable step-size ODE solvers, crossed? returns TRUE when the input signal reaches zero. crossed? returns FALSE in all other situations. If crossed? is TRUE, the ODE solver resets.

Detect Zero Crossing Details

Feedthrough Behavior

All input/output pairs of this function have [direct feedthrough behavior](#).

Example

Refer to the SimEx detect zero crossing VI in the labview\examples\Control and Simulation\Simulation\Nonlinear directory for an example of using the Detect Zero Crossing VI.

- Open example ■ Browse related examples

Friction Function

Owning Palette: [Nonlinear Systems Functions](#)

Installed With: Control Design and Simulation Module

Represents friction at zero but guarantees linear behavior for nonzero inputs. You also can use this function to represent Coulomb and viscous friction.

[Details](#) [Example](#)

[Dialog Box Options](#)

[Block Diagram Inputs](#)

[Block Diagram Outputs](#)

■ Place on the block diagram ■ Find on the **Functions** palette

Dialog Box Options

Parameter	Description
Polymorphic instance	Specifies whether this function is Scalar or Vector . The default value is Scalar .
Parameters	Lists all the parameters associated with this function. Select a parameter from this list to configure the parameter. When you select a parameter, the parameter and its associated Parameter source control appear in the Parameter Information section of the configuration dialog box.
Preview	Displays a graphical preview, if available, of the function output or configuration.
Parameter Information	Contains the parameters you can configure for this function. You must select a parameter from the Parameters list to make that parameter and its associated Parameter source control visible in the Parameter Information section of the configuration dialog box.
Parameter source	Specifies whether you configure this parameter using the Configuration Dialog Box or a Terminal on the simulation diagram. The default value is Configuration Dialog Box . If you select Terminal , LabVIEW displays an input for that parameter on the simulation diagram, and you can wire values to that input to configure this function programmatically. If you select Configuration Dialog Box , LabVIEW removes that input from the simulation diagram. You then must set the value for this parameter inside the configuration dialog box.
offset	Specifies the offset of the friction element.
gain	Specifies the slope of the friction element.

Block Diagram Inputs

Parameter	Description
offset	Specifies the offset of the friction element.
gain	Specifies the slope of the friction element.
input	Specifies the signal to which you want to apply the function.

Block Diagram Outputs

Parameter	Description
output	Returns the signal that results from applying the function to the input signal.

Friction Details

The following equations define the **output** of this function.

$$y = \begin{cases} 0 & \text{if } u = 0 \\ u * \text{gain} + \text{offset} & \text{if } u > 0 \\ u * \text{gain} - \text{offset} & \text{if } u < 0 \end{cases}$$

where u is the input

y is the output

Viscous and Coulomb Friction

If you set the value of the **gain** parameter to 0, this function models Coulomb friction.

If you set the value of the **offset** parameter to 0, this function models viscous friction.

Feedthrough Behavior

All input/output pairs of this function have [direct feedthrough behavior](#).

Example

Refer to the SimEx friction VI in the labview\examples\Control and Simulation\Simulation\Nonlinear directory for an example of using the Friction VI.

- Open example ■ Browse related examples

Quantizer Function

Owning Palette: [Nonlinear Systems Functions](#)

Installed With: Control Design and Simulation Module

Quantizes a continuous input signal to discrete states.

[Details](#) [Example](#)

[Dialog Box Options](#)

[Block Diagram Inputs](#)

[Block Diagram Outputs](#)

■ Place on the block diagram ■ Find on the **Functions** palette

Dialog Box Options

Parameter	Description
Polymorphic instance	Specifies whether this function is Scalar or Vector . The default value is Scalar .
Parameters	Lists all the parameters associated with this function. Select a parameter from this list to configure the parameter. When you select a parameter, the parameter and its associated Parameter source control appear in the Parameter Information section of the configuration dialog box.
Preview	Displays a graphical preview, if available, of the function output or configuration.
Parameter Information	Contains the parameters you can configure for this function. You must select a parameter from the Parameters list to make that parameter and its associated Parameter source control visible in the Parameter Information section of the configuration dialog box.
Parameter source	Specifies whether you configure this parameter using the Configuration Dialog Box or a Terminal on the simulation diagram. The default value is Configuration Dialog Box . If you select Terminal , LabVIEW displays an input for that parameter on the simulation diagram, and you can wire values to that input to configure this function programmatically. If you select Configuration Dialog Box , LabVIEW removes that input from the simulation diagram. You then must set the value for this parameter inside the configuration dialog box.
quantization interval	Specifies the height of the quantization levels. The default value is 1.

Block Diagram Inputs

Parameter	Description
quantization interval	Specifies the height of the quantization levels. The default value is 1.
input	Specifies the signal to which you want to apply the function.

Block Diagram Outputs

Parameter	Description
output	Returns the signal that results from applying the function to the input signal.

Quantizer Details

The following equations define the **output** of this function.

$$y = \left[\frac{u}{\text{quantization interval}} \right]^* \text{quantization interval}$$

where u is the input

y is the output

The square brackets indicate this function rounds the value to the nearest integer.

Feedthrough Behavior

All input/output pairs of this function have [direct feedthrough behavior](#).

Example

Refer to the SimEx quantizer VI in the labview\examples\Control and Simulation\Simulation\Nonlinear directory for an example of using the Quantizer VI.

- [Open example](#)
- [Browse related examples](#)

Rate Limiter Function

Owning Palette: [Nonlinear Systems Functions](#)

Installed With: Control Design and Simulation Module

Limits the rate of change of a signal.

[Details](#) [Example](#)

[Dialog Box Options](#)

[Block Diagram Inputs](#)

[Block Diagram Outputs](#)

■ Place on the block diagram ■ Find on the **Functions** palette

Dialog Box Options

Parameter	Description
Polymorphic instance	Specifies whether this function is Scalar or Vector . The default value is Scalar .
Parameters	Lists all the parameters associated with this function. Select a parameter from this list to configure the parameter. When you select a parameter, the parameter and its associated Parameter source control appear in the Parameter Information section of the configuration dialog box.
Preview	Displays a graphical preview, if available, of the function output or configuration.
Parameter Information	Contains the parameters you can configure for this function. You must select a parameter from the Parameters list to make that parameter and its associated Parameter source control visible in the Parameter Information section of the configuration dialog box.
Parameter source	Specifies whether you configure this parameter using the Configuration Dialog Box or a Terminal on the simulation diagram. The default value is Configuration Dialog Box . If you select Terminal , LabVIEW displays an input for that parameter on the simulation diagram, and you can wire values to that input to configure this function programmatically. If you select Configuration Dialog Box , LabVIEW removes that input from the simulation diagram. You then must set the value for this parameter inside the configuration dialog box.
negative slew rate	Specifies the lower limit of the rate of change of the signal. The default value is –1.
positive slew rate	Specifies the upper limit of the rate of change of the signal. The default value is 1.

Block Diagram Inputs

Parameter	Description
negative slew rate	Specifies the lower limit of the rate of change of the signal. The default value is –1.
positive slew rate	Specifies the upper limit of the rate of change of the signal. The default value is 1.
input	Specifies the signal to which you want to apply the function.

Block Diagram Outputs

Parameter	Description
output	Returns the signal that results from applying the function to the input signal.

Rate Limiter Details

The following equations define the **output** of this function.

Given $D = \frac{u(t) - y(t-1)}{dt}$

then

$$y = \begin{cases} \text{positive slew rate} \cdot dt + y(t-1) & \text{if } (D > \text{positive slew rate}) \\ \text{negative slew rate} \cdot dt + y(t-1) & \text{if } (D < \text{negative slew rate}) \\ u(t) & \text{otherwise} \end{cases}$$

where u is the input

y is the output

t is the current simulation time

D is the rate of change of the signal

Feedthrough Behavior

All input/output pairs of this function have [direct feedthrough behavior](#).

Example

Refer to the SimEx rate limiter VI in the labview\examples\Control and Simulation\Simulation\Nonlinear directory for an example of using the Rate Limiter VI.

- Open example ■ Browse related examples

Relay Function

Owning Palette: [Nonlinear Systems Functions](#)

Installed With: Control Design and Simulation Module

Returns one of two values based on the value of the **input** signal.

[Details](#) [Example](#)

[Dialog Box Options](#)

[Block Diagram Inputs](#)

[Block Diagram Outputs](#)

■ Place on the block diagram ■ Find on the **Functions** palette

Dialog Box Options

Parameter	Description
Polymorphic instance	Specifies whether this function is Scalar or Vector . The default value is Scalar .
Parameters	Lists all the parameters associated with this function. Select a parameter from this list to configure the parameter. When you select a parameter, the parameter and its associated Parameter source control appear in the Parameter Information section of the configuration dialog box.
Preview	Displays a graphical preview, if available, of the function output or configuration.
Parameter Information	Contains the parameters you can configure for this function. You must select a parameter from the Parameters list to make that parameter and its associated Parameter source control visible in the Parameter Information section of the configuration dialog box.
Parameter source	Specifies whether you configure this parameter using the Configuration Dialog Box or a Terminal on the simulation diagram. The default value is Configuration Dialog Box . If you select Terminal , LabVIEW displays an input for that parameter on the simulation diagram, and you can wire values to that input to configure this function programmatically. If you select Configuration Dialog Box , LabVIEW removes that input from the simulation diagram. You then must set the value for this parameter inside the configuration dialog box.
output when off	Specifies the value of the output when the relay is in the off state. The default value is -10.
output when on	Specifies the value of the output when the relay is in the on state. The default value is 10.
switch off point	Specifies the value below which the input must fall to switch from the on state to the off state. The default value is -1. The value of this parameter must be less than or

	equal to the value of the switch on point parameter, or LabVIEW returns an error.
switch on point	Specifies the value above which the input must rise to switch from the off state to the on state. The default value is 1. The value of this parameter must be greater than or equal to the value of the switch off point parameter, or LabVIEW returns an error.

Block Diagram Inputs

Parameter	Description
output when off	Specifies the value of the output when the relay is in the off state. The default value is –10.
output when on	Specifies the value of the output when the relay is in the on state. The default value is 10.
switch off point	Specifies the value below which the input must fall to switch from the on state to the off state. The default value is –1. The value of this parameter must be less than or equal to the value of the switch on point parameter, or LabVIEW returns an error.
switch on point	Specifies the value above which the input must rise to switch from the off state to the on state. The default value is 1. The value of this parameter must be greater than or equal to the value of the switch off point parameter, or LabVIEW returns an error.
input	Specifies the signal to which you want to apply the function.

Block Diagram Outputs

Parameter	Description
output	Returns the signal that results from applying the function to the input signal.

Relay Details

The following equations define the **output** of this function.

$$y = \begin{cases} \text{output when on} & \text{if the relay is in the on state} \\ \text{output when off} & \text{if the relay is in the off state} \end{cases}$$

When the relay is in the on state, the relay remains in the on state until the **input** falls below the value of the **switch off point**. When the relay is in the off state, the relay remains in the off state until the **input** exceeds the value of the **switch on point**. The relay initially is in the off state.

Feedthrough Behavior

All input/output pairs of this function have [direct feedthrough behavior](#).

Example

Refer to the SimEx relay VI in the labview\examples\Control and Simulation\Simulation\Nonlinear directory for an example of using the Relay VI.

- Open example ■ Browse related examples

Saturation Function

Owning Palette: [Nonlinear Systems Functions](#)

Installed With: Control Design and Simulation Module

Limits the valid range of a signal.

[Details](#) [Example](#)

[Dialog Box Options](#)

[Block Diagram Inputs](#)

[Block Diagram Outputs](#)

■ Place on the block diagram ■ Find on the **Functions** palette

Dialog Box Options

Parameter	Description
Polymorphic instance	Specifies whether this function is Scalar or Vector . The default value is Scalar .
Parameters	Lists all the parameters associated with this function. Select a parameter from this list to configure the parameter. When you select a parameter, the parameter and its associated Parameter source control appear in the Parameter Information section of the configuration dialog box.
Preview	Displays a graphical preview, if available, of the function output or configuration.
Parameter Information	Contains the parameters you can configure for this function. You must select a parameter from the Parameters list to make that parameter and its associated Parameter source control visible in the Parameter Information section of the configuration dialog box.
Parameter source	Specifies whether you configure this parameter using the Configuration Dialog Box or a Terminal on the simulation diagram. The default value is Configuration Dialog Box . If you select Terminal , LabVIEW displays an input for that parameter on the simulation diagram, and you can wire values to that input to configure this function programmatically. If you select Configuration Dialog Box , LabVIEW removes that input from the simulation diagram. You then must set the value for this parameter inside the configuration dialog box.
lower limit	Specifies the minimum value of the output . If the value of the input is less than the value of the lower limit , this function returns the value of the lower limit . The default value of the lower limit is –5. The value of this parameter must be less than or equal to the value of the upper limit parameter, or LabVIEW returns an error.
upper limit	Specifies the maximum value of the output . If the value

of the **input** is greater than the value of the **upper limit**, this function returns the value of the **upper limit**. The default value of the **upper limit** is 5. The value of this parameter must be greater than or equal to the value of the **lower limit** parameter, or LabVIEW returns an error.

Block Diagram Inputs

Parameter	Description
lower limit	Specifies the minimum value of the output . If the value of the input is less than the value of the lower limit , this function returns the value of the lower limit . The default value of the lower limit is –5. The value of this parameter must be less than or equal to the value of the upper limit parameter, or LabVIEW returns an error.
upper limit	Specifies the maximum value of the output . If the value of the input is greater than the value of the upper limit , this function returns the value of the upper limit . The default value of the upper limit is 5. The value of this parameter must be greater than or equal to the value of the lower limit parameter, or LabVIEW returns an error.
input	Specifies the signal to which you want to apply the function.

Block Diagram Outputs

Parameter	Description
output	Returns the signal that results from applying the function to the input signal.

Saturation Details

The following equations define the **output** of this function.

$$y = \begin{cases} \text{upper limit} & \text{if } u > \text{upper limit} \\ \text{lower limit} & \text{if } u < \text{lower limit} \\ \text{lower limit} = \text{upper limit} & \text{if } \text{lower limit} = \text{upper limit} \\ u & \text{if } \text{lower limit} \leq u \leq \text{upper limit} \end{cases}$$

where u is the input

y is the output

Feedthrough Behavior

All input/output pairs of this function have [direct feedthrough behavior](#).

Example

Refer to the SimEx saturation VI in the labview\examples\Control and Simulation\Simulation\Nonlinear directory for an example of using the Saturation VI.

- Open example ■ Browse related examples

Switch Function

Owning Palette: [Nonlinear Systems Functions](#)

Installed With: Control Design and Simulation Module

Switches the value of the **output** between **value above threshold** and **value below threshold** based on the value of the **control input**.

[Details](#) [Example](#)

[Dialog Box Options](#)

[Block Diagram Inputs](#)

[Block Diagram Outputs](#)

■ Place on the block diagram ■ Find on the **Functions** palette

Dialog Box Options

Parameter	Description
Polymorphic instance	Specifies whether this function is Scalar or Vector . The default value is Scalar .
Parameters	Lists all the parameters associated with this function. Select a parameter from this list to configure the parameter. When you select a parameter, the parameter and its associated Parameter source control appear in the Parameter Information section of the configuration dialog box.
Preview	Displays a graphical preview, if available, of the function output or configuration.
Parameter Information	Contains the parameters you can configure for this function. You must select a parameter from the Parameters list to make that parameter and its associated Parameter source control visible in the Parameter Information section of the configuration dialog box.
Parameter source	Specifies whether you configure this parameter using the Configuration Dialog Box or a Terminal on the simulation diagram. The default value is Configuration Dialog Box . If you select Terminal , LabVIEW displays an input for that parameter on the simulation diagram, and you can wire values to that input to configure this function programmatically. If you select Configuration Dialog Box , LabVIEW removes that input from the simulation diagram. You then must set the value for this parameter inside the configuration dialog box.
threshold	Specifies the point at which the output switches between value above threshold and value below threshold . The LabVIEW Control Design and Simulation Module compares the value of the control input to the value of the threshold to determine the value of the output . The default value of threshold is 1.

Block Diagram Inputs

Parameter	Description
threshold	Specifies the point at which the output switches between value above threshold and value below threshold . The LabVIEW Control Design and Simulation Module compares the value of the control input to the value of the threshold to determine the value of the output . The default value of threshold is 1.
value above threshold	Specifies the signal that this function returns when the value of the control input is greater than the threshold .
control input	Specifies the signal that this function compares against threshold to determine which input to select as the output.
value below threshold	Specifies the signal that this function returns when the value of the control input is less than or equal to the threshold .

Block Diagram Outputs

Parameter	Description
output	Returns the signal that results from applying the function to the input signal.

Switch Details

The following equations define the **output** of this function.

$$y = \begin{cases} \text{value above threshold} & \text{if controlinput} > \text{threshold} \\ \text{value below threshold} & \text{if controlinput} \leq \text{threshold} \end{cases}$$

where y is the output.

Feedthrough Behavior

All input/output pairs of this function have [direct feedthrough behavior](#).

Example

Refer to the SimEx switch VI in the labview\examples\Control and Simulation\Simulation\Nonlinear directory for an example of using the Switch VI.

- [Open example](#)
- [Browse related examples](#)

Optimal Design VIs

Owning Palette: [Simulation VIs and Functions](#)

Installed With: Control Design and Simulation Module. This topic might not match its corresponding palette in LabVIEW depending on your operating system, licensed product(s), and target.

Use the Optimal Design VIs to determine the optimal parameters for a system model given a cost function and set of constraints.



Note You do not need to place these VIs inside a [Simulation Loop](#). This palette is not available when you are developing a simulation for an embedded target.

The VIs on this palette can return [general LabVIEW error codes](#) or [specific Simulation error codes](#).

Palette Object	Description
SIM Construct Default System	Constructs a dynamic system model you can wire to the System Data input of the SIM Optimal Design VI. This dynamic system model consists of a controller, a plant, a sensor, and three filters.
SIM Optimal Design	Calculates parameter values that optimize a specified cost function while keeping parameters and outputs within specified bounds. This VI uses a finite-horizon time-domain simulation to obtain these values.

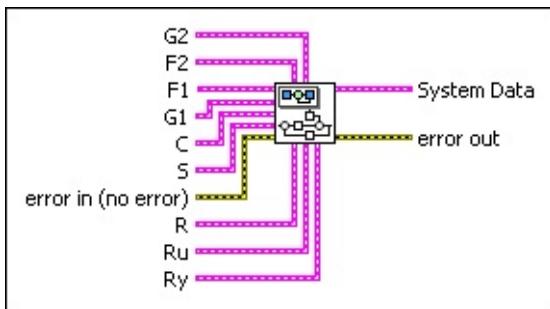
SIM Construct Default System VI

Owning Palette: [Optimal Design VIs](#)

Installed With: Control Design and Simulation Module

Constructs a dynamic system model you can wire to the **System Data** input of the [SIM Optimal Design](#) VI. This dynamic system model consists of a controller, a plant, a sensor, and three filters.

[Details](#) [Examples](#)



- Place on the block diagram ■ Find on the **Functions** palette
- **G2** specifies information about the output feedforward filter.
- **F2** specifies information about the control feedforward filter.
- **F1** specifies information about the reference filter.
- **G1** specifies information about the plant.
- **C** specifies information about the controller.
- **S** specifies information about the sensor.
- **error in** describes error conditions that occur before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function runs, it runs normally and sets its own error status in **error out**. Use the [Simple Error Handler](#) or [General Error Handler](#) VIs to display the description of the error code. Use [exception control](#) to treat what is normally an error as no error or to treat a warning as an error. Use **error in** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in** of the next node.

 **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

 **code** is the error or warning code. The default is 0. If **status** is TRUE, **code** is a nonzero [error code](#). If **status** is FALSE, **code** is 0 or a warning code.

 **source** specifies the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning. The default is an empty string.

 **R** specifies information about the reference input signal to the controller **C**. Refer to the [Details](#) section for the information about how this signal applies to the dynamic system model.

 **Signal Type** specifies the type of reference signal.

0	None —Specifies a reference signal of 0.
1	Step (default)—Specifies a step input signal. You can control the height of the step signal using the Step/impulse scaling factor parameter.
2	Impulse —Specifies an input impulse signal. You can control the height of the impulse signal using the Step/impulse scaling factor parameter.
3	User defined —Specifies a custom signal.

 **Step/impulse scaling factor** specifies the factor by which this VI scales the input signal. This parameter is valid only if you specify **Step** or **Impulse** for the **Signal Type** parameter.

