BridgeVIEW™ and LabVIEW™

Control and Simulation Software for G Reference Manual

NATIONAL INSTRUMENTS™
The Software is the Instrument™

February 1998 Edition
Part Number 321878A-01

Internet Support
E-mail: `support@natinst.com`
FTP Site: `ftp.natinst.com`
Web Address: `http://www.natinst.com`

Bulletin Board Support
BBS United States: 512 794 5422
BBS United Kingdom: 01635 551422
BBS France: 01 48 65 15 59

Fax-on-Demand Support
512 418 1111

Telephone Support (USA)
Tel: 512 795 8248
Fax: 512 794 5678

International Offices
Australia 03 9879 5166, Austria 0662 45 79 90 0, Belgium 02 757 00 20, Brazil 011 288 3336,
Canada (Ontario) 905 785 0085, Canada (Québec) 514 694 8521, Denmark 45 76 26 00, Finland 09 725 725 11,
France 01 48 14 24 24, Germany 089 741 31 30, Hong Kong 2645 3186, Israel 03 6120092, Italy 02 413091,
Japan 03 5472 2970, Korea 02 596 7456, Mexico 5 520 2635, Netherlands 0348 433466, Norway 32 84 84 00,
Singapore 2265886, Spain 91 640 0085, Sweden 08 730 49 70, Switzerland 056 200 51 51, Taiwan 02 377 1200,
United Kingdom 01635 523545

National Instruments Corporate Headquarters
6504 Bridge Point Parkway Austin, Texas 78730-5039 USA Tel: 512 794 0100

© Copyright 1998 National Instruments Corporation. All rights reserved.

Important Information

Warranty

The media on which you receive National Instruments software are warranted not to fail to execute programming instructions, due to defects in materials and workmanship, for a period of 90 days from date of shipment, as evidenced by receipts or other documentation. National Instruments will, at its option, repair or replace software media that do not execute programming instructions if National Instruments receives notice of such defects during the warranty period. National Instruments does not warrant that the operation of the software shall be uninterrupted or error free.

A Return Material Authorization (RMA) number must be obtained from the factory and clearly marked on the outside of the package before any equipment will be accepted for warranty work. National Instruments will pay the shipping costs of returning to the owner parts which are covered by warranty.

National Instruments believes that the information in this manual is accurate. The document has been carefully reviewed for technical accuracy. In the event that technical or typographical errors exist, National Instruments reserves the right to make changes to subsequent editions of this document without prior notice to holders of this edition. The reader should consult National Instruments if errors are suspected. In no event shall National Instruments be liable for any damages arising out of or related to this document or the information contained in it.

EXCEPT AS SPECIFIED HEREIN, NATIONAL INSTRUMENTS MAKES NO WARRANTIES, EXPRESS OR IMPLIED, AND SPECIFICALLY DISCLAIMS ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. CUSTOMER'S RIGHT TO RECOVER DAMAGES CAUSED BY FAULT OR NEGLIGENCE ON THE PART OF NATIONAL INSTRUMENTS SHALL BE LIMITED TO THE AMOUNT THERETOFORE PAID BY THE CUSTOMER. NATIONAL INSTRUMENTS WILL NOT BE LIABLE FOR DAMAGES RESULTING FROM LOSS OF DATA, PROFITS, USE OF PRODUCTS, OR INCIDENTAL OR CONSEQUENTIAL DAMAGES, EVEN IF ADVISED OF THE POSSIBILITY THEREOF. This limitation of the liability of National Instruments will apply regardless of the form of action, whether in contract or tort, including negligence. Any action against National Instruments must be brought within one year after the cause of action accrues. National Instruments shall not be liable for any delay in performance due to causes beyond its reasonable control. The warranty provided herein does not cover damages, defects, malfunctions, or service failures caused by owner's failure to follow the National Instruments installation, operation, or maintenance instructions; owner's modification of the product; owner's abuse, misuse, or negligent acts; and power failure or surges, fire, flood, accident, actions of third parties, or other events outside reasonable control.

Copyright

Under the copyright laws, this publication may not be reproduced or transmitted in any form, electronic or mechanical, including photocopying, recording, storing in an information retrieval system, or translating, in whole or in part, without the prior written consent of National Instruments Corporation.

Trademarks

BridgeVIEW™, LabVIEW™, and NI-DAQ™ are trademarks of National Instruments Corporation.

Product and company names listed are trademarks or trade names of their respective companies.