 **User Defined Input** specifies a custom input signal. This parameter is valid only if you select **User defined** for the **Signal Type** parameter. The values you enter in this parameter must be uniformly-spaced. The time step between values is determined by the **Time Step** of the fixed step-size ordinary differential equation (ODE) solver you specify in the **Solver Parameters** parameter of the SIM Optimal Design VI. If you specify a variable step-size ODE solver, this VI resamples the custom input values based on

the current **Time Step** of the ODE solver.

-  **Ru** specifies information about the input signal to the plant **G1**. This input signal passes through the control feedforward filter **F2**. Refer to the [Details](#) section for the information about how this signal applies to the dynamic system model.

-  **Signal Type** specifies the type of reference signal.

0	None —Specifies a reference signal of 0.
1	Step (default)—Specifies a step input signal. You can control the height of the step signal using the Step/impulse scaling factor parameter.
2	Impulse —Specifies an input impulse signal. You can control the height of the impulse signal using the Step/impulse scaling factor parameter.
3	User defined —Specifies a custom signal.

-  **Step/impulse scaling factor** specifies the factor by which this VI scales the input signal. This parameter is valid only if you specify **Step** or **Impulse** for the **Signal Type** parameter.

-  **User Defined Input** specifies a custom input signal. This parameter is valid only if you select **User defined** for the **Signal Type** parameter. The values you enter in this parameter must be uniformly-spaced. The time step between values is determined by the **Time Step** of the fixed step-size ordinary differential equation (ODE) solver you specify in the **Solver Parameters** parameter of the SIM Optimal Design VI. If you specify a variable step-size ODE solver, this VI resamples the custom input values based on the current **Time Step** of the ODE solver.

-  **Ry** specifies information about the input signal that this VI combines with output from the plant **G1**. This input signal passes through the output feedforward filter **G2**. Refer to the [Details](#) section for the information about how this signal applies to the dynamic system model.

-  **Signal Type** specifies the type of reference signal.

0	None —Specifies a reference signal of 0.
1	Step (default)—Specifies a step input signal. You can control the height of the step signal using the Step/impulse scaling factor parameter.
2	Impulse —Specifies an input impulse signal. You can control the height of the impulse signal using the Step/impulse scaling factor parameter.
3	User defined —Specifies a custom signal.

 **Step/impulse scaling factor** specifies the factor by which this VI scales the input signal. This parameter is valid only if you specify **Step** or **Impulse** for the **Signal Type** parameter.

 **User Defined Input** specifies a custom input signal. This parameter is valid only if you select **User defined** for the **Signal Type** parameter. The values you enter in this parameter must be uniformly-spaced. The time step between values is determined by the **Time Step** of the fixed step-size ordinary differential equation (ODE) solver you specify in the **Solver Parameters** parameter of the SIM Optimal Design VI. If you specify a variable step-size ODE solver, this VI resamples the custom input values based on the current **Time Step** of the ODE solver.

 **System Data** returns information about the dynamic system that this VI constructs. You can wire this output to the **System Data** input of the SIM Optimal Design VI. Refer to the [Details](#) section for the structure of this dynamic system.

 **error out** contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

 **status** is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.

 **code** is the error or warning code. If **status** is TRUE, **code** is a nonzero [error code](#). If **status** is FALSE, **code** is 0 or a

warning code.

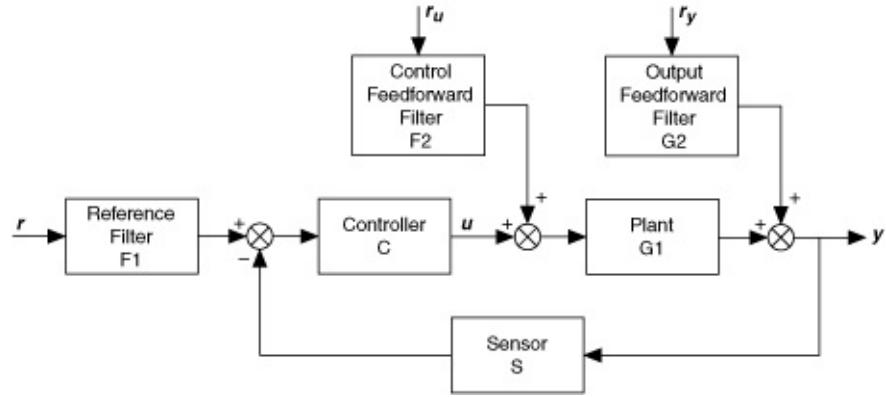


source describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning.

SIM Construct Default System Details

Dynamic System Structure

This VI constructs a dynamic system model with the following structure:



Dynamic System Components

G2, F2, F1, G1, C, and S consist of one or more transfer functions and information about the transfer function(s), such as delays.

By default, **C** is a parallel proportional integral derivative (PID) controller defined by the following equation:

$$U(s) = K_P e + \frac{K_I}{s} + \frac{K_D s}{\alpha s + 1}$$

Defining a Custom Reference Signal

The **System Data** parameter of this VI contains the **Signal Parameters Array** parameter, which specifies values for reference signals r_u , r_y , and r . If you define a custom dynamic system, r_u and r_y must be the first two elements of the **Signal Parameters Array**, respectively. You can [define custom reference signals](#) starting with the third element.

Examples

Refer to the following VIs for examples of using the SIM Construct Default System VI:

- PID Design for Second Order Continuous System VI:
labview\examples\Control and Simulation\Simulation\Optimal Control Design
 - Open example ■ Browse related examples
- PID Design for Second Order Discrete System VI:
labview\examples\Control and Simulation\Simulation\Optimal Control Design
 - Open example ■ Browse related examples

SIM Optimal Design VI

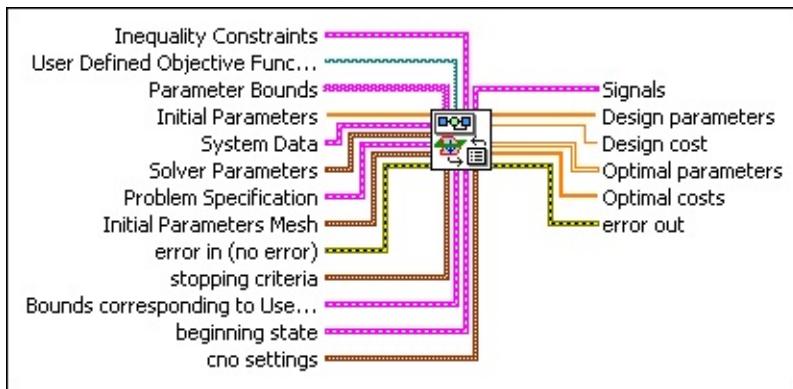
Owning Palette: [Optimal Design VIs](#)

Installed With: Control Design and Simulation Module

Calculates parameter values that [optimize](#) a specified cost function while keeping parameters and outputs within specified bounds. This VI uses a finite-horizon time-domain simulation to obtain these values.

 **Note** This VI uses the [Constrained Nonlinear Optimization](#) VI, which uses the sequential quadratic programming (SQP) algorithm to [compute](#) optimal parameter values.

[Details](#) [Examples](#)



 Place on the block diagram  Find on the **Functions** palette

 **Inequality Constraints** specifies the upper and lower envelopes within which the following variables must lie: control actions, outputs, rate of change of control actions, and rate of change of outputs. Refer to the [Details](#) section for information about these envelopes.

 **User Defined Objective Function Path** specifies the path to a custom objective function. If you define a custom objective function, you also must specify the **Bounds corresponding to User Defined Objective Function Path** parameter. You can use the SIM Objective Function template, located in the labview\templates\Simulation\Optimal Design directory, to design a custom objective function.

 **Note** When using a custom objective function, neither the **Design parameters** output nor the plots on the **Current**

Data page of this VI update.

-  **Parameter Bounds** specifies the minimum and maximum values of each parameter, forming the region in which this VI searches for optimal values. The n^{th} element of this parameter corresponds to the n^{th} element of the **Initial Parameters** parameter.
-  **Initial Parameters** specifies the parameters you want to optimize and the initial values of each parameter. This VI uses these initial values to search the parameter space for optimal values. You define the parameter space using the **Parameter Bounds** parameter. The structure of the **Initial Parameters** parameter depends on the value you specify for the **System response type** parameter. Refer to the [Details](#) section for information about the structure of this parameter.
 -  **Note** Use the **Initial Parameters Mesh** parameter to generate more sets of initial values using the method you specify.
-  **System Data** specifies information about the system on which this VI operates. You can use the [SIM Construct Default System](#) VI to obtain this information.
-  **Solver Parameters** specifies the parameters, such as the ordinary differential equation (ODE) solver, this VI uses to simulate the finite-horizon time-domain response of the dynamic system model. These parameters are identical to the ones you configure using the [Configure Simulation Parameters](#) dialog box.
-  **Problem Specification** specifies the type of system response, inequality constraints, and cost function. You can select from a pre-defined list or substitute custom types.
 -  **Note** The LabVIEW Control Design and Simulation Module includes several templates you can use to create custom specification types. To access these templates, select **File»New** to launch the [New](#) dialog box. Select **VI»From Template»Simulation»Optimal Design** from the **Create New** tree to display templates for each specification type. If you create a custom specification type that does not use a parameter from the template, do not delete the parameter. Deleting an unused parameter from a template breaks the

connector pane structure the SIM Optimal Design VI uses to operate.

 **System response type** specifies the type of system response for which this VI returns optimal parameters. Refer to the [Details](#) section for information about how this VI uses the **Initial Values** parameter depending on the **System response type** you select.

0	Default SISO - Optimize C (default)—Determines the optimal parameters of the controller C. C must be in parallel proportional integral derivative (PID) form.
1	Default SISO - Optimize F1 —Determines the optimal parameters of the reference filter F1.
2	Default SISO - Optimize F2 —Determines the optimal parameters of the control feedforward filter F2.
3	User defined (SISO/MIMO) —Specifies that you want to use a file to define a custom system response type. If you select this option, you must specify a file in the File path user defined system structure text box.

 **File path user defined system response** specifies a path to a file containing a custom system type. This parameter is valid only if you specify **User defined (SISO/MIMO)** for the **System response type** parameter.

 **Inequality constraints type** specifies the type of inequality constraints this VI places on the system. You also can define custom inequality constraints.

0	Time series constraint envelopes (default)—Specifies that this VI constrains the system using the upper and lower envelopes you specify for the Inequality Constraints parameter.
1	User defined —Specifies that you want to use a file to define custom inequality constraints. If you select this option, you must specify a file in the File path user defined inequality constraints text box.



File path user defined inequality constraints specifies a path to a file containing custom inequality constraints. This parameter is valid only if you specify **User defined** for the **Inequality constraints type** parameter.



Cost type specifies the type of cost function this VI minimizes. This VI evaluates all cost functions using discrete data points the system response calculation generates. Refer to the [Details](#) section for information about each cost function.

0	IE —Specifies a cost function that integrates the error.
1	IAE —Specifies a cost function that integrates the absolute value of the error.
2	ISE —Specifies a cost function that integrates the square of the error.
3	ITAE —Specifies a cost function that integrates the time multiplied by the absolute value of the error.
4	ITE —Specifies a cost function that integrates the time multiplied by the error.
5	ITSE —Specifies a cost function that integrates the time multiplied by the square of the error.
6	ISTE —Specifies a cost function that integrates the square of the time multiplied by the square of the error.
7	LQ (default)—Specifies a linear quadratic cost function.
8	SumOfVariances —Specifies a cost function based on the variance of the error multiplied by the variance of the control action.
9	User defined —Specifies that you want to use a file to define a custom cost function. If you select this option, you must specify a file in the File path user defined cost calculation text box.



File path user defined cost calculation specifies a path to a file containing a custom cost function. This parameter is valid only if you specify **User defined** for the **Cost type** parameter.

-  **Weights for cost function** specifies any weights this VI applies to the **Cost type** function you select.
-  **User data variant** specifies the variant data you use to pass parameters at run time.
-  **Initial Parameters Mesh** specifies the method this VI uses to generate additional sets of initial parameter values. Generating more than one set of initial parameter values increases the possibility of finding global optimal parameter values instead of local optimal parameter values. This parameter also specifies the number of initial parameter value sets to generate.
 -  **Method** specifies the method this VI uses to generate N additional points, where N equals the value you specify for the **Number of Points** parameter.

0	Uniform grid —Generates N equally spaced points between the minimum and maximum parameter values.
1	Uniform random —Generates N random points. Each point equals a random number between 0 and 1 scaled by the difference between the maximum and minimum parameter values.
2	Quasirandom (default)—Generates N quasirandom points using Halton sequences.
3	Random walk —Generates N random points within the constrained region.

-  **Number of Points** specifies the number of points for the **Initial Parameters Mesh**.

-  **error in** describes error conditions that occur before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function runs, it runs normally and sets its own error status in **error out**. Use the [Simple Error Handler](#) or [General Error Handler](#) VIs to display the description of the error code. Use [exception control](#) to treat what is normally an error as no error or to

treat a warning as an error. Use **error in** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in** of the next node.

 **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

 **code** is the error or warning code. The default is 0. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is 0 or a warning code.

 **source** specifies the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning. The default is an empty string.

 **stopping criteria** is the collection of conditions that terminate the optimization. If (**function tolerance** AND **parameter tolerance** AND **gradient tolerance**) OR **max iterations** OR **max function calls** then optimization terminates.

 **function tolerance** is the relative change in function value and is defined as $\text{abs}(\text{current } f - \text{prev } f) / (\text{abs}(\text{curr } f) + \text{machine eps})$. If the relative change in the function value falls below **function tolerance**, the optimization terminates.

 **parameter tolerance** is the relative change in parameter values and is defined as $\text{abs}(\text{current } p - \text{prev } p) / (\text{abs}(\text{curr } p) + \text{machine eps})$. If the relative change of all the parameter values falls below **parameters tolerance**, the optimization terminates.

 **max iterations** is the largest number of iterations of the major loop of the optimization. If the number of major loop iterations exceeds **max iterations**, the optimization terminates.

 **max function calls** is the largest number of objective function calls allowed before terminating the optimization process.

 **gradient tolerance** is the 2–norm of the gradient. If the 2–norm of the gradient falls below **gradient tolerance**, the optimization terminates.

-  **Bounds corresponding to User Defined Objective Function** specifies the minimum and maximum inequality constraint values for the **User Defined Objective Function**.
-  **beginning state** contains the initial values of the inequality constraint function, the lagrangian multipliers, and the hessian. **beginning state** is typically the ending state of a previous optimization and allows a warm start of the optimization.
 -  **inequality constraints** contains the value of the inequality constraint functions, typically from a previous call to the Constrained Nonlinear Optimization VI.
 -  **lagrangian multipliers** contains the value of the lagrangian multipliers, typically from a previous call to the Constrained Nonlinear Optimization VI.
 -  **hessian** contains an estimate of the hessian, typically from a previous call to the Constrained Nonlinear Optimization VI.
-  **cno settings** contains additional tolerance and termination settings that are specific to this algorithm.
 -  **constraint weight** is a relative weighting of the constraints as compared to the objective function and controls the distance that the optimization search travels outside the feasible range to seek an optimal solution. A value greater than 1 forces the search to stay close to the known feasible range, and a smaller value allows a broader range of searching.
 -  **maximum minor iterations** is an upper bound on the number of iterations allowed to solve the quadratic subproblem. If the constraint functions are highly nonlinear, the linearized constraints used in the quadratic subproblem might not be very accurate. In this case, it might be sensible to limit the number of minor iterations, forcing the linear approximation to be updated more often.
-  **Signals** returns the time response for all outputs and control actions corresponding to each element of the **Optimal costs** parameter.

 **Output Signals** returns information about the time response of the outputs.

 **Time** returns the time steps at which this VI calculated the time response.

 **y** returns the time response for the outputs. The elements of this parameter correspond with the elements of the **Time** parameter.

 **Control Signals** returns information about the time response of the control actions.

 **Time** returns the time steps at which this VI calculated the time response.

 **u** returns the time response for the control actions. The elements of this parameter correspond with the elements of the **Time** parameter.

 **Design Parameters** returns the set of parameter values that minimize the specified cost function. These values are the optimal parameter values.

 **Design Cost** returns the result of the specified cost function if you apply the values from the **Design parameters** parameter.

 **Optimal parameters** returns a list of possible optimal parameter values. Each column of this array corresponds to one parameter you specified in the **Parameter Bounds** parameter. Each row of this array corresponds to one search for optimal values.

 **Optimal costs** returns the results of the specified cost function that correspond to each row of the **Optimal parameters** array.

 **error out** contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

 **status** is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.

 **code** is the error or warning code. If **status** is TRUE, **code**

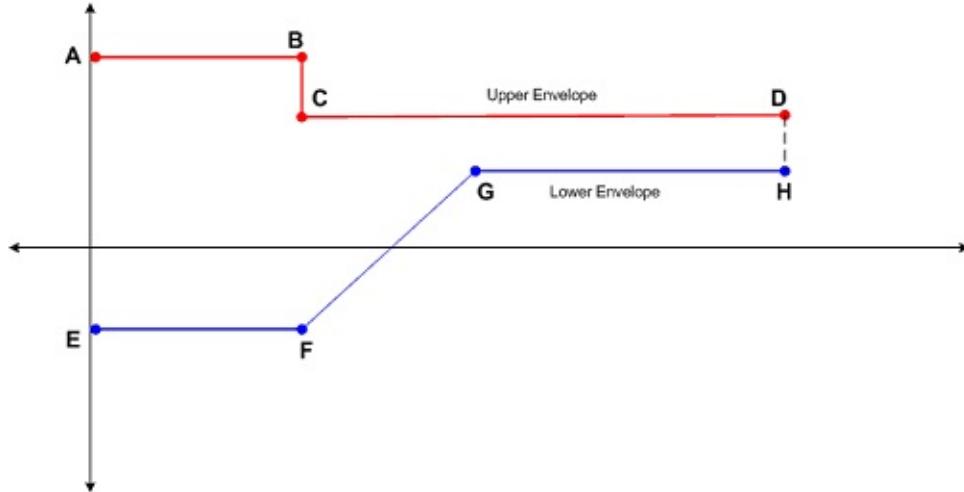
is a nonzero [error code](#). If **status** is FALSE, **code** is 0 or a warning code.

 **source** describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning.

SIM Optimal Design Details

Defining Inequality Constraint Envelopes

An envelope is a piecewise linear curve defined by the end points of linear segments. The following figure shows a typical pair of envelopes.



where points A, B, C, and D define the upper envelope and points E, F, G, and H define the lower envelope.

The **Inequality Constraints** parameter contains four arrays. Each array specifies the constraints on either the control actions, outputs, rate of change of control actions, or rate of change of outputs. Each array also includes two clusters, **Upper Envelope Points** and **Lower Envelope Points**. Both of these clusters include two arrays, **Time** and **Amplitude**. These arrays specify the x and y coordinates, respectively, that define the points that make up the envelope.



Note Instead of specifying individual points, you can use the Graphically Specify Inequality Constraints VI, located in the labview\examples\Control and Simulation\Simulation\Optimal Control Design\Graphically Specify Inequality Constraints directory, to draw the upper and lower envelopes.

Cost Function Types

The following equations define the types of cost function J you can specify for the **Cost type** parameter.

$$J_{IE} = \sum_{i=0}^T e_i(t) dt \approx \sum_i \sum_{n=0}^N (\Delta t e_i(n))$$

$$J_{IAE} = \sum_{i=0}^T |e_i(t)| dt \approx \sum_i \sum_{n=0}^N (\Delta t |e_i(n)|)$$

$$J_{ISE} = \sum_{i=0}^T e_i^2(t) dt \approx \sum_i \sum_{n=0}^N (\Delta t e_i^2(n))$$

$$J_{ITAE} = \sum_{i=0}^T t |e_i(t)| dt \approx \sum_i \sum_{n=0}^N (t \Delta t |e_i(n)|)$$

$$J_{ITE} = \sum_{i=0}^T t e_i(t) dt \approx \sum_i \sum_{n=0}^N (t \Delta t e_i(n))$$

$$J_{ITSE} = \sum_{i=0}^T t e_i^2(t) dt \approx \sum_i \sum_{n=0}^N (t \Delta t e_i^2(n))$$

$$J_{ISTE} = \sum_{i=0}^T t^2 e_i^2(t) dt \approx \sum_i \sum_{n=0}^N (t^2 \Delta t e_i^2(n))$$

$$J_{LQ} = \int_0^T [e_0(t) \dots e_P(t) u_0(t) \dots u_Q(t)] [\mathbf{w}] \begin{bmatrix} e_0(t) \\ \vdots \\ e_P(t) \\ u_0(t) \\ \vdots \\ u_Q(t) \end{bmatrix} dt$$

$$\cong \sum_{\text{all elements}} \left(\sum_{n=0}^N \Delta t [e_0(n) \dots e_P(n) u_0(n) \dots u_Q(n)] [\mathbf{w}] \begin{bmatrix} e_0(n) \\ \vdots \\ e_P(n) \\ u_0(n) \\ \vdots \\ u_Q(n) \end{bmatrix} \right)$$

$$J_{SumOfVariances} = \sum_{\text{all elements}} \left(\Delta t [\sigma^2 e_0 \dots \sigma^2 e_P \sigma^2 u_0 \dots \sigma^2 u_Q] [\mathbf{w}] \begin{bmatrix} \sigma^2 e_0 \\ \vdots \\ \sigma^2 e_P \\ \sigma^2 u_0 \\ \vdots \\ \sigma^2 u_Q \end{bmatrix} \right)$$

$$\sigma^2_{e_i} = \frac{1}{N} \sum_{n=1}^N (r_i(n) - y_i(n))^2; \sigma^2_{u_i} = \frac{1}{N} \sum_{n=1}^N (u_i(n) - \bar{u}_i)^2; \bar{u}_i = \frac{1}{N} \sum_{n=1}^N u_i(n)$$

where $e(t)$ is the error, defined as $r(t) - y(t)$

t is the simulation time

r is the reference signal

y is the output

u is the control action

n is the current time response sample

N is the maximum number of samples in the time response

\mathbf{W} is the weight matrix you define using the **Weights for cost function** parameter

i is the index of the current output channel

Specifying the System Response Type and Defining Initial Parameter Values

The structure of the **Initial Parameters** parameter depends on the value you select for the **System response type** parameter.

If you select **Default SISO - Optimize C**, this VI returns optimal gain values K_P , K_I , and K_D for a parallel PID controller defined by the following equation:

$$U(s) = K_P e + \frac{K_I}{s} + \frac{K_D s}{as + 1}$$

where $a = 0.01$.

In this situation, the first element of the **Initial Parameters** parameter must be the initial value of K_P . The second and third elements must be the initial values of K_I and K_D , respectively.

If you select **Default SISO - Optimize F1** or **Default SISO - Optimize F2**, this VI returns optimal coefficient values for the following transfer function equation that defines these filters.

$$F = \frac{a_n s^n + a_{n-1} s^{n-1} + \dots + a_1 s + a_0}{b_n s^n + b_{n-1} s^{n-1} + \dots + b_1 s + b_0}$$

In this situation, the **Initial Parameters** parameter contains the initial values of the numerator and denominator coefficients. You must separate this parameter into an equal number of numerator and denominator coefficients. Therefore, if a coefficient is 0, you must enter 0 in the appropriate element instead of leaving that element blank.

For example, if **Initial Parameters** is [5 6 0 0 0 6 3 0 9 2], this VI interprets the numerator coefficients as [5 6 0 0 0] in ascending powers of s . This VI interprets the denominator coefficients as [6 3 0 9 2] in ascending powers of s . Therefore, the following transfer function equation initially defines filter.

$$F = \frac{6s + 5}{2s^4 + 9s^3 + 3s + 6}$$

Examples

Refer to the following VIs for examples of using the SIM Optimal Design VI:

- PID Design for Second Order Continuous System VI:
labview\examples\Control and Simulation\Simulation\Optimal Control Design
 - Open example ■ Browse related examples
- PID Design for Second Order Discrete System VI:
labview\examples\Control and Simulation\Simulation\Optimal Control Design
 - Open example ■ Browse related examples

Signal Arithmetic Functions

Owning Palette: [Simulation VIs and Functions](#)

Installed With: Control Design and Simulation Module. This topic might not match its corresponding palette in LabVIEW depending on your operating system, licensed product(s), and target.

Use the Signal Arithmetic functions to perform basic arithmetic operations on signals in a simulation system.

The functions on this palette can return [general LabVIEW error codes](#) or [specific Simulation error codes](#). If you use the functions on this palette in a [Simulation Loop](#), LabVIEW sends any errors that these functions return to the **Error** output on the Output Node of the Simulation Loop.

Palette Object	Description
Gain	Multiplies the input you specify by the gain .
Multiplication	Multiplies and/or divides the input signals. This function accepts mixed scalar/vector inputs.
Summation	Adds and/or subtracts the input signals. This function accepts mixed vector/scalar inputs.

Gain Function

Owning Palette: [Signal Arithmetic Functions](#)

Installed With: Control Design and Simulation Module

Multiplies the **input** you specify by the **gain**.

[Details](#)

[Dialog Box Options](#)

[Block Diagram Inputs](#)

[Block Diagram Outputs](#)

■ Place on the block diagram ■ Find on the **Functions** palette

Dialog Box Options

Parameter	Description
Polymorphic instance	Specifies information about the input and gain of this function. You can select from the following options: <ul style="list-style-type: none">• Scalar—(Default) Specifies that this function takes a scalar input and applies a scalar gain.• Vector—Specifies that this function takes a vector input and applies a vector gain.• Vector,ScalarGain—Specifies that this function takes a vector input and applies a scalar gain.• Vector,MatrixGain—Specifies that this function performs the matrix multiplication $\mathbf{Y} = \mathbf{G} \mathbf{U}$, where \mathbf{Y} is the $m \times 1$ output matrix, \mathbf{G} is the $m \times n$ gain matrix, and \mathbf{U} is the $n \times 1$ input vector.• Dot product—Specifies that this function calculates the dot product of the input and gain vectors. If you select this option, these vectors must be of equal length.
Parameters	Lists all the parameters associated with this function. Select a parameter from this list to configure the parameter. When you select a parameter, the parameter and its associated Parameter source control appear in the Parameter Information section of the configuration dialog box.
Parameter Information	Contains the parameters you can configure for this function. You must select a parameter from the Parameters list to make that parameter and its associated Parameter source control visible in the Parameter Information section of the configuration dialog box.
Parameter source	Specifies whether you configure this parameter using the Configuration Dialog Box or a Terminal on the simulation diagram. The default value is Configuration Dialog Box . If you select Terminal , LabVIEW displays an input for that parameter on the simulation diagram, and

	you can wire values to that input to configure this function programmatically. If you select Configuration Dialog Box , LabVIEW removes that input from the simulation diagram. You then must set the value for this parameter inside the configuration dialog box.
gain	Specifies the value by which you want to multiply the value of the input .

Block Diagram Inputs

Parameter	Description
gain	Specifies the value by which you want to multiply the value of the input .
input	Specifies the signal to which you want to apply the gain.

Block Diagram Outputs

Parameter	Description
output	Returns the product of the input signal and the gain.

Gain Details

Use this function in place of the LabVIEW [Multiply](#) function to represent gain on the simulation diagram.

Feedthrough Behavior

All input/output pairs of this function have [direct feedthrough behavior](#).

Multiplication Function

Owning Palette: [Signal Arithmetic Functions](#)

Installed With: Control Design and Simulation Module

Multiplies and/or divides the input signals. This function accepts mixed scalar/vector inputs.

[Details](#)

[Dialog Box Options](#)

[Block Diagram Inputs](#)

[Block Diagram Outputs](#)

■ Place on the block diagram ■ Find on the **Functions** palette

Dialog Box Options

Parameter	Description
Icon shape	Changes the display style of the icon. You can select to display the icon as a circle or a rectangle. The default is Circle .
Inputs	Specifies the number of input terminals to use.
Picture	Displays the icon of the function. You can click a terminal to switch its operation mode between multiply, divide, and no operation.

Block Diagram Inputs

Parameter	Description
Operand1	(Operand1–Operandn) specify the input signals to multiply or divide.

Block Diagram Outputs

Parameter	Description
Result	Returns the result of the function.

Multiplication Details

Use this function in place of the LabVIEW [Compound Arithmetic](#) function to represent multiplication or division on the simulation diagram.

Feedthrough Behavior

All input/output pairs of this function have [direct feedthrough behavior](#).

Summation Function

Owning Palette: [Signal Arithmetic Functions](#)

Installed With: Control Design and Simulation Module

Adds and/or subtracts the input signals. This function accepts mixed vector/scalar inputs.

[Details](#)

[Dialog Box Options](#)

[Block Diagram Inputs](#)

[Block Diagram Outputs](#)

■ Place on the block diagram ■ Find on the **Functions** palette

Dialog Box Options

Parameter	Description
Icon shape	Changes the display style of the icon. You can select to display the icon as a circle or a rectangle. The default is Circle .
Inputs	Specifies the number of input terminals to use.
Picture	Displays the icon of the function. You can click a terminal to switch its operation mode between add, subtract, and no operation.

Block Diagram Inputs

Parameter	Description
Operand1	(Operand1–Operandn) specify the input signals to add or subtract.

Block Diagram Outputs

Parameter	Description
Result	Returns the result of the function.

Summation Details

Use this function in place of the LabVIEW [Compound Arithmetic](#) function to represent addition or subtraction on the simulation diagram.

Feedthrough Behavior

All input/output pairs of this function have [direct feedthrough behavior](#).

Signal Generation Functions

Owning Palette: [Simulation VIs and Functions](#)

Installed With: Control Design and Simulation Module. This topic might not match its corresponding palette in LabVIEW depending on your operating system, licensed product(s), and target.

Use the Signal Generation functions to generate specific waveform patterns and to index signals into an array.

 **Note** If you configure a parameter for a Signal Generation function programmatically, the function updates its output value using only the current parameters and ignores the previous parameters. If you change the function parameters while the ordinary differential equation (ODE) solver is evaluating the simulation, you might introduce a discontinuity into the generated signal.

The functions on this palette can return [general LabVIEW error codes](#) or [specific Simulation error codes](#). If you use the functions on this palette in a [Simulation Loop](#), LabVIEW sends any errors that these functions return to the **Error** output on the Output Node of the Simulation Loop.

Palette Object	Description
Chirp Signal	Generates the point-by-point value of a chirp signal.
Indexer	Indexes a waveform or array by the current simulation time.
Pulse Signal	Generates the point-by-point value of a pulse train signal. A pulse train signal is a signal whose value switches between zero and a specified amplitude. The current value depends on the current simulation time.
Ramp Signal	Generates the point-by-point value of a ramp signal.
Signal Generator	Generates the point-by-point value of a periodic signal.
Simulation Time	Returns the current simulation time. This value is useful if you terminate the simulation before the final time and you want to know the time at which the simulation terminated. This value also is useful if you want to generate a signal

	based upon the simulation time without using the Signal Generation functions. This function does not have a configuration dialog box.
Sine Signal	Generates the point-by-point value of a sine signal.
Step Signal	Generates the point-by-point value of a step signal.

Chirp Signal Function

Owning Palette: [Signal Generation Functions](#)

Installed With: Control Design and Simulation Module

Generates the point-by-point value of a chirp signal.

[Details](#) [Example](#)

[Dialog Box Options](#)

[Block Diagram Inputs](#)

[Block Diagram Outputs](#)

■ Place on the block diagram ■ Find on the **Functions** palette

Dialog Box Options

Parameter	Description
Polymorphic instance	Specifies whether this function is Scalar or Vector . The default value is Scalar .
Parameters	Lists all the parameters associated with this function. Select a parameter from this list to configure the parameter. When you select a parameter, the parameter and its associated Parameter source control appear in the Parameter Information section of the configuration dialog box.
Preview	Displays a graphical preview, if available, of the function output or configuration.
Parameter Information	Contains the parameters you can configure for this function. You must select a parameter from the Parameters list to make that parameter and its associated Parameter source control visible in the Parameter Information section of the configuration dialog box.
Parameter source	Specifies whether you configure this parameter using the Configuration Dialog Box or a Terminal on the simulation diagram. The default value is Configuration Dialog Box . If you select Terminal , LabVIEW displays an input for that parameter on the simulation diagram, and you can wire values to that input to configure this function programmatically. If you select Configuration Dialog Box , LabVIEW removes that input from the simulation diagram. You then must set the value for this parameter inside the configuration dialog box.
target frequency	Specifies the frequency, in hertz, the chirp signal reaches at the target time . The default value is 0.5.
target time	Specifies the simulation time, in seconds, at which the chirp signal reaches the target frequency . After the simulation reaches the target time , the frequency continues to change at the same rate the frequency changed from initial frequency to target frequency . The

	default value of target time is 10.
initial frequency	Specifies the frequency, in hertz, of the chirp signal at the initial time of the simulation. The default value is 0.01.

Block Diagram Inputs

Parameter	Description
target frequency	Specifies the frequency, in hertz, the chirp signal reaches at the target time . The default value is 0.5.
target time	Specifies the simulation time, in seconds, at which the chirp signal reaches the target frequency . After the simulation reaches the target time , the frequency continues to change at the same rate the frequency changed from initial frequency to target frequency . The default value of target time is 10.
initial frequency	Specifies the frequency, in hertz, of the chirp signal at the initial time of the simulation. The default value is 0.01.

Block Diagram Outputs

Parameter	Description
output	Returns the value of the signal evaluated at the current simulation time. The default value is 1.

Chirp Signal Details

A chirp signal is a sine wave whose frequency varies from **initial frequency** to **target frequency** during the simulation time. The current value depends on the current simulation time.

The following equations define the chirp signal.

$$y = A * \sin(2\pi(f(t)t))$$

$$\text{where } f(t) = f_0 + \frac{(f_1 - f_0)}{T}t$$

A is the **amplitude**

t is the current simulation time

f_0 is the **initial frequency**

f_1 is the **target frequency**

T is the number of time steps between the simulation initial time and the function **target time**

y is the output

Feedthrough Behavior

All input/output pairs of this function have [direct feedthrough behavior](#).

Example

Refer to the SimEx Chirp VI in the labview\examples\Control and Simulation\Simulation\Signal Generation directory for an example of using the Chirp Signal VI.

- Open example ■ Browse related examples

Indexer Function

Owning Palette: [Utilities Functions](#)

Installed With: Control Design and Simulation Module

Indexes a waveform or array by the current simulation time.

[Details](#) [Example](#)

[Dialog Box Options](#)

[Block Diagram Inputs](#)

[Block Diagram Outputs](#)

■ Place on the block diagram ■ Find on the **Functions** palette

Dialog Box Options

Parameter	Description
Polymorphic instance	Specifies whether the type of the Input to this function is an Array , Point Array , Waveform , XY Graph , 2D Array , or 2D Point Array .
Parameters	Lists all the parameters associated with this function. Select a parameter from this list to configure the parameter. When you select a parameter, the parameter and its associated Parameter source control appear in the Parameter Information section of the configuration dialog box.
Preview	Displays a graphical preview, if available, of the function output or configuration.
Parameter Information	Contains the parameters you can configure for this function. You must select a parameter from the Parameters list to make that parameter and its associated Parameter source control visible in the Parameter Information section of the configuration dialog box.
Parameter source	Specifies whether you configure this parameter using the Configuration Dialog Box or a Terminal on the simulation diagram. The default value is Configuration Dialog Box . If you select Terminal , LabVIEW displays an input for that parameter on the simulation diagram, and you can wire values to that input to configure this function programmatically. If you select Configuration Dialog Box , LabVIEW removes that input from the simulation diagram. You then must set the value for this parameter inside the configuration dialog box.
Extrapolate?	Extrapolate? is TRUE if you want this function to extrapolate linearly values outside of the values you specify in the Input . Extrapolate? is FALSE if you do not want this function to extrapolate linearly values outside the value you specify in the Input . The default value is FALSE.

Indexing Mode	<p>Specifies the method by which this function indexes the Input. The available methods depend the value you specify for the Polymorphic instance parameter. If you specify the Array or 2D Array version of this function, you can select from the following methods:</p> <ul style="list-style-type: none"> - Fraction of Simulation Time Interval—(Default) Specifies that this function determines the time interval, or duration, between data points in the array by dividing the interval between the initial time and the final time of the simulation by the number of data points in the array. - Base Time Step—Specifies that this function uses the time step of the simulation as the time interval, or duration, between data points in the array. <p>If you specify the Waveform version of the function, you can select from the following methods:</p> <ul style="list-style-type: none"> - Set Waveform t0 to Simulation Initial Time—(Default) Specifies that this function equates the simulation initial time to the t0 of the waveform and indexes the waveform as the simulation time progresses. If the simulation time exceeds the waveform time, the function fixes the output to the final value of the waveform. - Use actual Waveform t0 and assume Simulation Time in seconds—Specifies that this function indexes the waveform directly from the simulation time. If the simulation time is less than the t0 of the waveform, the function sets the output equal to the t0 waveform value. If the simulation time exceeds the waveform time, the function fixes the output to the final value of the waveform.
Indexed Dimension	<p>Specifies which dimension of this two-dimensional array contains the array index. You can choose from Columns or Rows. This parameter is valid only when you select 2D Array from the Polymorphic instance pull-down menu.</p>

Block Diagram Inputs

Parameter	Description
Extrapolate?	Extrapolate? is TRUE if you want this function to extrapolate linearly values outside of the values you specify in the Input . Extrapolate? is FALSE if you do not want this function to extrapolate linearly values outside the value you specify in the Input . The default value is FALSE.
Input	Specifies the input to this function. This input either is a waveform, XY graph, point array, array, two-dimensional array, or two-dimensional point array, depending on the value you select from the Polymorphic instance parameter on the configuration dialog box of this function. You can choose from the following options: <ul style="list-style-type: none">- Array—Specifies an array of values to index by the current simulation time step. This function returns the n^{th} value of the array where n is the current time step of the simulation. This function indexes the array according to the Indexing Mode you specify. You also can select the 2D Array option to specify the input is a two-dimensional array.- Point Array—Specifies an array of x, y points to index by simulation time. This function returns the y value corresponding to the x value that matches the current simulation time. If the simulation time lies between two x values, this function linearly interpolates the result. You also can select the 2D Point Array option to specify a two-dimensional point array.- Waveform—(Default) Specifies a waveform containing Y values to index by simulation time. This function outputs the Y value corresponding to the point in the waveform whose time matches to the current simulation time. If the simulation time lies between two time steps in the waveform,

this function linearly interpolates the result. This function indexes the array according to the **Indexing Mode** you specify. Waveform inputs include the following components:

- **t0**—Specifies the start time of the waveform.
- **Y**—Contains the data values of the waveform.
- **dt**—Specifies the time interval, or duration, between data points in the waveform.
- **XY Graph**—Specifies an XY graph containing an array of x, y points to index by simulation time. This function returns the y value corresponding to the x value that matches to the current simulation time. If the simulation time lies between two x values, this function linearly interpolates the result. XY graph inputs include the following components:
 - **X Array**—Contains the time values of the XY graph that correspond to the data values stored in **Y Array**.
 - **Y Array**—Contains the data values of the XY graph that correspond to the time values stored in **X Array**.

Indexing Mode	Specifies the method by which this function indexes the Input . The available methods depend the value you specify for the Polymorphic instance parameter. If you specify the Array or 2D Array version of this function, you can select from the following methods: <ul style="list-style-type: none">- Fraction of Simulation Time Interval—(Default) Specifies that this function determines the time interval, or duration, between data points in the array by dividing the interval between the initial time and the final time of the simulation by the number of data points in the array.- Base Time Step—Specifies that this function
----------------------	--

uses the time step of the simulation as the time interval, or duration, between data points in the array.

If you specify the **Waveform** version of the function, you can select from the following methods:

- **Set Waveform t0 to Simulation Initial Time**—
(Default) Specifies that this function equates the simulation initial time to the **t0** of the waveform and indexes the waveform as the simulation time progresses. If the simulation time exceeds the waveform time, the function fixes the output to the final value of the waveform.
- **Use actual Waveform t0 and assume Simulation Time in seconds**—Specifies that this function indexes the waveform directly from the simulation time. If the simulation time is less than the **t0** of the waveform, the function sets the output equal to the **t0** waveform value. If the simulation time exceeds the waveform time, the function fixes the output to the final value of the waveform.

Indexed Dimension	Specifies which dimension of this two-dimensional array contains the array index. You can choose from Columns or Rows . This parameter is valid only when you select 2D Array from the Polymorphic instance pull-down menu.
--------------------------	---

Block Diagram Outputs

Parameter	Description
Interpolated value	Returns the output value associated with the current simulation time using the method you specify for the Indexing Mode parameter.