WARNING REGARDING MEDICAL AND CLINICAL USE OF NATIONAL INSTRUMENTS PRODUCTS

National Instruments products are not designed with components and testing intended to ensure a level of reliability suitable for use in treatment and diagnosis of humans. Applications of National Instruments products involving medical or clinical treatment can create a potential for accidental injury caused by product failure, or by errors on the part of the user or application designer. Any use or application of National Instruments products for or involving medical or clinical treatment must be performed by properly trained and qualified medical personnel, and all traditional medical safeguards, equipment, and procedures that are appropriate in the particular situation to prevent serious injury or death should always continue to be used when National Instruments products are being used. National Instruments products are NOT intended to be a substitute for any form of established process, procedure, or equipment used to monitor or safeguard human health and safety in medical or clinical treatment.

Contents

About This Manual

Organization of This Manual xi
Conventions Used in This Manual xii
Related Documentation xiii
Customer Communication xiii

Chapter 1
Overview

Introduction to Control and Simulation Software for G 1-1
GSIM Features 1-2
Control and Simulation Problems 1-3
Control and Simulation Applications 1-5
Control and Simulation Libraries and VIs 1-6
Installation Procedure 1-9
    Windows 95/NT 1-9
    Windows 3.*x* 1-10
    Power Macintosh 1-10
Getting Help 1-10

Chapter 2
Control and Simulation Problems

Example: House Temperature Control 2-1
Example: PID Controllers in GSIM 2-3
A Step-By-Step Approach to GSIM Programs 2-4
    Step 1: Generate the General GSIM Frame 2-4
    Step 2: Create the Feedback Structure 2-5
    Step 3: Adjust the User Interface 2-6
    Step 4: Modify the Program to Solve First-Order Differential Equations 2-8
    Step 5: Study Other GSIM Programs 2-9
Some Restrictions and Hints 2-10
    Guideline 1: Do Not Change Execution Options on Reentrant VIs 2-10
    Guideline 2: Include GSim Initialize and GSim Manager 2-10
    Guideline 3: Run Only One GSIM Program at a Time 2-10
    Guideline 4: Hide Control/Indicator Feedback Structures 2-10
    Guideline 5: Evaluate Results Carefully 2-10
    Guideline 6: Position Controls Carefully on the Diagram 2-11

Chapter 3
Basic VIs

GSim Series 3-2
GSim Parallel 3-4
GSim Feedback 3-6
GSim Compute Zeros 3-8
GSim Compute Poles 3-9
GSim Compute Parameters 3-10
GSim Normalize 3-12

Chapter 4
Conversion VIs

State-Space (SS) Representation 4-1
Modal-Form (MF) Representation 4-2
Transfer-Function (TF) Representation 4-2
Zero-Pole (ZP) Representation 4-2
Residual-Pole (RP) Representation 4-3
GSim SS2RP 4-3
GSim SS2TF 4-5
GSim SS2ZP 4-7
GSim TF2SS 4-9
GSim ZP2TF 4-10
GSim Modal Form 4-11

Chapter 5
Feedback VIs

GSim Design from SS 5-3
GSim Design from TF 5-4
GSim Design from ZP 5-5

Chapter 6
Frequency Response VIs

GSim Bode Plot from SS 6-2
GSim Bode Plot from TF 6-3
GSim Root Locus 6-4
GSim Root Locus Plot Utility 6-5
GSim Nyquist Plot 6-6

Chapter 7
Connections

GSim Initial Condition 7-1
GSim Input Selector 7-2
GSim Trigger 7-3

Chapter 8
Discrete Systems

GSim Discrete Filter 8-1
GSim Discrete State-Space 8-2
GSim Discrete Transfer Function 8-4
GSim Discrete Zero-Pole 8-5
GSim Discrete-Time Integrator 8-7
GSim First-Order Hold 8-8
GSim Unit Delay 8-10
GSim Zero-Order Hold 8-11

Chapter 9
Linear Systems

GSim Derivative 9-1
GSim Integrator 9-2
GSim Limited Integrator 9-3
GSim PID Parallel 9-4
GSim PID Serial 9-5
GSim State-Space 9-6
GSim Transfer Function IC 9-8
GSim Transfer Function 9-9
GSim Zero-Pole 9-11

Chapter 10
Nonlinear Systems

GSim Dead Zone 10-1
GSim Friction 10-3
GSim Memory 10-4
GSim Quantizer 10-5
GSim Rate Limiter 10-6
GSim Relay 10-7
GSim Saturation 10-8