Indexer Details

This function operates like the [auto-indexing](#) input tunnel of a [While Loop](#) or [For Loop](#). However unlike the auto-indexing input tunnel, this function indexes the waveform or array based on the simulation time rather than the iteration of the loop.

This function linearly interpolates the value if there is no point in the signal that directly corresponds to the current simulation time.

Feedthrough Behavior

All input/output pairs of this function have [direct feedthrough behavior](#).

Example

Refer to the indexer VI in the labview\examples\Control and Simulation\Simulation\Utilities directory for an example of using the Indexer VI.

- [Open example](#)
- [Browse related examples](#)

Pulse Signal Function

Owning Palette: [Signal Generation Functions](#)

Installed With: Control Design and Simulation Module

Generates the point-by-point value of a pulse train signal. A pulse train signal is a signal whose value switches between zero and a specified amplitude. The current value depends on the current simulation time.

[Details](#) [Example](#)

[Dialog Box Options](#)

[Block Diagram Inputs](#)

[Block Diagram Outputs](#)

■ Place on the block diagram ■ Find on the **Functions** palette

Dialog Box Options

Parameter	Description
Polymorphic instance	Specifies whether this function is Scalar or Vector . The default value is Scalar .
Parameters	Lists all the parameters associated with this function. Select a parameter from this list to configure the parameter. When you select a parameter, the parameter and its associated Parameter source control appear in the Parameter Information section of the configuration dialog box.
Preview	Displays a graphical preview, if available, of the function output or configuration.
Parameter Information	Contains the parameters you can configure for this function. You must select a parameter from the Parameters list to make that parameter and its associated Parameter source control visible in the Parameter Information section of the configuration dialog box.
Parameter source	Specifies whether you configure this parameter using the Configuration Dialog Box or a Terminal on the simulation diagram. The default value is Configuration Dialog Box . If you select Terminal , LabVIEW displays an input for that parameter on the simulation diagram, and you can wire values to that input to configure this function programmatically. If you select Configuration Dialog Box , LabVIEW removes that input from the simulation diagram. You then must set the value for this parameter inside the configuration dialog box.
start time	Specifies the simulation time, in seconds, at which this function triggers the pulse signal. The default value is 0.
amplitude	Specifies the maximum amplitude of the pulse signal. The default value is 1.
offset	Specifies the amount by which to offset the y-values of the signal.
duty cycle	Specifies the ratio, expressed as a percent, between the

	high and low levels of the pulse signal. The default value is 50.
period	Specifies the period, in seconds, of the pulse signal. The default value is 2.

Block Diagram Inputs

Parameter	Description
start time	Specifies the simulation time, in seconds, at which this function triggers the pulse signal. The default value is 0.
amplitude	Specifies the maximum amplitude of the pulse signal. The default value is 1.
offset	Specifies the amount by which to offset the y-values of the signal.
duty cycle	Specifies the ratio, expressed as a percent, between the high and low levels of the pulse signal. The default value is 50.
period	Specifies the period, in seconds, of the pulse signal. The default value is 2.

Block Diagram Outputs

Parameter	Description
output	Returns the value of the signal evaluated at the current simulation time. The default value is 1.

Pulse Signal Details

The following equations define the pulse train signal.

$$y = A * \text{square}$$

where $\text{square} = \begin{cases} 1 & \text{if } \left(2\pi ft + \frac{\pi p}{180}\right) \bmod(2\pi) < \frac{\pi*D}{100} \\ -1 & \text{otherwise} \end{cases}$

A is the **amplitude**

f is the **frequency**

t is the current simulation time

p is the **phase**

D is the **duty cycle**

y is the output

Feedthrough Behavior

All input/output pairs of this function have [direct feedthrough behavior](#).

Example

Refer to the SimEx Pulse VI in the labview\examples\Control and Simulation\Simulation\Signal Generation directory for an example of using the Pulse Signal VI.

- Open example ■ Browse related examples

Signal Generator Function

Owning Palette: [Signal Generation Functions](#)

Installed With: Control Design and Simulation Module

Generates the point-by-point value of a periodic signal.

[Details](#) [Example](#)

[Dialog Box Options](#)

[Block Diagram Inputs](#)

[Block Diagram Outputs](#)

■ Place on the block diagram ■ Find on the **Functions** palette

Dialog Box Options

Parameter	Description
Polymorphic instance	Specifies whether this function is Scalar or Vector . The default value is Scalar .
Parameters	Lists all the parameters associated with this function. Select a parameter from this list to configure the parameter. When you select a parameter, the parameter and its associated Parameter source control appear in the Parameter Information section of the configuration dialog box.
Preview	Displays a graphical preview, if available, of the function output or configuration.
Parameter Information	Contains the parameters you can configure for this function. You must select a parameter from the Parameters list to make that parameter and its associated Parameter source control visible in the Parameter Information section of the configuration dialog box.
Parameter source	Specifies whether you configure this parameter using the Configuration Dialog Box or a Terminal on the simulation diagram. The default value is Configuration Dialog Box . If you select Terminal , LabVIEW displays an input for that parameter on the simulation diagram, and you can wire values to that input to configure this function programmatically. If you select Configuration Dialog Box , LabVIEW removes that input from the simulation diagram. You then must set the value for this parameter inside the configuration dialog box.
signal type	Specifies the type of signal to generate. You can generate the following types of signals: Sine (Default), Square , Sawtooth , or Random . The Random signal is uniformly distributed.
amplitude	Specifies the maximum amplitude of the signal. The default value is 1.
offset	Specifies the amount by which to offset the y-values of

	the signal.
frequency	Specifies the frequency, in hertz, of the signal. If you are generating a Random signal, this function ignores the value of the frequency parameter. The default value is 0.5.
phase	Specifies the phase, in degrees, of the signal. If you are generating a Random signal, this function ignores the value of the phase parameter. The default value is 0.

Block Diagram Inputs

Parameter	Description
signal type	Specifies the type of signal to generate. You can generate the following types of signals: Sine (Default), Square , Sawtooth , or Random . The Random signal is uniformly distributed.
amplitude	Specifies the maximum amplitude of the signal. The default value is 1.
offset	Specifies the amount by which to offset the y-values of the signal.
frequency	Specifies the frequency, in hertz, of the signal. If you are generating a Random signal, this function ignores the value of the frequency parameter. The default value is 0.5.
phase	Specifies the phase, in degrees, of the signal. If you are generating a Random signal, this function ignores the value of the phase parameter. The default value is 0.

Block Diagram Outputs

Parameter	Description
output	Returns the value of the signal evaluated at the current simulation time. The default value is 1.

Signal Generator Details

The following equations define the signals you can generate with this function.

Sine:

$$y = A * \sin\left(2\pi ft + \frac{p}{180}\right)$$

where **A** is the **amplitude**

f is the **frequency**

t is the current simulation time

p is the **phase**

y is the output

Square:

$$y = A * \text{square}$$

where $\text{square} = \begin{cases} 1 & \text{if } \left(2\pi ft + \frac{p}{180}\right) \bmod(2\pi) < \pi \\ -1 & \text{otherwise} \end{cases}$

A is the **amplitude**

f is the **frequency**

t is the current simulation time

p is the **phase**

y is the output

Sawtooth:

$$y = A * \text{sawtooth}(q)$$

where $\text{sawtooth}(q) = 1 - \frac{q}{\pi}$

$$q = \left(2\pi ft + \frac{p}{180}\right) \bmod(2\pi)$$

A is the **amplitude**

f is the **frequency**

t is the current simulation time

p is the **phase**

y is the output

Random:

The function returns a uniformly distributed random value between
-amplitude and **amplitude**.

The current value of any signal depends on the current simulation time.

Feedthrough Behavior

All input/output pairs of this function have [direct feedthrough behavior](#).

Example

Refer to the SimEx Signal Generator VI in the labview\examples\Control and Simulation\Simulation\Signal Generation directory for an example of using the Signal Generator VI.

- [Open example](#)
- [Browse related examples](#)

Simulation Time Function

Owning Palette: [Utilities Functions](#)

Installed With: Control Design and Simulation Module

Returns the current simulation time. This value is useful if you terminate the simulation before the final time and you want to know the time at which the simulation terminated. This value also is useful if you want to generate a signal based upon the simulation time without using the [Signal Generation](#) functions. This function does not have a configuration dialog box.

[Block Diagram Outputs](#)

- Place on the block diagram ■ Find on the **Functions** palette

Block Diagram Outputs

Parameter	Description
simulation time	Returns the current time of the simulation, in seconds.

Sine Signal Function

Owning Palette: [Signal Generation Functions](#)

Installed With: Control Design and Simulation Module

Generates the point-by-point value of a sine signal.

[Details](#) [Example](#)

[Dialog Box Options](#)

[Block Diagram Inputs](#)

[Block Diagram Outputs](#)

■ Place on the block diagram ■ Find on the **Functions** palette

Dialog Box Options

Parameter	Description
Polymorphic instance	Specifies whether this function is Scalar or Vector . The default value is Scalar .
Parameters	Lists all the parameters associated with this function. Select a parameter from this list to configure the parameter. When you select a parameter, the parameter and its associated Parameter source control appear in the Parameter Information section of the configuration dialog box.
Preview	Displays a graphical preview, if available, of the function output or configuration.
Parameter Information	Contains the parameters you can configure for this function. You must select a parameter from the Parameters list to make that parameter and its associated Parameter source control visible in the Parameter Information section of the configuration dialog box.
Parameter source	Specifies whether you configure this parameter using the Configuration Dialog Box or a Terminal on the simulation diagram. The default value is Configuration Dialog Box . If you select Terminal , LabVIEW displays an input for that parameter on the simulation diagram, and you can wire values to that input to configure this function programmatically. If you select Configuration Dialog Box , LabVIEW removes that input from the simulation diagram. You then must set the value for this parameter inside the configuration dialog box.
amplitude	Specifies the maximum amplitude of the sine signal. The default value is 1.
offset	Specifies the amount by which to offset the y-values of the signal.
frequency	Specifies the frequency, in hertz, of the sine signal. The default value is 0.5.
phase	Specifies the phase, in degrees, of the sine signal. The

|default value is 0.

Block Diagram Inputs

Parameter	Description
amplitude	Specifies the maximum amplitude of the sine signal. The default value is 1.
offset	Specifies the amount by which to offset the y-values of the signal.
frequency	Specifies the frequency, in hertz, of the sine signal. The default value is 0.5.
phase	Specifies the phase, in degrees, of the sine signal. The default value is 0.

Block Diagram Outputs

Parameter	Description
output	Returns the value of the signal evaluated at the current simulation time. The default value is 1.

Sine Signal Details

The following equation defines the sine signal.

$$y = A \cdot \sin\left(2\pi f t + \frac{\pi p}{180}\right)$$

where **A** is the **amplitude**

f is the **frequency**

t is the current simulation time

p is the **phase**

y is the output

The current value of the signal depends on the current simulation time.

Feedthrough Behavior

All input/output pairs of this function have [direct feedthrough behavior](#).

Example

Refer to the SimEx Sine VI in the labview\examples\Control and Simulation\Simulation\Signal Generation directory for an example of using the Sine Signal VI.

- Open example ■ Browse related examples

Ramp Signal Function

Owning Palette: [Signal Generation Functions](#)

Installed With: Control Design and Simulation Module

Generates the point-by-point value of a ramp signal.

[Details](#) [Example](#)

[Dialog Box Options](#)

[Block Diagram Inputs](#)

[Block Diagram Outputs](#)

■ Place on the block diagram ■ Find on the **Functions** palette

Dialog Box Options

Parameter	Description
Polymorphic instance	Specifies whether this function is Scalar or Vector . The default value is Scalar .
Parameters	Lists all the parameters associated with this function. Select a parameter from this list to configure the parameter. When you select a parameter, the parameter and its associated Parameter source control appear in the Parameter Information section of the configuration dialog box.
Preview	Displays a graphical preview, if available, of the function output or configuration.
Parameter Information	Contains the parameters you can configure for this function. You must select a parameter from the Parameters list to make that parameter and its associated Parameter source control visible in the Parameter Information section of the configuration dialog box.
Parameter source	Specifies whether you configure this parameter using the Configuration Dialog Box or a Terminal on the simulation diagram. The default value is Configuration Dialog Box . If you select Terminal , LabVIEW displays an input for that parameter on the simulation diagram, and you can wire values to that input to configure this function programmatically. If you select Configuration Dialog Box , LabVIEW removes that input from the simulation diagram. You then must set the value for this parameter inside the configuration dialog box.
initial output	Specifies the y offset of the ramp signal from simulation initial time to start time . The default value is 0.
start time	Specifies the simulation time, in seconds, of the start of the rising edge of the ramp signal. The default value is 0.
slope	Specifies the slew rate of the ramp signal.

Block Diagram Inputs

Parameter	Description
initial output	Specifies the y offset of the ramp signal from simulation initial time to start time . The default value is 0.
start time	Specifies the simulation time, in seconds, of the start of the rising edge of the ramp signal. The default value is 0.
slope	Specifies the slew rate of the ramp signal.

Block Diagram Outputs

Parameter	Description
output	Returns the value of the signal evaluated at the current simulation time. The default value is 1.

Ramp Signal Details

The following equations define the ramp signal.

$$y(t) = \begin{cases} y(0) + m*t & \text{if } (t > \text{start time}) \\ y(0) & \text{if } (t \leq \text{start time}) \end{cases}$$

where m is the **slope**

t is the current simulation time

y is the output

A ramp signal begins at an initial value and changes over time according to the **slope** you specify. The current value depends on the current simulation time.

Feedthrough Behavior

All input/output pairs of this function have [direct feedthrough behavior](#).

Example

Refer to the SimEx Ramp VI in the labview\examples\Control and Simulation\Simulation\Signal Generation directory for an example of using the Ramp Signal VI.

- [Open example](#)
- [Browse related examples](#)

Step Signal Function

Owning Palette: [Signal Generation Functions](#)

Installed With: Control Design and Simulation Module

Generates the point-by-point value of a step signal.

[Details](#) [Example](#)

[Dialog Box Options](#)

[Block Diagram Inputs](#)

[Block Diagram Outputs](#)

■ Place on the block diagram ■ Find on the **Functions** palette

Dialog Box Options

Parameter	Description
Polymorphic instance	Specifies whether this function is Scalar or Vector . The default value is Scalar .
Parameters	Lists all the parameters associated with this function. Select a parameter from this list to configure the parameter. When you select a parameter, the parameter and its associated Parameter source control appear in the Parameter Information section of the configuration dialog box.
Preview	Displays a graphical preview, if available, of the function output or configuration.
Parameter Information	Contains the parameters you can configure for this function. You must select a parameter from the Parameters list to make that parameter and its associated Parameter source control visible in the Parameter Information section of the configuration dialog box.
Parameter source	Specifies whether you configure this parameter using the Configuration Dialog Box or a Terminal on the simulation diagram. The default value is Configuration Dialog Box . If you select Terminal , LabVIEW displays an input for that parameter on the simulation diagram, and you can wire values to that input to configure this function programmatically. If you select Configuration Dialog Box , LabVIEW removes that input from the simulation diagram. You then must set the value for this parameter inside the configuration dialog box.
initial value	Specifies the lower limit of the step signal. The default value is 0.
final value	Specifies the upper limit of the step signal. The default value is 1.
step time	Specifies the simulation time, in seconds, at which the signal value changes from the initial value to the final value . The default value is 1.

Block Diagram Inputs

Parameter	Description
initial value	Specifies the lower limit of the step signal. The default value is 0.
final value	Specifies the upper limit of the step signal. The default value is 1.
step time	Specifies the simulation time, in seconds, at which the signal value changes from the initial value to the final value . The default value is 1.

Block Diagram Outputs

Parameter	Description
output	Returns the output of this function.

Step Signal Details

The following equations define the step signal.

$$y = \begin{cases} \text{initial value} & \text{if } (t < \text{step time}) \\ \text{final value} & \text{if } (t \geq \text{step time}) \end{cases}$$

where t is the current simulation time and y is the output.

The current value of the signal depends on the current simulation time.

Feedthrough Behavior

All input/output pairs of this function have [direct feedthrough behavior](#).

Example

Refer to the SimEx Step VI in the labview\examples\Control and Simulation\Simulation\Signal Generation directory for an example of using the Step Signal VI.

- Open example ■ Browse related examples

Trim & Linearize VIs

Owning Palette: [Simulation VIs and Functions](#)

Installed With: Control Design and Simulation Module. This topic might not match its corresponding palette in LabVIEW depending on your operating system, licensed product(s), and target.

Use the Trim & Linearize VIs to trim and linearize a simulation subsystem.

 **Note** This palette is not available when you are developing a simulation for an embedded target.

Trimming a simulation subsystem involves searching for a set of states and/or inputs that satisfy state/input/output/state derivative conditions that you specify. Linearizing a nonlinear simulation subsystem involves approximating the behavior of that subsystem around a certain operating point. You can trim and/or linearize only continuous subsystems.

 **Note** You do not need to place these VIs inside a [Simulation Loop](#).

The VIs on this palette can return [general LabVIEW error codes](#) or [specific Simulation error codes](#).

Palette Object	Description
SIM Get Parameter Names	Returns the names, dimensions, and types of all parameters in the Parameters In input.
SIM Get Parameter Value	Returns the value and type of a parameter you specify. The data types you wire to the Item Name input and the Default Value input determine the polymorphic instance to use.
SIM Linearize	Returns a linear time-invariant (LTI) system model given a nonlinear simulation subsystem. Linearizing a nonlinear simulation subsystem involves approximating the behavior of that subsystem around an operating point. The operating point consists of the Inputs and States of the subsystem. The data type you wire to the Path input determines the polymorphic instance to use.

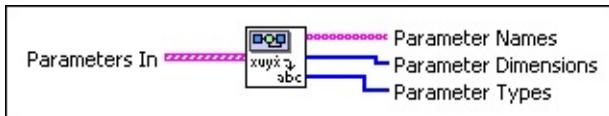
<u>SIM Query Subsystem</u>	Returns the names and values of the parameters of the states, inputs, outputs, and state derivatives of a simulation subsystem. Use these parameters with the <u>SIM Set Parameter Value</u> VI to specify conditions and values that the <u>SIM Trim</u> VI and the <u>SIM Linearize</u> VI use. The data type you wire to the Path input determines the polymorphic instance to use.
<u>SIM Set Parameter Value</u>	Provides information to the <u>SIM Linearize</u> VI and <u>SIM Trim</u> VI about which parameter values to modify and how to modify those parameter values. The data type you wire to the Item Name input or the Value input determines the polymorphic instance to use.
<u>SIM Trim</u>	Calculates the states and/or inputs of a simulation subsystem necessary to produce a set of outputs and/or state derivatives that you specify. Trimming a subsystem involves searching for a set of states and/or inputs that satisfy one or more specified output and/or state derivative conditions. The data type you wire to the Path input determines the polymorphic instance to use.

SIM Get Parameter Names VI

Owning Palette: [Trim & Linearize VIs](#)

Installed With: Control Design and Simulation Module

Returns the names, dimensions, and types of all parameters in the **Parameters In** input.



■ Place on the block diagram ■ Find on the **Functions** palette

Parameters In specifies either the **States**, **Inputs**, **Outputs**, or **State Derivatives** array that the [SIM Query Subsystem](#) VI returns.

Name specifies the name of the parameter.

Value specifies the value of the parameter.

Static Input? is TRUE if the parameter type **Static**. **Static Input?** is FALSE if the parameter type is either **Variable** or **Fixed**.

Vector? is TRUE if the parameter value is a vector or an array. **Vector?** is FALSE if the parameter value is a scalar.

Fixed Size? is TRUE if you can replace this parameter only with a vector or an array of the same size. **Fixed Size?** is FALSE if you can replace this parameter with a vector or array of any size.

Fixed Value? is TRUE if the parameter value is a condition that you specify for trimming a subsystem. **Fixed Value?** is FALSE if the parameter value is not a condition that you specify for trimming a subsystem.

Parameter Names returns the names of the parameters in the **Parameters In** array.

Parameter Dimensions returns the dimensions of the parameters in the **Parameters In** array.

0	Vector (default)—Specifies that the parameter is a vector.
1	Scalar —Specifies that the parameter is a scalar.



Parameter Types returns the types of parameters in the **Parameters In** array.

0	Variable (default)—Specifies that the SIM Trim VI can change the value of this parameter to satisfy any conditions you specify. The SIM Linearize VI includes Variable inputs and outputs in the LTI model.
1	Static —Specifies that the SIM Trim VI excludes this parameter from the search. The SIM Linearize VI excludes Static inputs and outputs from the LTI model. This VI does not distinguish between Static and Variable states and state derivatives.
2	Fixed —Specifies that the SIM Trim VI uses the value of this parameter as a condition for trimming a subsystem. The SIM Linearize VI does not distinguish between Fixed and Variable parameters.

SIM Get Parameter Value VI

Owning Palette: [Trim & Linearize VIs](#)

Installed With: Control Design and Simulation Module

Returns the value and type of a parameter you specify. The data types you wire to the **Item Name** input and the **Default Value** input determine the polymorphic instance to use.

Use the pull-down menu to select an instance of this VI.

Select an instance ▾

- Place on the block diagram ■ Find on the **Functions** palette

SIM Get Parameter Value (by Name, Scalar)



Parameters In specifies either the **States**, **Inputs**, **Outputs**, or **State Derivatives** array that the [SIM Query Subsystem](#) VI returns.

- Name** specifies the name of the parameter.
- Value** specifies the value of the parameter.
- Static Input?** is TRUE if the parameter type **Static**. **Static Input?** is FALSE if the parameter type is either **Variable** or **Fixed**.
- Vector?** is TRUE if the parameter value is a vector or an array. **Vector?** is FALSE if the parameter value is a scalar.
- Fixed Size?** is TRUE if you can replace this parameter only with a vector or an array of the same size. **Fixed Size?** is FALSE if you can replace this parameter with a vector or array of any size.
- Fixed Value?** is TRUE if the parameter value is a condition that you specify for trimming a subsystem. **Fixed Value?** is FALSE if the parameter value is not a condition that you specify for trimming a subsystem.



Item Name specifies the name of the parameter in the **Parameters In** array for which this VI returns a **Value**.



Default Value specifies the **Value** this VI returns if the parameter you specify is not in the **Parameters In** array. The default value is NaN. The data type you wire to this parameter determines the polymorphic instance to use. If you leave this parameter unwired, you must [manually select the polymorphic instance](#) to use.



error in describes error conditions that occur before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function runs, it runs normally and sets its own error status in **error out**. Use the [Simple Error Handler](#) or [General Error Handler](#) VIs to display the description of the error code. Use

[exception control](#) to treat what is normally an error as no error or to treat a warning as an error. Use **error in** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in** of the next node.

 **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

 **code** is the error or warning code. The default is 0. If **status** is TRUE, **code** is a nonzero [error code](#). If **status** is FALSE, **code** is 0 or a warning code.

 **source** specifies the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning. The default is an empty string.

Parameters Out returns the elements of **Parameters In**.

 **Name** returns the name of the parameter.

 **Value** returns the value of the parameter.

 **Static Input?** returns TRUE if the parameter type is **Static**. **Static Input?** returns FALSE if the parameter type is either **Variable** or **Fixed**.

 **Vector?** returns TRUE if the parameter value is a vector or an array. **Vector?** returns FALSE if the parameter value is a scalar.

 **Fixed Size?** returns TRUE if you can replace this parameter only with a vector or an array of the same size. **Fixed Size?** returns FALSE if you can replace this parameter with a vector or array of any size.

 **Fixed Value?** returns TRUE if the parameter value is a condition that you specify for trimming a subsystem. **Fixed Value?** returns FALSE if the parameter value is not a condition that you specify for trimming a subsystem.

Parameter Type returns the type of the parameter you specified.

0 **Variable** (default)—Indicates that the [SIM Trim](#) VI can change

	the value of this parameter to satisfy any conditions you specify. The SIM Linearize VI includes Variable inputs and outputs in the linear time-invariant (LTI) model.
1	Static —Indicates that the SIM Trim VI excludes this parameter from the search. The SIM Linearize VI excludes Static inputs and outputs from the LTI model. This VI does not distinguish between Static and Variable states and state derivatives.
2	Fixed —Indicates that the SIM Trim VI uses the value of this parameter as a condition for trimming a subsystem. The SIM Linearize VI does not distinguish between Fixed and Variable parameters.



Value returns the value of the parameter you specified.



error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.



status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.

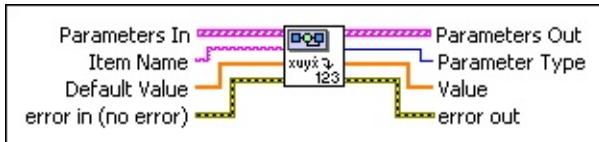


code is the error or warning code. If **status** is TRUE, **code** is a nonzero [error code](#). If **status** is FALSE, **code** is 0 or a warning code.



source describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning.

SIM Get Parameter Value (by Name, Vector)



- Parameters In specifies either the **States**, **Inputs**, **Outputs**, or **State Derivatives** array that the [SIM Query Subsystem](#) VI returns.
 - Name** specifies the name of the parameter.
 - Value** specifies the value of the parameter.
 - Static Input?** is TRUE if the parameter type **Static**. **Static Input?** is FALSE if the parameter type is either **Variable** or **Fixed**.
 - Vector?** is TRUE if the parameter value is a vector or an array. **Vector?** is FALSE if the parameter value is a scalar.
 - Fixed Size?** is TRUE if you can replace this parameter only with a vector or an array of the same size. **Fixed Size?** is FALSE if you can replace this parameter with a vector or array of any size.
 - Fixed Value?** is TRUE if the parameter value is a condition that you specify for trimming a subsystem. **Fixed Value?** is FALSE if the parameter value is not a condition that you specify for trimming a subsystem.
- Item Name** specifies the name of the parameter in the **Parameters In** array for which this VI returns a **Value**.
- Default Value** specifies the **Value** this VI returns if the parameter you specify is not in the **Parameters In** array.
- error in** describes error conditions that occur before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function runs, it runs normally and sets its own error status in **error out**. Use the [Simple Error Handler](#) or [General Error Handler](#) VIs to display the description of the error code. Use [exception control](#) to treat what is normally an error as no error or to

treat a warning as an error. Use **error in** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in** of the next node.

-  **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.
-  **code** is the error or warning code. The default is 0. If **status** is TRUE, **code** is a nonzero [error code](#). If **status** is FALSE, **code** is 0 or a warning code.
-  **source** specifies the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning. The default is an empty string.

Parameters Out returns the elements of **Parameters In**.

-  **Name** returns the name of the parameter.
-  **Value** returns the value of the parameter.
-  **Static Input?** returns TRUE if the parameter type is **Static**. **Static Input?** returns FALSE if the parameter type is either **Variable** or **Fixed**.
-  **Vector?** returns TRUE if the parameter value is a vector or an array. **Vector?** returns FALSE if the parameter value is a scalar.
-  **Fixed Size?** returns TRUE if you can replace this parameter only with a vector or an array of the same size. **Fixed Size?** returns FALSE if you can replace this parameter with a vector or array of any size.
-  **Fixed Value?** returns TRUE if the parameter value is a condition that you specify for trimming a subsystem. **Fixed Value?** returns FALSE if the parameter value is not a condition that you specify for trimming a subsystem.

Parameter Type returns the type of the parameter you specified.

- | | |
|---|--|
| 0 | Variable (default)—Indicates that the SIM Trim VI can change the value of this parameter to satisfy any conditions you specify. |
|---|--|

	The SIM Linearize VI includes Variable inputs and outputs in the linear time-invariant (LTI) model.
1	Static —Indicates that the SIM Trim VI excludes this parameter from the search. The SIM Linearize VI excludes Static inputs and outputs from the LTI model. This VI does not distinguish between Static and Variable states and state derivatives.
2	Fixed —Indicates that the SIM Trim VI uses the value of this parameter as a condition for trimming a subsystem. The SIM Linearize VI does not distinguish between Fixed and Variable parameters.



Value returns the values of the parameter you specified.



error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.



status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.



code is the error or warning code. If **status** is TRUE, **code** is a nonzero [error code](#). If **status** is FALSE, **code** is 0 or a warning code.



source describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning.

SIM Get Parameter Value (by Index, Scalar)



Parameters In specifies either the **States**, **Inputs**, **Outputs**, or **State Derivatives** array that the [SIM Query Subsystem](#) VI returns.

- Name** specifies the name of the parameter.
- Value** specifies the value of the parameter.
- Static Input?** is TRUE if the parameter type **Static**. **Static Input?** is FALSE if the parameter type is either **Variable** or **Fixed**.
- Vector?** is TRUE if the parameter value is a vector or an array. **Vector?** is FALSE if the parameter value is a scalar.
- Fixed Size?** is TRUE if you can replace this parameter only with a vector or an array of the same size. **Fixed Size?** is FALSE if you can replace this parameter with a vector or array of any size.
- Fixed Value?** is TRUE if the parameter value is a condition that you specify for trimming a subsystem. **Fixed Value?** is FALSE if the parameter value is not a condition that you specify for trimming a subsystem.

- Item Index** specifies the index number of the parameter in the **Parameters In** array for which this VI returns a **Value**.
- Default Value** specifies the **Value** this VI returns if the parameter you specify is not in the **Parameters In** array. The default value is NaN. The data type you wire to this parameter determines the polymorphic instance to use. If you leave this parameter unwired, you must [manually select the polymorphic instance](#) to use.
- error in** describes error conditions that occur before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function runs, it runs normally and sets its own error status in **error out**. Use the [Simple Error Handler](#) or [General Error Handler](#) VIs to display the description of the error code. Use

[exception control](#) to treat what is normally an error as no error or to treat a warning as an error. Use **error in** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in** of the next node.

 **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

 **code** is the error or warning code. The default is 0. If **status** is TRUE, **code** is a nonzero [error code](#). If **status** is FALSE, **code** is 0 or a warning code.

 **source** specifies the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning. The default is an empty string.

Parameters Out returns the elements of **Parameters In**.

 **Name** returns the name of the parameter.

 **Value** returns the value of the parameter.

 **Static Input?** returns TRUE if the parameter type is **Static**. **Static Input?** returns FALSE if the parameter type is either **Variable** or **Fixed**.

 **Vector?** returns TRUE if the parameter value is a vector or an array. **Vector?** returns FALSE if the parameter value is a scalar.

 **Fixed Size?** returns TRUE if you can replace this parameter only with a vector or an array of the same size. **Fixed Size?** returns FALSE if you can replace this parameter with a vector or array of any size.

 **Fixed Value?** returns TRUE if the parameter value is a condition that you specify for trimming a subsystem. **Fixed Value?** returns FALSE if the parameter value is not a condition that you specify for trimming a subsystem.

Parameter Type returns the type of the parameter you specified.

0 **Variable** (default)—Indicates that the [SIM Trim](#) VI can change

	the value of this parameter to satisfy any conditions you specify. The SIM Linearize VI includes Variable inputs and outputs in the linear time-invariant (LTI) model.
1	Static —Indicates that the SIM Trim VI excludes this parameter from the search. The SIM Linearize VI excludes Static inputs and outputs from the LTI model. This VI does not distinguish between Static and Variable states and state derivatives.
2	Fixed —Indicates that the SIM Trim VI uses the value of this parameter as a condition for trimming a subsystem. The SIM Linearize VI does not distinguish between Fixed and Variable parameters.



Value returns the value of the parameter you specified.



error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.



status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.



code is the error or warning code. If **status** is TRUE, **code** is a nonzero [error code](#). If **status** is FALSE, **code** is 0 or a warning code.



source describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning.

SIM Get Parameter Value (by Index, Vector)



Parameters In specifies either the **States**, **Inputs**, **Outputs**, or **State Derivatives** array that the [SIM Query Subsystem](#) VI returns.

- Name** specifies the name of the parameter.
- Value** specifies the value of the parameter.
- Static Input?** is TRUE if the parameter type **Static**. **Static Input?** is FALSE if the parameter type is either **Variable** or **Fixed**.
- Vector?** is TRUE if the parameter value is a vector or an array. **Vector?** is FALSE if the parameter value is a scalar.
- Fixed Size?** is TRUE if you can replace this parameter only with a vector or an array of the same size. **Fixed Size?** is FALSE if you can replace this parameter with a vector or array of any size.
- Fixed Value?** is TRUE if the parameter value is a condition that you specify for trimming a subsystem. **Fixed Value?** is FALSE if the parameter value is not a condition that you specify for trimming a subsystem.

- Item Index** specifies the index number of the parameter in the **Parameters In** array for which this VI returns a **Value**.
- Default Value** specifies the **Value** this VI returns if the parameter you specify is not in the **Parameters In** array.
- error in** describes error conditions that occur before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function runs, it runs normally and sets its own error status in **error out**. Use the [Simple Error Handler](#) or [General Error Handler](#) VIs to display the description of the error code. Use [exception control](#) to treat what is normally an error as no error or to treat a warning as an error. Use **error in** and **error out** to check errors and to specify execution order by wiring **error out** from one

node to **error in** of the next node.

-  **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.
-  **code** is the error or warning code. The default is 0. If **status** is TRUE, **code** is a nonzero [error code](#). If **status** is FALSE, **code** is 0 or a warning code.
-  **source** specifies the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning. The default is an empty string.

 **Parameters Out** returns the elements of **Parameters In**.

-  **Name** returns the name of the parameter.
-  **Value** returns the value of the parameter.
-  **Static Input?** returns TRUE if the parameter type is **Static**. **Static Input?** returns FALSE if the parameter type is either **Variable** or **Fixed**.
-  **Vector?** returns TRUE if the parameter value is a vector or an array. **Vector?** returns FALSE if the parameter value is a scalar.
-  **Fixed Size?** returns TRUE if you can replace this parameter only with a vector or an array of the same size. **Fixed Size?** returns FALSE if you can replace this parameter with a vector or array of any size.
-  **Fixed Value?** returns TRUE if the parameter value is a condition that you specify for trimming a subsystem. **Fixed Value?** returns FALSE if the parameter value is not a condition that you specify for trimming a subsystem.

 **Parameter Type** returns the type of the parameter you specified.

0	Variable (default)—Indicates that the SIM Trim VI can change the value of this parameter to satisfy any conditions you specify. The SIM Linearize VI includes Variable inputs and outputs in the linear time-invariant (LTI) model.
---	---

1	Static —Indicates that the SIM Trim VI excludes this parameter from the search. The SIM Linearize VI excludes Static inputs and outputs from the LTI model. This VI does not distinguish between Static and Variable states and state derivatives.
2	Fixed —Indicates that the SIM Trim VI uses the value of this parameter as a condition for trimming a subsystem. The SIM Linearize VI does not distinguish between Fixed and Variable parameters.



Value returns the values of the parameter you specified.



error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.



status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.



code is the error or warning code. If **status** is TRUE, **code** is a nonzero [error code](#). If **status** is FALSE, **code** is 0 or a warning code.



source describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning.

SIM Linearize VI

Owning Palette: [Trim & Linearize VIs](#)

Installed With: Control Design and Simulation Module

Returns a linear time-invariant (LTI) system model given a nonlinear simulation subsystem. Linearizing a nonlinear simulation subsystem involves approximating the behavior of that subsystem around an operating point. The operating point consists of the **Inputs** and **States** of the subsystem. The data type you wire to the **Path** input determines the polymorphic instance to use.



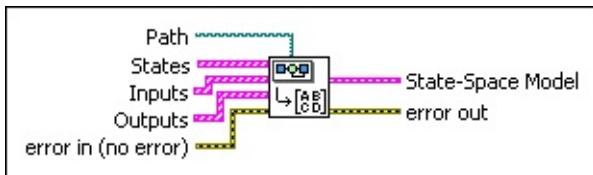
Note You can use this VI to linearize only a continuous subsystem.

Use the pull-down menu to select an instance of this VI.

Select an instance 

 Place on the block diagram  Find on the **Functions** palette

SIM Linearize (Path)



- [Dashed Box]** **Path** specifies the path to the simulation subsystem on which you want to operate.
- [Front Panel]** **States** specifies the initial states of the simulation subsystem. The SIM Linearize VI uses the initial subsystem states to calculate the operating point. You can specify the value of these states using the [SIM Set Parameter Value](#) VI.
 - [abc]** **Name** specifies the name of the parameter.
 - [DBL]** **Value** specifies the value of the parameter.
 - [TF]** **Static Input?** is TRUE if the parameter type **Static**. The SIM Linearize VI excludes **Static** inputs and outputs from the linear time-invariant (LTI) model. **Static Input?** is FALSE if the parameter type is either **Variable** or **Fixed**.
 - [TF]** **Vector?** is TRUE if the parameter value is a vector or an array. **Vector?** is FALSE if the parameter value is a scalar.
 - [TF]** **Fixed Size?** is TRUE if you can replace this parameter only with a vector or an array of the same size. **Fixed Size?** is FALSE if you can replace this parameter with a vector or array of any size.
 - [TF]** **Fixed Value?** is TRUE if the parameter value is a condition that you specify for trimming a subsystem. **Fixed Value?** is FALSE if the parameter value is not a condition that you specify for trimming a subsystem.
- [Front Panel]** **Inputs** specifies the initial inputs of the simulation subsystem. The SIM Linearize VI uses the initial subsystem inputs to calculate the operating point. You specify the inputs to include in the LTI model and the value of these inputs using the [SIM Set Parameter Value](#) VI.
 - [abc]** **Name** specifies the name of the parameter.
 - [DBL]** **Value** specifies the value of the parameter.

-  **Static Input?** is TRUE if the parameter type **Static**. The SIM Linearize VI excludes **Static** inputs and outputs from the linear time-invariant (LTI) model. **Static Input?** is FALSE if the parameter type is either **Variable** or **Fixed**.
 -  **Vector?** is TRUE if the parameter value is a vector or an array. **Vector?** is FALSE if the parameter value is a scalar.
 -  **Fixed Size?** is TRUE if you can replace this parameter only with a vector or an array of the same size. **Fixed Size?** is FALSE if you can replace this parameter with a vector or array of any size.
 -  **Fixed Value?** is TRUE if the parameter value is a condition that you specify for trimming a subsystem. **Fixed Value?** is FALSE if the parameter value is not a condition that you specify for trimming a subsystem.
-  **Outputs** specifies the outputs to include in the LTI model. You can exclude outputs from the LTI model using the SIM Set Parameter Value VI.
-  **Name** specifies the name of the parameter.
 -  **Value** specifies the value of the parameter.
-  **Static Input?** is TRUE if the parameter type **Static**. The SIM Linearize VI excludes **Static** inputs and outputs from the linear time-invariant (LTI) model. **Static Input?** is FALSE if the parameter type is either **Variable** or **Fixed**.
 -  **Vector?** is TRUE if the parameter value is a vector or an array. **Vector?** is FALSE if the parameter value is a scalar.
 -  **Fixed Size?** is TRUE if you can replace this parameter only with a vector or an array of the same size. **Fixed Size?** is FALSE if you can replace this parameter with a vector or array of any size.
 -  **Fixed Value?** is TRUE if the parameter value is a condition that you specify for trimming a subsystem. **Fixed Value?** is FALSE if the parameter value is not a condition that you specify for trimming a subsystem.
-  **error in** describes error conditions that occur before this VI or

function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function runs, it runs normally and sets its own error status in **error out**. Use the [Simple Error Handler](#) or [General Error Handler](#) VIs to display the description of the error code. Use [exception control](#) to treat what is normally an error as no error or to treat a warning as an error. Use **error in** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in** of the next node.

 **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

 **code** is the error or warning code. The default is 0. If **status** is TRUE, **code** is a nonzero [error code](#). If **status** is FALSE, **code** is 0 or a warning code.

 **source** specifies the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning. The default is an empty string.

 **State-Space Model** returns the LTI state-space form of the given simulation subsystem. This model is compatible with the [Control Design](#) VIs and functions.

 **error out** contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

 **status** is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.

 **code** is the error or warning code. If **status** is TRUE, **code** is a nonzero [error code](#). If **status** is FALSE, **code** is 0 or a warning code.

 **source** describes the origin of the error or warning and is, in

most cases, the name of the VI or function that produced the error or warning.

SIM Linearize (Reference)



Reference specifies a reference to the subsystem on which you want to operate.