GSim Switch 10-9
GSim Transport Delay 10-10
GSim Zero Crossing 10-11

Chapter 11
Shared VIs

GSim Initialize 11-1
GSim Manager 11-2
GSim Synchronizer 11-4
GSim Timer 11-5

Chapter 12
Sinks

GSim Data Out 12-1
GSim Scope 1D 12-2
GSim Stop 12-3

Chapter 13
Data Sources

GSim Chirp Signal 13-1
GSim Collect Data In 13-2
GSim Data In 13-3
GSim Pulse Generator 13-4
GSim Ramp 13-5
GSim Realtime Clock 13-6
GSim Signal Generator 13-7
GSim Simulation Clock 13-8
GSim Sine Wave 13-9
GSim Step 13-10

Appendix A
Error Codes

Appendix B
Description of the Betas and Alphas Parameters

Appendix C
Customer Communication

Glossary

Index

Figures

Figure 1-1. Differential Equations (Simulations) Realized as Feedback Structures 1-3
Figure 1-2. Organization of Control Systems Where the Process Provides Feedback 1-4

Figure 2-1. House Temperature Control Example VI Front Panel 2-2
Figure 2-2. House Temperature Control Example VI Diagram 2-2
Figure 2-3. PID Controller in GSIM Notation 2-3

Figure 3-1. Two Systems, System 1 and System 2, in Series and Parallel Connection 3-1

Figure 5-1. Closed-Loop Feedback Structure 5-1

Figure 8-1. Zero-Order Sample and Hold of a Sine Signal 8-8
Figure 8-2. First-Order Sample and Hold of a Sine Signal 8-9

Figure 9-1. A Typical Integration Process with GSim Integrator 9-2

Figure 10-1. Output of the GSim Dead Zone VI 10-2
Figure 10-2. Output of the GSim Friction VI 10-3
Figure 10-3. Output of the GSim Relay VI 10-7
Figure 10-4. Output of the GSim Saturation VI 10-8

Figure 11-1. Include GSim Initializer in All Control and Simulation Tasks 11-2
Figure 11-2. GSim Manager VI Is Essential to All Simulations 11-3
Figure 11-3. GSim Synchronizer in the Zeroth Frame 11-4

Figure B-1. An Example of H(s) in a 1D-Array Input on the Front Panel B-1
Figure B-2. Another Example of H(s) in a 1D-Array Input on the Front Panel B-2

Tables

Table 1-1. Control and Simulation Libraries and VIs 1-6

Table A-1. Error Codes A-1

About This Manual

The *Control and Simulation Software for G Reference Manual* contains descriptions of LabVIEW and BridgeVIEW virtual instruments (VIs) for control and simulation. In addition, this manual contains examples of typical applications and programming details.

Organization of This Manual

The *Control and Simulation Software for G Reference Manual* is organized as follows:

- Chapter 1, *Overview*, introduces the Control and Simulation Software for G; describes features, applications, and contents; and provides installation instructions.
- Chapter 2, *Control and Simulation Problems*, presents control and simulation examples, prescribes a step-by-step approach, and offers some restrictions and hints for developing control and simulation programs.
- Chapter 3, *Basic VIs*, describes the VIs that perform basic operations required for the analysis and design of linear, time-invariant systems.
- Chapter 4, *Conversion VIs*, explains different forms of system representation and the VIs that you can use to convert one form to another.
- Chapter 5, *Feedback VIs*, describes VIs you can use to design linear state feedback systems from state-space, transfer-function, and residual-pole representations.
- Chapter 6, *Frequency Response VIs*, describes VIs you can use for the frequency domain analysis of linear, time-invariant control systems.
- Chapter 7, *Connections*, describes the VIs that allow multiplexing, realize triggers, and produce initial values explicitly.
- Chapter 8, *Discrete Systems*, describes the routines in GSIM that handle discrete-time linear systems.
- Chapter 9, *Linear Systems*, describes the routines that handle linear continuous systems, which can be represented as state-space, transfer-function, or zero-pole representations and implemented with three different integration methods: Euler, Adams, or Runge-Kutta.
- Chapter 10, *Nonlinear Systems*, describes the nonlinear elements you can combine with other routines to build complex nonlinear systems.