- States** specifies the initial states of the simulation subsystem. The SIM Linearize VI uses the initial subsystem states to calculate the operating point. You can specify the value of these states using the [SIM Set Parameter Value](#) VI.
 - Name** specifies the name of the parameter.
 - Value** specifies the value of the parameter.
 - Static Input?** is TRUE if the parameter type **Static**. The SIM Linearize VI excludes **Static** inputs and outputs from the linear time-invariant (LTI) model. **Static Input?** is FALSE if the parameter type is either **Variable** or **Fixed**.
 - Vector?** is TRUE if the parameter value is a vector or an array. **Vector?** is FALSE if the parameter value is a scalar.
 - Fixed Size?** is TRUE if you can replace this parameter only with a vector or an array of the same size. **Fixed Size?** is FALSE if you can replace this parameter with a vector or array of any size.
 - Fixed Value?** is TRUE if the parameter value is a condition that you specify for trimming a subsystem. **Fixed Value?** is FALSE if the parameter value is not a condition that you specify for trimming a subsystem.
- Inputs** specifies the initial inputs of the simulation subsystem. The SIM Linearize VI uses the initial subsystem inputs to calculate the operating point. You specify the inputs to include in the LTI model and the value of these inputs using the [SIM Set Parameter Value](#) VI.
 - Name** specifies the name of the parameter.
 - Value** specifies the value of the parameter.
 - Static Input?** is TRUE if the parameter type **Static**. The SIM Linearize VI excludes **Static** inputs and outputs from

the linear time-invariant (LTI) model. **Static Input?** is FALSE if the parameter type is either **Variable** or **Fixed**.

-  **Vector?** is TRUE if the parameter value is a vector or an array. **Vector?** is FALSE if the parameter value is a scalar.
-  **Fixed Size?** is TRUE if you can replace this parameter only with a vector or an array of the same size. **Fixed Size?** is FALSE if you can replace this parameter with a vector or array of any size.
-  **Fixed Value?** is TRUE if the parameter value is a condition that you specify for trimming a subsystem. **Fixed Value?** is FALSE if the parameter value is not a condition that you specify for trimming a subsystem.
-  **Outputs** specifies the outputs to include in the LTI model. You can exclude outputs from the LTI model using the SIM Set Parameter Value VI.
 -  **Name** specifies the name of the parameter.
 -  **Value** specifies the value of the parameter.
 -  **Static Input?** is TRUE if the parameter type **Static**. The SIM Linearize VI excludes **Static** inputs and outputs from the linear time-invariant (LTI) model. **Static Input?** is FALSE if the parameter type is either **Variable** or **Fixed**.
 -  **Vector?** is TRUE if the parameter value is a vector or an array. **Vector?** is FALSE if the parameter value is a scalar.
 -  **Fixed Size?** is TRUE if you can replace this parameter only with a vector or an array of the same size. **Fixed Size?** is FALSE if you can replace this parameter with a vector or array of any size.
 -  **Fixed Value?** is TRUE if the parameter value is a condition that you specify for trimming a subsystem. **Fixed Value?** is FALSE if the parameter value is not a condition that you specify for trimming a subsystem.

-  **error in** describes error conditions that occur before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error

occurred before this VI or function runs. If an error occurs while this VI or function runs, it runs normally and sets its own error status in **error out**. Use the [Simple Error Handler](#) or [General Error Handler](#) VIs to display the description of the error code. Use [exception control](#) to treat what is normally an error as no error or to treat a warning as an error. Use **error in** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in** of the next node.

 **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

 **code** is the error or warning code. The default is 0. If **status** is TRUE, **code** is a nonzero [error code](#). If **status** is FALSE, **code** is 0 or a warning code.

 **source** specifies the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning. The default is an empty string.

 **State-Space Model** returns the LTI state-space form of the given simulation subsystem. This model is compatible with the [Control Design](#) VIs and functions.

 **error out** contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

 **status** is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.

 **code** is the error or warning code. If **status** is TRUE, **code** is a nonzero [error code](#). If **status** is FALSE, **code** is 0 or a warning code.

 **source** describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning.

SIM Query Subsystem VI

Owning Palette: [Trim & Linearize VIs](#)

Installed With: Control Design and Simulation Module

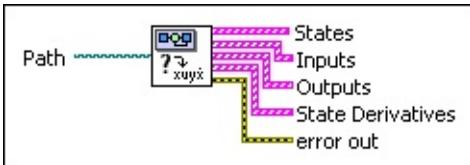
Returns the names and values of the parameters of the states, inputs, outputs, and state derivatives of a simulation subsystem. Use these parameters with the [SIM Set Parameter Value](#) VI to specify conditions and values that the [SIM Trim](#) VI and the [SIM Linearize](#) VI use. The data type you wire to the **Path** input determines the polymorphic instance to use.

Use the pull-down menu to select an instance of this VI.

Select an instance ▾

Place on the block diagram Find on the **Functions** palette

SIM Query Subsystem (Path)



- Path** specifies the path to the simulation subsystem on which you want to operate.
- States** returns the initial states of the simulation subsystem.
 - Name** returns the name of the parameter.
 - Value** returns the value of the parameter.
 - Static Input?** returns TRUE if the parameter type is **Static**. **Static Input?** returns FALSE if the parameter type is either **Variable** or **Fixed**.
 - Vector?** returns TRUE if the parameter value is a vector or an array. **Vector?** returns FALSE if the parameter value is a scalar.
 - Fixed Size?** returns TRUE if you can replace this parameter only with a vector or an array of the same size. **Fixed Size?** returns FALSE if you can replace this parameter with a vector or array of any size.
 - Fixed Value?** returns TRUE if the parameter value is a condition that you specify for trimming a subsystem. **Fixed Value?** returns FALSE if the parameter value is not a condition that you specify for trimming a subsystem.
- Inputs** returns the current values of the inputs of the simulation subsystem.
 - Name** returns the name of the parameter.
 - Value** returns the value of the parameter.
 - Static Input?** returns TRUE if the parameter type is **Static**. **Static Input?** returns FALSE if the parameter type is either **Variable** or **Fixed**.
 - Vector?** returns TRUE if the parameter value is a vector or an array. **Vector?** returns FALSE if the parameter value is a

scalar.

-  **Fixed Size?** returns TRUE if you can replace this parameter only with a vector or an array of the same size. **Fixed Size?** returns FALSE if you can replace this parameter with a vector or array of any size.
-  **Fixed Value?** returns TRUE if the parameter value is a condition that you specify for trimming a subsystem. **Fixed Value?** returns FALSE if the parameter value is not a condition that you specify for trimming a subsystem.

 **Outputs** returns the current values of the outputs of the simulation subsystem.

-  **Name** returns the name of the parameter.
-  **Value** returns the value of the parameter.
-  **Static Input?** returns TRUE if the parameter type is **Static**. **Static Input?** returns FALSE if the parameter type is either **Variable** or **Fixed**.
-  **Vector?** returns TRUE if the parameter value is a vector or an array. **Vector?** returns FALSE if the parameter value is a scalar.
-  **Fixed Size?** returns TRUE if you can replace this parameter only with a vector or an array of the same size. **Fixed Size?** returns FALSE if you can replace this parameter with a vector or array of any size.
-  **Fixed Value?** returns TRUE if the parameter value is a condition that you specify for trimming a subsystem. **Fixed Value?** returns FALSE if the parameter value is not a condition that you specify for trimming a subsystem.

 **State Derivatives** returns the derivatives of the states of the simulation subsystem. By default, **State Derivatives** contains all zeros of type **Fixed**, which indicates that the [SIM Trim](#) VI attempts to satisfy a steady-state condition.

-  **Name** returns the name of the parameter.
-  **Value** returns the value of the parameter.
-  **Static Input?** returns TRUE if the parameter type is **Static**.

Static Input? returns FALSE if the parameter type is either **Variable** or **Fixed**.

 **Vector?** returns TRUE if the parameter value is a vector or an array. **Vector?** returns FALSE if the parameter value is a scalar.

 **Fixed Size?** returns TRUE if you can replace this parameter only with a vector or an array of the same size. **Fixed Size?** returns FALSE if you can replace this parameter with a vector or array of any size.

 **Fixed Value?** returns TRUE if the parameter value is a condition that you specify for trimming a subsystem. **Fixed Value?** returns FALSE if the parameter value is not a condition that you specify for trimming a subsystem.

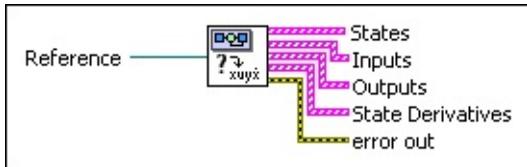
 **error out** contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

 **status** is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.

 **code** is the error or warning code. If **status** is TRUE, **code** is a nonzero **error code**. If **status** is FALSE, **code** is 0 or a warning code.

 **source** describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning.

SIM Query Subsystem (Reference)



- Reference** specifies a reference to the subsystem on which you want to operate.
- States** returns the initial states of the simulation subsystem.
 - Name** returns the name of the parameter.
 - Value** returns the value of the parameter.
 - Static Input?** returns TRUE if the parameter type is **Static**. **Static Input?** returns FALSE if the parameter type is either **Variable** or **Fixed**.
 - Vector?** returns TRUE if the parameter value is a vector or an array. **Vector?** returns FALSE if the parameter value is a scalar.
 - Fixed Size?** returns TRUE if you can replace this parameter only with a vector or an array of the same size. **Fixed Size?** returns FALSE if you can replace this parameter with a vector or array of any size.
 - Fixed Value?** returns TRUE if the parameter value is a condition that you specify for trimming a subsystem. **Fixed Value?** returns FALSE if the parameter value is not a condition that you specify for trimming a subsystem.
- Inputs** returns the current values of the inputs of the simulation subsystem.
 - Name** returns the name of the parameter.
 - Value** returns the value of the parameter.
 - Static Input?** returns TRUE if the parameter type is **Static**. **Static Input?** returns FALSE if the parameter type is either **Variable** or **Fixed**.
 - Vector?** returns TRUE if the parameter value is a vector or an array. **Vector?** returns FALSE if the parameter value is a

scalar.

 **Fixed Size?** returns TRUE if you can replace this parameter only with a vector or an array of the same size. **Fixed Size?** returns FALSE if you can replace this parameter with a vector or array of any size.

 **Fixed Value?** returns TRUE if the parameter value is a condition that you specify for trimming a subsystem. **Fixed Value?** returns FALSE if the parameter value is not a condition that you specify for trimming a subsystem.

 **Outputs** returns the current values of the outputs of the simulation subsystem.

 **Name** returns the name of the parameter.

 **Value** returns the value of the parameter.

 **Static Input?** returns TRUE if the parameter type is **Static**. **Static Input?** returns FALSE if the parameter type is either **Variable** or **Fixed**.

 **Vector?** returns TRUE if the parameter value is a vector or an array. **Vector?** returns FALSE if the parameter value is a scalar.

 **Fixed Size?** returns TRUE if you can replace this parameter only with a vector or an array of the same size. **Fixed Size?** returns FALSE if you can replace this parameter with a vector or array of any size.

 **Fixed Value?** returns TRUE if the parameter value is a condition that you specify for trimming a subsystem. **Fixed Value?** returns FALSE if the parameter value is not a condition that you specify for trimming a subsystem.

 **State Derivatives** returns the derivatives of the states of the simulation subsystem. By default, **State Derivatives** contains all zeros of type **Fixed**, which indicates that the [SIM Trim](#) VI attempts to satisfy a steady-state condition.

 **Name** returns the name of the parameter.

 **Value** returns the value of the parameter.

 **Static Input?** returns TRUE if the parameter type is **Static**.

Static Input? returns FALSE if the parameter type is either **Variable** or **Fixed**.

 **Vector?** returns TRUE if the parameter value is a vector or an array. **Vector?** returns FALSE if the parameter value is a scalar.

 **Fixed Size?** returns TRUE if you can replace this parameter only with a vector or an array of the same size. **Fixed Size?** returns FALSE if you can replace this parameter with a vector or array of any size.

 **Fixed Value?** returns TRUE if the parameter value is a condition that you specify for trimming a subsystem. **Fixed Value?** returns FALSE if the parameter value is not a condition that you specify for trimming a subsystem.

 **error out** contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

 **status** is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.

 **code** is the error or warning code. If **status** is TRUE, **code** is a nonzero **error code**. If **status** is FALSE, **code** is 0 or a warning code.

 **source** describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning.

SIM Set Parameter Value VI

Owning Palette: [Trim & Linearize VIs](#)

Installed With: Control Design and Simulation Module

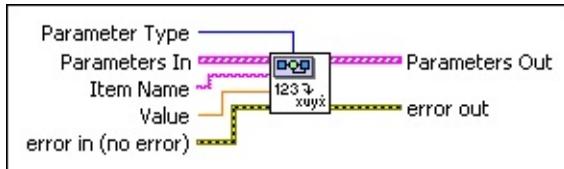
Provides information to the [SIM Linearize VI](#) and [SIM Trim](#) VI about which parameter values to modify and how to modify those parameter values. The data type you wire to the **Item Name** input or the **Value** input determines the polymorphic instance to use.

Use the pull-down menu to select an instance of this VI.

Select an instance ▾

- Place on the block diagram ■ Find on the **Functions** palette

SIM Set Parameter Value (by Name, Scalar)



Parameter Type specifies the type of parameter.

0	Variable (default)—Specifies that the SIM Trim VI can change the value of this parameter to satisfy any conditions you specify. The SIM Linearize VI includes Variable inputs and outputs in the LTI model.
1	Static —Specifies that the SIM Trim VI excludes this parameter from the search. The SIM Linearize VI excludes Static inputs and outputs from the LTI model. This VI does not distinguish between Static and Variable states and state derivatives.
2	Fixed —Specifies that the SIM Trim VI uses the value of this parameter as a condition for trimming a subsystem. The SIM Linearize VI does not distinguish between Fixed and Variable parameters.

Parameters In specifies either the **States**, **Inputs**, **Outputs**, or **State Derivatives** array that the [SIM Query Subsystem](#) VI returns.

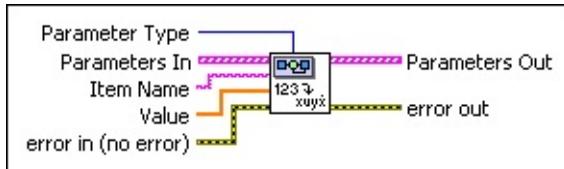
- Name** specifies the name of the parameter.
- Value** specifies the value of the parameter.
- Static Input?** is TRUE if the parameter type **Static**. **Static Input?** is FALSE if the parameter type is either **Variable** or **Fixed**.
- Vector?** is TRUE if the parameter value is a vector or an array. **Vector?** is FALSE if the parameter value is a scalar.
- Fixed Size?** is TRUE if you can replace this parameter only with a vector or an array of the same size. **Fixed Size?** is FALSE if you can replace this parameter with a vector or array of any size.
- Fixed Value?** is TRUE if the parameter value is a condition that you specify for trimming a subsystem. **Fixed Value?** is

FALSE if the parameter value is not a condition that you specify for trimming a subsystem.

-  **Item Name** specifies the name of the parameter in the **Parameters In** array that you want to change.
-  **Value** specifies the new value for the parameter in the **Parameters In** array.
-  **error in** describes error conditions that occur before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function runs, it runs normally and sets its own error status in **error out**. Use the [Simple Error Handler](#) or [General Error Handler](#) VIs to display the description of the error code. Use [exception control](#) to treat what is normally an error as no error or to treat a warning as an error. Use **error in** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in** of the next node.
 -  **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.
 -  **code** is the error or warning code. The default is 0. If **status** is TRUE, **code** is a nonzero [error code](#). If **status** is FALSE, **code** is 0 or a warning code.
 -  **source** specifies the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning. The default is an empty string.
-  **Parameters Out** returns the modified parameter values and types of parameters in the **Parameters In** array.
 -  **Name** returns the name of the parameter.
 -  **Value** returns the value of the parameter.
 -  **Static Input?** returns TRUE if the parameter type is **Static**. **Static Input?** returns FALSE if the parameter type is either **Variable** or **Fixed**.

-  **Vector?** returns TRUE if the parameter value is a vector or an array. **Vector?** returns FALSE if the parameter value is a scalar.
 -  **Fixed Size?** returns TRUE if you can replace this parameter only with a vector or an array of the same size. **Fixed Size?** returns FALSE if you can replace this parameter with a vector or array of any size.
 -  **Fixed Value?** returns TRUE if the parameter value is a condition that you specify for trimming a subsystem. **Fixed Value?** returns FALSE if the parameter value is not a condition that you specify for trimming a subsystem.
-  **error out** contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.
-  **status** is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.
 -  **code** is the error or warning code. If **status** is TRUE, **code** is a nonzero **error code**. If **status** is FALSE, **code** is 0 or a warning code.
 -  **source** describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning.

SIM Set Parameter Value (by Name, Vector)



Parameter Type specifies the type of parameter.

0	Variable (default)—Specifies that the SIM Trim VI can change the value of this parameter to satisfy any conditions you specify. The SIM Linearize VI includes Variable inputs and outputs in the LTI model.
1	Static —Specifies that the SIM Trim VI excludes this parameter from the search. The SIM Linearize VI excludes Static inputs and outputs from the LTI model. This VI does not distinguish between Static and Variable states and state derivatives.
2	Fixed —Specifies that the SIM Trim VI uses the value of this parameter as a condition for trimming a subsystem. The SIM Linearize VI does not distinguish between Fixed and Variable parameters.

Parameters In specifies either the **States**, **Inputs**, **Outputs**, or **State Derivatives** array that the [SIM Query Subsystem](#) VI returns.

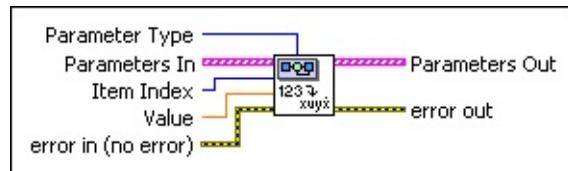
- Name** specifies the name of the parameter.
- Value** specifies the value of the parameter.
- Static Input?** is TRUE if the parameter type **Static**. **Static Input?** is FALSE if the parameter type is either **Variable** or **Fixed**.
- Vector?** is TRUE if the parameter value is a vector or an array. **Vector?** is FALSE if the parameter value is a scalar.
- Fixed Size?** is TRUE if you can replace this parameter only with a vector or an array of the same size. **Fixed Size?** is FALSE if you can replace this parameter with a vector or array of any size.
- Fixed Value?** is TRUE if the parameter value is a condition that you specify for trimming a subsystem. **Fixed Value?** is

FALSE if the parameter value is not a condition that you specify for trimming a subsystem.

-  **Item Name** specifies the name of the parameter in the **Parameters In** array that you want to change.
-  **Value** specifies the new values for the parameter in the **Parameters In** array.
-  **error in** describes error conditions that occur before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function runs, it runs normally and sets its own error status in **error out**. Use the [Simple Error Handler](#) or [General Error Handler](#) VIs to display the description of the error code. Use [exception control](#) to treat what is normally an error as no error or to treat a warning as an error. Use **error in** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in** of the next node.
 -  **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.
 -  **code** is the error or warning code. The default is 0. If **status** is TRUE, **code** is a nonzero [error code](#). If **status** is FALSE, **code** is 0 or a warning code.
 -  **source** specifies the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning. The default is an empty string.
-  **Parameters Out** returns the modified parameter values and types of parameters in the **Parameters In** array.
 -  **Name** returns the name of the parameter.
 -  **Value** returns the value of the parameter.
 -  **Static Input?** returns TRUE if the parameter type is **Static**. **Static Input?** returns FALSE if the parameter type is either **Variable** or **Fixed**.

-  **Vector?** returns TRUE if the parameter value is a vector or an array. **Vector?** returns FALSE if the parameter value is a scalar.
 -  **Fixed Size?** returns TRUE if you can replace this parameter only with a vector or an array of the same size. **Fixed Size?** returns FALSE if you can replace this parameter with a vector or array of any size.
 -  **Fixed Value?** returns TRUE if the parameter value is a condition that you specify for trimming a subsystem. **Fixed Value?** returns FALSE if the parameter value is not a condition that you specify for trimming a subsystem.
-  **error out** contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.
-  **status** is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.
 -  **code** is the error or warning code. If **status** is TRUE, **code** is a nonzero **error code**. If **status** is FALSE, **code** is 0 or a warning code.
 -  **source** describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning.

SIM Set Parameter Value (by Index, Scalar)



Parameter Type specifies the type of parameter.

0	Variable (default)—Specifies that the SIM Trim VI can change the value of this parameter to satisfy any conditions you specify. The SIM Linearize VI includes Variable inputs and outputs in the LTI model.
1	Static —Specifies that the SIM Trim VI excludes this parameter from the search. The SIM Linearize VI excludes Static inputs and outputs from the LTI model. This VI does not distinguish between Static and Variable states and state derivatives.
2	Fixed —Specifies that the SIM Trim VI uses the value of this parameter as a condition for trimming a subsystem. The SIM Linearize VI does not distinguish between Fixed and Variable parameters.

Parameters In specifies either the **States**, **Inputs**, **Outputs**, or **State Derivatives** array that the [SIM Query Subsystem](#) VI returns.

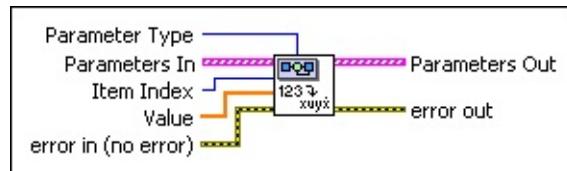
- Name** specifies the name of the parameter.
- Value** specifies the value of the parameter.
- Static Input?** is TRUE if the parameter type **Static**. **Static Input?** is FALSE if the parameter type is either **Variable** or **Fixed**.
- Vector?** is TRUE if the parameter value is a vector or an array. **Vector?** is FALSE if the parameter value is a scalar.
- Fixed Size?** is TRUE if you can replace this parameter only with a vector or an array of the same size. **Fixed Size?** is FALSE if you can replace this parameter with a vector or array of any size.
- Fixed Value?** is TRUE if the parameter value is a condition that you specify for trimming a subsystem. **Fixed Value?** is

FALSE if the parameter value is not a condition that you specify for trimming a subsystem.

-  **Item Index** specifies the index number of the parameter in the **Parameters In** array that you want to change.
-  **Value** specifies the new value for the parameter in the **Parameters In** array.
-  **error in** describes error conditions that occur before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function runs, it runs normally and sets its own error status in **error out**. Use the [Simple Error Handler](#) or [General Error Handler](#) VIs to display the description of the error code. Use [exception control](#) to treat what is normally an error as no error or to treat a warning as an error. Use **error in** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in** of the next node.
 -  **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.
 -  **code** is the error or warning code. The default is 0. If **status** is TRUE, **code** is a nonzero [error code](#). If **status** is FALSE, **code** is 0 or a warning code.
 -  **source** specifies the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning. The default is an empty string.
-  **Parameters Out** returns the modified parameter values and types of parameters in the **Parameters In** array.
 -  **Name** returns the name of the parameter.
 -  **Value** returns the value of the parameter.
 -  **Static Input?** returns TRUE if the parameter type is **Static**. **Static Input?** returns FALSE if the parameter type is either **Variable** or **Fixed**.

-  **Vector?** returns TRUE if the parameter value is a vector or an array. **Vector?** returns FALSE if the parameter value is a scalar.
 -  **Fixed Size?** returns TRUE if you can replace this parameter only with a vector or an array of the same size. **Fixed Size?** returns FALSE if you can replace this parameter with a vector or array of any size.
 -  **Fixed Value?** returns TRUE if the parameter value is a condition that you specify for trimming a subsystem. **Fixed Value?** returns FALSE if the parameter value is not a condition that you specify for trimming a subsystem.
-  **error out** contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.
-  **status** is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.
 -  **code** is the error or warning code. If **status** is TRUE, **code** is a nonzero **error code**. If **status** is FALSE, **code** is 0 or a warning code.
 -  **source** describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning.

SIM Set Parameter Value (by Index, Vector)



Parameter Type specifies the type of parameter.

0	Variable (default)—Specifies that the SIM Trim VI can change the value of this parameter to satisfy any conditions you specify. The SIM Linearize VI includes Variable inputs and outputs in the LTI model.
1	Static —Specifies that the SIM Trim VI excludes this parameter from the search. The SIM Linearize VI excludes Static inputs and outputs from the LTI model. This VI does not distinguish between Static and Variable states and state derivatives.
2	Fixed —Specifies that the SIM Trim VI uses the value of this parameter as a condition for trimming a subsystem. The SIM Linearize VI does not distinguish between Fixed and Variable parameters.

Parameters In specifies either the **States**, **Inputs**, **Outputs**, or **State Derivatives** array that the [SIM Query Subsystem](#) VI returns.

- Name** specifies the name of the parameter.
- Value** specifies the value of the parameter.
- Static Input?** is TRUE if the parameter type **Static**. **Static Input?** is FALSE if the parameter type is either **Variable** or **Fixed**.
- Vector?** is TRUE if the parameter value is a vector or an array. **Vector?** is FALSE if the parameter value is a scalar.
- Fixed Size?** is TRUE if you can replace this parameter only with a vector or an array of the same size. **Fixed Size?** is FALSE if you can replace this parameter with a vector or array of any size.
- Fixed Value?** is TRUE if the parameter value is a condition that you specify for trimming a subsystem. **Fixed Value?** is

FALSE if the parameter value is not a condition that you specify for trimming a subsystem.

-  **Item Index** specifies the index number of the parameter in the **Parameters In** array that you want to change.
-  **Value** specifies the new values for the parameter in the **Parameters In** array.
-  **error in** describes error conditions that occur before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function runs, it runs normally and sets its own error status in **error out**. Use the [Simple Error Handler](#) or [General Error Handler](#) VIs to display the description of the error code. Use [exception control](#) to treat what is normally an error as no error or to treat a warning as an error. Use **error in** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in** of the next node.
 -  **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.
 -  **code** is the error or warning code. The default is 0. If **status** is TRUE, **code** is a nonzero [error code](#). If **status** is FALSE, **code** is 0 or a warning code.
 -  **source** specifies the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning. The default is an empty string.
-  **Parameters Out** returns the modified parameter values and types of parameters in the **Parameters In** array.
 -  **Name** returns the name of the parameter.
 -  **Value** returns the value of the parameter.
 -  **Static Input?** returns TRUE if the parameter type is **Static**. **Static Input?** returns FALSE if the parameter type is either **Variable** or **Fixed**.

-  **Vector?** returns TRUE if the parameter value is a vector or an array. **Vector?** returns FALSE if the parameter value is a scalar.
 -  **Fixed Size?** returns TRUE if you can replace this parameter only with a vector or an array of the same size. **Fixed Size?** returns FALSE if you can replace this parameter with a vector or array of any size.
 -  **Fixed Value?** returns TRUE if the parameter value is a condition that you specify for trimming a subsystem. **Fixed Value?** returns FALSE if the parameter value is not a condition that you specify for trimming a subsystem.
-  **error out** contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.
-  **status** is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.
 -  **code** is the error or warning code. If **status** is TRUE, **code** is a nonzero **error code**. If **status** is FALSE, **code** is 0 or a warning code.
 -  **source** describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning.

SIM Trim VI

Owning Palette: [Trim & Linearize VIs](#)

Installed With: Control Design and Simulation Module

Calculates the states and/or inputs of a simulation subsystem necessary to produce a set of outputs and/or state derivatives that you specify. Trimming a subsystem involves searching for a set of states and/or inputs that satisfy one or more specified output and/or state derivative conditions. The data type you wire to the **Path** input determines the polymorphic instance to use.

If the trim algorithm cannot satisfy the conditions that you specified, the algorithm returns the closest values to the conditions that you specified.



Note You can use this VI to trim only a continuous subsystem.

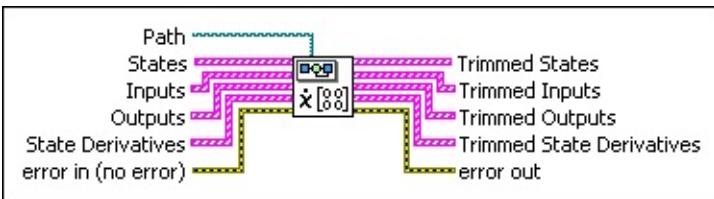
Details

Use the pull-down menu to select an instance of this VI.

Select an instance 

Place on the block diagram Find on the **Functions** palette

SIM Trim (Path)



- **Path** specifies the path to the simulation subsystem on which you want to operate.
- **States** specifies the initial states of the simulation subsystem. The [SIM Trim](#) VI uses these states to begin the search for optimal trimmed values.
 - **Name** specifies the name of the parameter.
 - **Value** specifies the value of the parameter.
 - **Static Input?** is TRUE if the parameter type **Static**. The SIM Trim VI excludes static inputs and states from the search. **Static Input?** is FALSE if the parameter type is either **Variable** or **Fixed**.
 - **Vector?** is TRUE if the parameter value is a vector or an array. **Vector?** is FALSE if the parameter value is a scalar.
 - **Fixed Size?** is TRUE if you can replace this parameter only with a vector or an array of the same size. **Fixed Size?** is FALSE if you can replace this parameter with a vector or array of any size.
 - **Fixed Value?** is TRUE if the parameter value is a condition that you specify for trimming a subsystem. **Fixed Value?** is FALSE if the parameter value is not a condition that you specify for trimming a subsystem.
- **Inputs** specifies the initial inputs of the simulation subsystem. The SIM Trim VI uses these inputs to begin the search for optimal trimmed values.
 - **Name** specifies the name of the parameter.
 - **Value** specifies the value of the parameter.
 - **Static Input?** is TRUE if the parameter type **Static**. The SIM Trim VI excludes static inputs and states from the

search. **Static Input?** is FALSE if the parameter type is either **Variable** or **Fixed**.

-  **Vector?** is TRUE if the parameter value is a vector or an array. **Vector?** is FALSE if the parameter value is a scalar.
 -  **Fixed Size?** is TRUE if you can replace this parameter only with a vector or an array of the same size. **Fixed Size?** is FALSE if you can replace this parameter with a vector or array of any size.
 -  **Fixed Value?** is TRUE if the parameter value is a condition that you specify for trimming a subsystem. **Fixed Value?** is FALSE if the parameter value is not a condition that you specify for trimming a subsystem.
-  **Outputs** specifies the output conditions that you want to satisfy when trimming a subsystem. You specify conditions and the values of these conditions using the [SIM Set Parameter Value VI](#).
-  **Name** specifies the name of the parameter.
 -  **Value** specifies the value of the parameter.
 -  **Static Input?** is TRUE if the parameter type **Static**. The SIM Trim VI excludes static inputs and states from the search. **Static Input?** is FALSE if the parameter type is either **Variable** or **Fixed**.
 -  **Vector?** is TRUE if the parameter value is a vector or an array. **Vector?** is FALSE if the parameter value is a scalar.
 -  **Fixed Size?** is TRUE if you can replace this parameter only with a vector or an array of the same size. **Fixed Size?** is FALSE if you can replace this parameter with a vector or array of any size.
 -  **Fixed Value?** is TRUE if the parameter value is a condition that you specify for trimming a subsystem. **Fixed Value?** is FALSE if the parameter value is not a condition that you specify for trimming a subsystem.

-  **State Derivatives** specifies the state derivative conditions that you want to satisfy when trimming a subsystem. You specify conditions and the values of these conditions using the [SIM Set Parameter Value VI](#).

-  **Name** specifies the name of the parameter.
 -  **Value** specifies the value of the parameter.
 -  **Static Input?** is TRUE if the parameter type **Static**. The SIM Trim VI excludes static inputs and states from the search. **Static Input?** is FALSE if the parameter type is either **Variable** or **Fixed**.
 -  **Vector?** is TRUE if the parameter value is a vector or an array. **Vector?** is FALSE if the parameter value is a scalar.
 -  **Fixed Size?** is TRUE if you can replace this parameter only with a vector or an array of the same size. **Fixed Size?** is FALSE if you can replace this parameter with a vector or array of any size.
 -  **Fixed Value?** is TRUE if the parameter value is a condition that you specify for trimming a subsystem. **Fixed Value?** is FALSE if the parameter value is not a condition that you specify for trimming a subsystem.
-  **error in** describes error conditions that occur before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function runs, it runs normally and sets its own error status in **error out**. Use the [Simple Error Handler](#) or [General Error Handler](#) VIs to display the description of the error code. Use [exception control](#) to treat what is normally an error as no error or to treat a warning as an error. Use **error in** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in** of the next node.
-  **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.
 -  **code** is the error or warning code. The default is 0. If **status** is TRUE, **code** is a nonzero [error code](#). If **status** is FALSE, **code** is 0 or a warning code.



source specifies the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning. The default is an empty string.



Trimmed States returns the state values necessary to satisfy any conditions that you specified.



Name returns the name of the parameter.



Value returns the value of the parameter.



Static Input? returns TRUE if the parameter type **Static**. The SIM Trim VI excludes **Static** inputs and outputs from the search. **Static Input?** returns FALSE if the parameter type is either **Variable** or **Fixed**.



Vector? returns TRUE if the parameter value is a vector or an array. **Vector?** returns FALSE if the parameter value is a scalar.



Fixed Size? returns TRUE if you can replace this parameter only with a vector or an array of the same size. **Fixed Size?** returns FALSE if you can replace this parameter with a vector or array of any size.



Fixed Value? returns TRUE if the parameter value is a condition that you specify for trimming a subsystem. **Fixed Value?** returns FALSE if the parameter value is not a condition that you specify for trimming a subsystem.



Trimmed Inputs returns the input values necessary to satisfy any conditions you specified.



Name returns the name of the parameter.



Value returns the value of the parameter.



Static Input? returns TRUE if the parameter type **Static**. The SIM Trim VI excludes **Static** inputs and outputs from the search. **Static Input?** returns FALSE if the parameter type is either **Variable** or **Fixed**.



Vector? returns TRUE if the parameter value is a vector or an array. **Vector?** returns FALSE if the parameter value is a scalar.

 **Fixed Size?** returns TRUE if you can replace this parameter only with a vector or an array of the same size. **Fixed Size?** returns FALSE if you can replace this parameter with a vector or array of any size.

 **Fixed Value?** returns TRUE if the parameter value is a condition that you specify for trimming a subsystem. **Fixed Value?** returns FALSE if the parameter value is not a condition that you specify for trimming a subsystem.

 **Trimmed Outputs** returns the output values that occur when you apply the **Trimmed States** and **Trimmed Inputs** parameters to the subsystem.

 **Name** returns the name of the parameter.

 **Value** returns the value of the parameter.

 **Static Input?** returns TRUE if the parameter type **Static**. The SIM Trim VI excludes **Static** inputs and outputs from the search. **Static Input?** returns FALSE if the parameter type is either **Variable** or **Fixed**.

 **Vector?** returns TRUE if the parameter value is a vector or an array. **Vector?** returns FALSE if the parameter value is a scalar.

 **Fixed Size?** returns TRUE if you can replace this parameter only with a vector or an array of the same size. **Fixed Size?** returns FALSE if you can replace this parameter with a vector or array of any size.

 **Fixed Value?** returns TRUE if the parameter value is a condition that you specify for trimming a subsystem. **Fixed Value?** returns FALSE if the parameter value is not a condition that you specify for trimming a subsystem.

 **Trimmed State Derivatives** returns the state derivative values that occur when you apply the **Trimmed States** and **Trimmed Inputs** parameters to the subsystem.

 **Name** returns the name of the parameter.

 **Value** returns the value of the parameter.

 **Static Input?** returns TRUE if the parameter type **Static**.

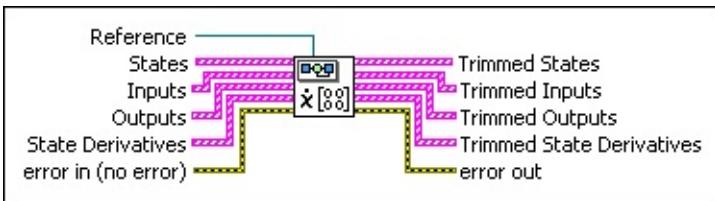
The SIM Trim VI excludes **Static** inputs and outputs from the search. **Static Input?** returns FALSE if the parameter type is either **Variable** or **Fixed**.

-  **Vector?** returns TRUE if the parameter value is a vector or an array. **Vector?** returns FALSE if the parameter value is a scalar.
-  **Fixed Size?** returns TRUE if you can replace this parameter only with a vector or an array of the same size. **Fixed Size?** returns FALSE if you can replace this parameter with a vector or array of any size.
-  **Fixed Value?** returns TRUE if the parameter value is a condition that you specify for trimming a subsystem. **Fixed Value?** returns FALSE if the parameter value is not a condition that you specify for trimming a subsystem.

 **error out** contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

-  **status** is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.
-  **code** is the error or warning code. If **status** is TRUE, **code** is a nonzero [error code](#). If **status** is FALSE, **code** is 0 or a warning code.
-  **source** describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning.

SIM Trim (Reference)



- **Reference** specifies a reference to the subsystem on which you want to operate.
- **States** specifies the initial states of the simulation subsystem. The [SIM Trim](#) VI uses these states to begin the search for optimal trimmed values.
 - **Name** specifies the name of the parameter.
 - **Value** specifies the value of the parameter.
 - **Static Input?** is TRUE if the parameter type **Static**. The SIM Trim VI excludes static inputs and states from the search. **Static Input?** is FALSE if the parameter type is either **Variable** or **Fixed**.
 - **Vector?** is TRUE if the parameter value is a vector or an array. **Vector?** is FALSE if the parameter value is a scalar.
 - **Fixed Size?** is TRUE if you can replace this parameter only with a vector or an array of the same size. **Fixed Size?** is FALSE if you can replace this parameter with a vector or array of any size.
 - **Fixed Value?** is TRUE if the parameter value is a condition that you specify for trimming a subsystem. **Fixed Value?** is FALSE if the parameter value is not a condition that you specify for trimming a subsystem.
- **Inputs** specifies the initial inputs of the simulation subsystem. The SIM Trim VI uses these inputs to begin the search for optimal trimmed values.
 - **Name** specifies the name of the parameter.
 - **Value** specifies the value of the parameter.
 - **Static Input?** is TRUE if the parameter type **Static**. The SIM Trim VI excludes static inputs and states from the

search. **Static Input?** is FALSE if the parameter type is either **Variable** or **Fixed**.

-  **Vector?** is TRUE if the parameter value is a vector or an array. **Vector?** is FALSE if the parameter value is a scalar.
 -  **Fixed Size?** is TRUE if you can replace this parameter only with a vector or an array of the same size. **Fixed Size?** is FALSE if you can replace this parameter with a vector or array of any size.
 -  **Fixed Value?** is TRUE if the parameter value is a condition that you specify for trimming a subsystem. **Fixed Value?** is FALSE if the parameter value is not a condition that you specify for trimming a subsystem.
-  **Outputs** specifies the output conditions that you want to satisfy when trimming a subsystem. You specify conditions and the values of these conditions using the [SIM Set Parameter Value VI](#).
-  **Name** specifies the name of the parameter.
 -  **Value** specifies the value of the parameter.
 -  **Static Input?** is TRUE if the parameter type **Static**. The SIM Trim VI excludes static inputs and states from the search. **Static Input?** is FALSE if the parameter type is either **Variable** or **Fixed**.
 -  **Vector?** is TRUE if the parameter value is a vector or an array. **Vector?** is FALSE if the parameter value is a scalar.
 -  **Fixed Size?** is TRUE if you can replace this parameter only with a vector or an array of the same size. **Fixed Size?** is FALSE if you can replace this parameter with a vector or array of any size.
 -  **Fixed Value?** is TRUE if the parameter value is a condition that you specify for trimming a subsystem. **Fixed Value?** is FALSE if the parameter value is not a condition that you specify for trimming a subsystem.

-  **State Derivatives** specifies the state derivative conditions that you want to satisfy when trimming a subsystem. You specify conditions and the values of these conditions using the [SIM Set Parameter Value VI](#).

-  **Name** specifies the name of the parameter.
 -  **Value** specifies the value of the parameter.
 -  **Static Input?** is TRUE if the parameter type **Static**. The SIM Trim VI excludes static inputs and states from the search. **Static Input?** is FALSE if the parameter type is either **Variable** or **Fixed**.
 -  **Vector?** is TRUE if the parameter value is a vector or an array. **Vector?** is FALSE if the parameter value is a scalar.
 -  **Fixed Size?** is TRUE if you can replace this parameter only with a vector or an array of the same size. **Fixed Size?** is FALSE if you can replace this parameter with a vector or array of any size.
 -  **Fixed Value?** is TRUE if the parameter value is a condition that you specify for trimming a subsystem. **Fixed Value?** is FALSE if the parameter value is not a condition that you specify for trimming a subsystem.
-  **error in** describes error conditions that occur before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function runs, it runs normally and sets its own error status in **error out**. Use the [Simple Error Handler](#) or [General Error Handler](#) VIs to display the description of the error code. Use [exception control](#) to treat what is normally an error as no error or to treat a warning as an error. Use **error in** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in** of the next node.
-  **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.
 -  **code** is the error or warning code. The default is 0. If **status** is TRUE, **code** is a nonzero [error code](#). If **status** is FALSE, **code** is 0 or a warning code.



source specifies the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning. The default is an empty string.



Trimmed States returns the state values necessary to satisfy any conditions that you specified.



Name returns the name of the parameter.



Value returns the value of the parameter.



Static Input? returns TRUE if the parameter type **Static**. The SIM Trim VI excludes **Static** inputs and outputs from the search. **Static Input?** returns FALSE if the parameter type is either **Variable** or **Fixed**.



Vector? returns TRUE if the parameter value is a vector or an array. **Vector?** returns FALSE if the parameter value is a scalar.



Fixed Size? returns TRUE if you can replace this parameter only with a vector or an array of the same size. **Fixed Size?** returns FALSE if you can replace this parameter with a vector or array of any size.



Fixed Value? returns TRUE if the parameter value is a condition that you specify for trimming a subsystem. **Fixed Value?** returns FALSE if the parameter value is not a condition that you specify for trimming a subsystem.



Trimmed Inputs returns the input values necessary to satisfy any conditions you specified.



Name returns the name of the parameter.



Value returns the value of the parameter.



Static Input? returns TRUE if the parameter type **Static**. The SIM Trim VI excludes **Static** inputs and outputs from the search. **Static Input?** returns FALSE if the parameter type is either **Variable** or **Fixed**.



Vector? returns TRUE if the parameter value is a vector or an array. **Vector?** returns FALSE if the parameter value is a scalar.

 **Fixed Size?** returns TRUE if you can replace this parameter only with a vector or an array of the same size. **Fixed Size?** returns FALSE if you can replace this parameter with a vector or array of any size.

 **Fixed Value?** returns TRUE if the parameter value is a condition that you specify for trimming a subsystem. **Fixed Value?** returns FALSE if the parameter value is not a condition that you specify for trimming a subsystem.

 **Trimmed Outputs** returns the output values that occur when you apply the **Trimmed States** and **Trimmed Inputs** parameters to the subsystem.

 **Name** returns the name of the parameter.

 **Value** returns the value of the parameter.

 **Static Input?** returns TRUE if the parameter type **Static**. The SIM Trim VI excludes **Static** inputs and outputs from the search. **Static Input?** returns FALSE if the parameter type is either **Variable** or **Fixed**.

 **Vector?** returns TRUE if the parameter value is a vector or an array. **Vector?** returns FALSE if the parameter value is a scalar.

 **Fixed Size?** returns TRUE if you can replace this parameter only with a vector or an array of the same size. **Fixed Size?** returns FALSE if you can replace this parameter with a vector or array of any size.

 **Fixed Value?** returns TRUE if the parameter value is a condition that you specify for trimming a subsystem. **Fixed Value?** returns FALSE if the parameter value is not a condition that you specify for trimming a subsystem.

 **Trimmed State Derivatives** returns the state derivative values that occur when you apply the **Trimmed States** and **Trimmed Inputs** parameters to the subsystem.

 **Name** returns the name of the parameter.

 **Value** returns the value of the parameter.

 **Static Input?** returns TRUE if the parameter type **Static**.

The SIM Trim VI excludes **Static** inputs and outputs from the search. **Static Input?** returns FALSE if the parameter type is either **Variable** or **Fixed**.

-  **Vector?** returns TRUE if the parameter value is a vector or an array. **Vector?** returns FALSE if the parameter value is a scalar.
-  **Fixed Size?** returns TRUE if you can replace this parameter only with a vector or an array of the same size. **Fixed Size?** returns FALSE if you can replace this parameter with a vector or array of any size.
-  **Fixed Value?** returns TRUE if the parameter value is a condition that you specify for trimming a subsystem. **Fixed Value?** returns FALSE if the parameter value is not a condition that you specify for trimming a subsystem.

 **error out** contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

-  **status** is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.
-  **code** is the error or warning code. If **status** is TRUE, **code** is a nonzero [error code](#). If **status** is FALSE, **code** is 0 or a warning code.
-  **source** describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning.

SIM Trim Details

You specify the inputs to this VI as a set of **State Derivatives** dx_0 and an initial search point that consists of **Inputs** u_0 , **States** x_0 , and **Outputs** y_0 . This VI varies the values of the initial search point to minimize the maximum value resulting from the absolute value of $(u-u_0)$, $(x-x_0)$, and $(y-y_0)$, such that the absolute value of $(dx-dx_0)$ equals 0.

This VI then returns the result of this search as a set of **Trimmed States**, **Trimmed Inputs**, **Trimmed Outputs**, and **Trimmed State Derivatives** that approximates the optimal solution. If this VI cannot find a point such that the absolute value of $(dx-dx_0) = 0$, this VI returns a point where the absolute value of $(dx-dx_0)$ is as small as possible.

Utilities Functions

Owning Palette: [Simulation VIs and Functions](#)

Installed With: Control Design and Simulation Module. This topic might not match its corresponding palette in LabVIEW depending on your operating system, licensed product(s), and target.

Use the Utilities functions to perform various tasks such as creating a history of signal values and the times at which they were recorded, indexing a signal by simulation time, reporting simulation time, reporting simulation errors, stopping the simulation programmatically, and returning the simulation parameters.

The functions on this palette can return [general LabVIEW error codes](#) or [specific Simulation error codes](#). If you use the functions on this palette in a [Simulation Loop](#), LabVIEW sends any errors that these functions return to the **Error** output on the Output Node of the Simulation Loop.

Palette Object	Description
Collector	Collects a signal at each time step of the simulation and returns a history of the signal value and the time at which this function recorded each value in the history.
Get Simulation Parameters	Returns a cluster containing the current simulation parameters. You set these parameters using the Configure Simulation Parameters dialog box and/or the Input Node of the Simulation Loop . This function does not have a configuration dialog box.
Halt Simulation	Stops the simulation at the end of the current time step, if the value of the Halt? parameter is TRUE. This function does not have a configuration dialog box.
Indexer	Indexes a waveform or array by the current simulation time.
Memory	Stores the value of the Input signal from the previous iteration of the simulation. Use this polymorphic function to transfer values from one iteration of the Simulation Loop to the next. The data type you wire to the Initial Value input determines the polymorphic instance to use.
Report	Reports an error to the top-level Simulation Loop . This

<u>Simulation Error</u>	function does not have a configuration dialog box.
<u>Simulation Time</u>	Returns the current simulation time. This value is useful if you terminate the simulation before the final time and you want to know the time at which the simulation terminated. This value also is useful if you want to generate a signal based upon the simulation time without using the <u>Signal Generation</u> functions. This function does not have a configuration dialog box.

Collector Function

Owning Palette: [Utilities Functions](#)

Installed With: Control Design and Simulation Module

Collects a signal at each time step of the simulation and returns a history of the signal value and the time at which this function recorded each value in the history.

[Details](#) [Example](#)

[Dialog Box Options](#)

[Block Diagram Inputs](#)

[Block Diagram Outputs](#)

■ Place on the block diagram ■ Find on the **Functions** palette

Dialog Box Options

Parameter	Description
Polymorphic instance	Specifies whether this function is Scalar or Vector . The default value is Scalar .
Parameters	Lists all the parameters associated with this function. Select a parameter from this list to configure the parameter. When you select a parameter, the parameter and its associated Parameter source control appear in the Parameter Information section of the configuration dialog box.
Preview	Displays a graphical preview, if available, of the function output or configuration.
Parameter Information	Contains the parameters you can configure for this function. You must select a parameter from the Parameters list to make that parameter and its associated Parameter source control visible in the Parameter Information section of the configuration dialog box.
Parameter source	Specifies whether you configure this parameter using the Configuration Dialog Box or a Terminal on the simulation diagram. The default value is Configuration Dialog Box . If you select Terminal , LabVIEW displays an input for that parameter on the simulation diagram, and you can wire values to that input to configure this function programmatically. If you select Configuration Dialog Box , LabVIEW removes that input from the simulation diagram. You then must set the value for this parameter inside the configuration dialog box.
history length	Specifies the time interval of the history to store, in seconds. If t is the current simulation time, this function stores the signal values for the time interval from $t - \text{history length}$ to t . The default value is Inf, which stores the entire history.
decimation	Specifies the value the function uses to determine when to record a value. If you set the value of decimation to 1,

	this function records a value every time step. If you set the value of decimation to n , where $n > 1$, this function plots a value once every n time steps. The default value is 1.
execution type	Specifies when this function updates. You can choose from the following options: <ul style="list-style-type: none"> • Continuous • Continuous(include minor time steps) • Discrete
sample period (sec)	Specifies the length of the discrete time step, in seconds, of this function. If you enter a value of –1, this function inherits the discrete time step you specify for the simulation diagram. Otherwise, the value of sample period (sec) must be a multiple of the discrete time step you specify for the simulation diagram. The default value of sample period (sec) is 1.
sample skew (sec)	Specifies the length of time by which you want to delay the execution of this function. The default value is 0. The value of this parameter must satisfy the following relationship: $0 \leq \text{sample skew (sec)} \leq \text{sample period (sec)}$

Block Diagram Inputs

Parameter	Description
history length	Specifies the time interval of the history to store, in seconds. If t is the current simulation time, this function stores the signal values for the time interval from $t - \text{history length}$ to t . The default value is Inf, which stores the entire history.
decimation	Specifies the value the function uses to determine when to record a value. If you set the value of decimation to 1, this function records a value every time step. If you set the value of decimation to n , where $n > 1$, this function plots a value once every n time steps. The default value is 1.
execution type	Specifies when this function updates. You can choose from the following options: <ul style="list-style-type: none"> • Continuous • Continuous(include minor time steps) • Discrete
sample period (sec)	Specifies the length of the discrete time step, in seconds, of this function. If you enter a value of -1, this function inherits the discrete time step you specify for the simulation diagram. Otherwise, the value of sample period (sec) must be a multiple of the discrete time step you specify for the simulation diagram. The default value of sample period (sec) is 1.
sample skew (sec)	Specifies the length of time by which you want to delay the execution of this function. The default value is 0. The value of this parameter must satisfy the following relationship: $0 \leq \text{sample skew (sec)} \leq \text{sample period (sec)}$
input signal	Specifies the signal from which to collect a history.

Block Diagram Outputs

Parameter	Description
signal history	Returns the history of the signal value and the time at which this function recorded each value in the history as a cluster of time-signal arrays.

Collector Details

This function operates like an [auto-indexing](#) output tunnel of a [While Loop](#) or [For Loop](#) but ensures that the simulation data is correlated to the simulation time correctly.

Feedthrough Behavior

All input/output pairs of this function have [direct feedthrough behavior](#).

Example

Refer to the collector VI in the labview\examples\Control and Simulation\Simulation\Utilities directory for an example of using the Collector VI.

- Open example ■ Browse related examples

Get Simulation Parameters Function

Owning Palette: [Utilities Functions](#)

Installed With: Control Design and Simulation Module

Returns a cluster containing the current simulation parameters. You set these parameters using the [Configure Simulation Parameters](#) dialog box and/or the Input Node of the [Simulation Loop](#). This function does not have a configuration dialog box.

[Example](#)

[Block Diagram Outputs](#)

- Place on the block diagram ■ Find on the **Functions** palette

Block Diagram Outputs

Parameter	Description
Simulation Parameters	<p>Returns the values of the following simulation parameters:</p> <ul style="list-style-type: none">• Method Order—Returns the order of the ordinary differential equation (ODE) solver.• Initial Time—Returns the time at which the ODE solver starts.• Final Time—Returns the time at which the ODE solver stops.• Step Size (s)—Returns the interval between the times at which the ODE solver evaluates the model and updates the model output, in seconds. This option is available only if you select a fixed step-size ODE solver.• Absolute Tolerance—Returns the absolute tolerance that defines the acceptable error of the ODE solver $a(y)$. The LabVIEW Control Design and Simulation Module compares $a(y)$ with the estimated error of the ODE solver $e(y)$ and adjusts the step size of the ODE solver accordingly. This option is available only if you select a variable step-size ODE solver. <p>Absolute Tolerance defines the acceptable error $a(y)$ as follows:</p> <p>For $y' = f(y)$, $a(y) \approx y * \text{Relative Tolerance} + \text{Absolute Tolerance}$.</p> <p>During each iteration of the Simulation Loop, the Control Design and Simulation Module estimates the error of the ODE solver as $e(y)$. If $e(y) > a(y)$, the variable step-size solver reduces the step size such that the step size $\geq \text{Minimum Step Size (s)}$. If $e(y) < a(y)$, the variable step-size solver increases the step size such that the step size \leq</p>

Maximum Step Size (s).

In general, a larger **Absolute Tolerance** increases the step size. This increase reduces the time the simulation needs to complete. **Absolute Tolerance** has a larger effect on $a(y)$ when y is small.

- **Relative Tolerance**—Returns the relative tolerance that defines the acceptable error of the ODE solver $a(y)$. The Control Design and Simulation Module compares $a(y)$ with the estimated error of the ODE solver $e(y)$ and adjusts the step size of the ODE solver accordingly. This option is available only if you select a variable step-size solver. **Relative Tolerance** defines the acceptable error $a(y)$ as follows:

For $y' = f(y)$, $a(y) \approx |y| * \text{Relative Tolerance} + \text{Absolute Tolerance}$.

During each iteration of the Simulation Loop, the Control Design and Simulation Module estimates the error of the ODE solver as $e(y)$. If $e(y) > a(y)$, the variable step-size solver reduces the step size such that the step size $\geq \text{Minimum Step Size (s)}$. If $e(y) < a(y)$, the variable step-size solver increases the step size such that the step size $\leq \text{Maximum Step Size (s)}$.

In general, a larger **Relative Tolerance** increases the step size. This increase reduces the time the simulation needs to complete. **Relative Tolerance** has a larger effect on $a(y)$ when y is large.

- **Solver Method**—Returns the type of [ODE solver](#) used to evaluate the simulation diagram.
- **Discrete Step Size (s)**—Returns the base time step used for discrete functions on the simulation diagram, in seconds.
- **Minimum Step Size (s)**—Returns the smallest

time step size the ODE solver can use to evaluate the simulation diagram.

- **Maximum Step Size (s)**—Returns the largest time step size the ODE solver can use to evaluate the simulation diagram.
- **Initial Step Size (s)**—Returns the time step size for the first time step of the simulation diagram evaluation.
- **Simulation Time**—Returns the current time of the simulation. This value is useful if you terminate the simulation before the final time and you need to know the time at which the simulation terminated.
- **Finished Late?**—Returns TRUE if the Simulation Loop iteration did not complete within the period you specified in the **Configure Simulation Parameters** dialog box. **Finished Late?** returns FALSE if the Simulation Loop iteration completed within the **Period** you specified.
- **Timestep Index**—Returns the number of time steps the ODE solver has taken.

Example

Refer to the get simulation parameters VI in the labview\examples\Control and Simulation\Simulation\Utilities directory for an example of using the Get Simulation Parameters VI.

- [Open example](#)
- [Browse related examples](#)

Halt Simulation Function

Owning Palette: [Utilities Functions](#)

Installed With: Control Design and Simulation Module

Stops the simulation at the end of the current time step, if the value of the **Halt?** parameter is TRUE. This function does not have a configuration dialog box.



Note You can place this function in a simulation subsystem to stop the execution of the parent simulation diagram.

[Block Diagram Inputs](#)

- Place on the block diagram ■ Find on the **Functions** palette

Block Diagram Inputs

Parameter	Description
Halt?	Specifies whether you want to stop the simulation at the end of the current time step. Specify a value of TRUE to stop the simulation. Specify a value of FALSE if you do not want to stop the simulation. The default value is FALSE.

Memory Function

Owning Palette: [Utilities Functions](#)

Installed With: Control Design and Simulation Module

Stores the value of the **Input** signal from the previous iteration of the simulation. Use this polymorphic function to transfer values from one iteration of the [Simulation Loop](#) to the next. The data type you wire to the **Initial Value** input determines the polymorphic instance to use.

[Details](#) [Example](#)

[Dialog Box Options](#)

[Block Diagram Inputs](#)

[Block Diagram Outputs](#)

■ Place on the block diagram ■ Find on the **Functions** palette

Dialog Box Options

Parameter	Description
Parameters	Lists all the parameters associated with this function. Select a parameter from this list to configure the parameter. When you select a parameter, the parameter and its associated Parameter source control appear in the Parameter Information section of the configuration dialog box.
Preview	Displays a graphical preview, if available, of the function output or configuration.
Parameter Information	Contains the parameters you can configure for this function. You must select a parameter from the Parameters list to make that parameter and its associated Parameter source control visible in the Parameter Information section of the configuration dialog box.
Parameter source	Specifies whether you configure this parameter using the Configuration Dialog Box or a Terminal on the simulation diagram. The default value is Configuration Dialog Box . If you select Terminal , LabVIEW displays an input for that parameter on the simulation diagram, and you can wire values to that input to configure this function programmatically. If you select Configuration Dialog Box , LabVIEW removes that input from the simulation diagram. You then must set the value for this parameter inside the configuration dialog box.
execution type	Specifies when this function updates. You can choose from the following options: <ul style="list-style-type: none">• Continuous• Continuous(include minor time steps)• Discrete
sample period (sec)	Specifies the length of the discrete time step, in seconds, of this function. If you enter a value of –1, this function inherits the discrete time step you specify for the simulation diagram. Otherwise, the value of sample period (sec) must be a multiple of the discrete time step you specify for

	the simulation diagram. The default value of sample period (sec) is 1.
sample skew (sec)	Specifies the length of time by which you want to delay the execution of this function. The default value is 0. The value of this parameter must satisfy the following relationship: $0 \leq \text{sample skew (sec)} \leq \text{sample period (sec)}$

Block Diagram Inputs

Parameter	Description
execution type	Specifies when this function updates. You can choose from the following options: <ul style="list-style-type: none">• Continuous• Continuous(include minor time steps)• Discrete
sample period (sec)	Specifies the length of the discrete time step, in seconds, of this function. If you enter a value of –1, this function inherits the discrete time step you specify for the simulation diagram. Otherwise, the value of sample period (sec) must be a multiple of the discrete time step you specify for the simulation diagram. The default value of sample period (sec) is 1.
sample skew (sec)	Specifies the length of time by which you want to delay the execution of this function. The default value is 0. The value of this parameter must satisfy the following relationship: $0 \leq \text{sample skew (sec)} \leq \text{sample period (sec)}$
input	Specifies the signal to which you want to apply the function.
initial value	Specifies the value at the first iteration of the Simulation Loop.

Block Diagram Outputs

Parameter	Description
output	Returns the value of the input signal from the previous iteration of the Simulation Loop. On the first iteration of the Simulation Loop, output returns the value of the initial value parameter.

Memory Details

Feedthrough Behavior

For this function, the **input** input has [indirect feedthrough](#) to the **output** output. All other input/output pairs have direct feedthrough behavior.

Example

Refer to the memory VI in the labview\examples\Control and Simulation\Simulation\Utilities directory for an example of using the Memory VI.

- Open example ■ Browse related examples

Report Simulation Error Function

Owning Palette: [Utilities Functions](#)

Installed With: Control Design and Simulation Module

Reports an error to the top-level [Simulation Loop](#). This function does not have a configuration dialog box.

[Details](#) [Example](#)

[Block Diagram Inputs](#)

■ Place on the block diagram ■ Find on the **Functions** palette

Block Diagram Inputs

Parameter	Description
error in	<p>Describes the error condition to report to the owning simulation diagram. The default is no error.</p> <ul style="list-style-type: none">• status—Displays TRUE (X) if an error occurred and is reported to the owning simulation diagram. If status is FALSE (checkmark), no error is reported to the owning simulation diagram. The default is FALSE.• code—Specifies the error or warning code. The default is 0.• source—Specifies the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning. The default is an empty string.

Report Simulation Error Details

If the simulation receives an error condition, the simulation halts after the current time step. However, because this function reports the error immediately and because most functions do not run if an error has been reported, the simulation does not run as expected between the time the function reports the error and the end of the current time step.

Example

Refer to the report simulation error VI in the labview\examples\Control and Simulation\Simulation\Utilities directory for an example of using the Report Simulation Error VI.

- [Open example](#)
- [Browse related examples](#)