- Chapter 11, *Shared VIs*, describes the VIs that form the backbone of all control and simulation tasks based on this control and simulation software. These VIs organize and control the data flow of all calculations.
- Chapter 12, *Sinks*, describes the three VIs in the `SINKS.llb` library.
- Chapter 13, *Data Sources*, describes the signal-producing sources in the `SOURCES.llb` library. You can use these VIs as inputs for other routines in GSIM.
- Appendix A, *Error Codes*, lists and describes error codes returned by the control and simulation VIs.
- Appendix B, *Description of the Betas and Alphas Parameters*, provides a general description of the Betas and Alphas parameters. You should understand how these parameters work before implementing them in functions.
- Appendix C, *Customer Communication*, contains forms to help you gather the information necessary to help us solve your technical problems and a form you can use to comment on the product documentation.
- The *Glossary* contains an alphabetical list and description of terms used in this manual, including abbreviations, acronyms, metric prefixes, mnemonics, and symbols.
- The *Index* contains an alphabetical list of key terms and topics in this manual, including the page where you can find each one.

Conventions Used in This Manual

The following conventions are used in this manual:

☞	This icon to the left of bold italicized text denotes a note, which alerts you to important information.
bold	Bold text denotes the names of parameters and one- and two-dimensional arrays. One-dimensional arrays appear in lowercase, and two-dimensional arrays appear in uppercase.
bold italic	Bold italic text denotes a note.
italic	Italic text denotes variables, emphasis, or a cross reference.
`monospace`	Text in this font denotes text or characters that you should literally enter from the keyboard and examples. This font is also used for the proper

names of programs, subprograms, subroutines, device names, functions, operations, variables, and filenames and extensions.

Related Documentation

The following documents contain information you might find helpful as you read this manual:

- *LabVIEW User Manual*
- *BridgeVIEW User Manual*
- *G Programming Reference Manual*
- *LabVIEW QuickStart Guide*
- *LabVIEW Function and VI Reference Manual*
- *LabVIEW Data Acquisition Basics Manual*

Customer Communication

National Instruments wants to receive your comments on our products and manuals. We are interested in the applications you develop with our products, and we want to help if you have problems with them. To make it easy for you to contact us, this manual contains comment and configuration forms for you to complete. These forms are in Appendix C, *Customer Communication*, at the end of this manual.

1

Overview

This chapter introduces the Control and Simulation Software for G; describes features, applications, and contents; and provides installation instructions.

Introduction to Control and Simulation Software for G

The Control and Simulation Software for G (GSIM) provides routines for modeling, analyzing, and simulating dynamic systems. This software handles both linear and nonlinear, continuous and discrete time systems. GSIM supports the combinations of these two time models and handles real-world control problems, simulation tasks, and a combination of both. This support is useful while developing a control system that is partly simulated and partly realized.

Graphical objects such as LabVIEW diagrams completely define systems. SubVIs with well-defined connectors provide special elements such as transfer functions, relays, PID controllers, or signal generators. Because the end user can alter most necessary parameters during runtime, you have a lot of flexibility when designing systems.

You can use all elements of the LabVIEW user interface and all existing LabVIEW VIs from the full development system and toolkits with GSIM. With complete compatibility, you can add the functionality of G Math to define input signals in formulas. You can use special transformations to analyze GSIM results (for example, wavelets when you use the LabVIEW Wavelet and Filter Bank Design Toolkit).

All GSIM models are hierarchal. Although you usually do not need to construct deeper structures, you can build more complicated models or control systems in hierarchal structures. Most of the predefined GSIM VIs are written in G, so you can open and investigate them any time. Because they are reentrant, you can use the same VI multiple times in a single solution, regardless of its position in the hierarchy.

You usually can complete a GSIM project in four steps. Depending on the problem, you might be able to combine some steps.

1. Define and program the control system or simulation problem.
2. Choose parameters and initial values.
3. Execute the solution.
4. Process the results (for example, visualization or further investigation).

GSIM Features

The following list enumerates the main features of GSIM.

- Simulates and offers real-time control of linear and nonlinear systems
- Connects directly to data acquisition (DAQ) boards
- Displays solutions in graphical form
- Supports many predefined special control routines and elements (for example, PID, relay, and filter)
- Allows direct user actions on the front panel during runtime
- Handles both continuous and discrete problems
- Supports LabVIEW and offers complete G compatibility (that is, you can combine GSIM with other LabVIEW VIs and toolkits)
- Includes many VIs completely written in G
- Offers an open system where a user can add new elements and control structures
- Animates results
- Executes compiled code quickly

GSIM also includes the following special features:

- Three different continuous integrators: Euler, Adams, and Runge-Kutta
- Three different discrete integrators: Euler backward, Euler forward, and trapezoidal
- Zero-pole, transfer-function, and state-space representations of systems
- Graphical analysis tools such as Bode plot, Nyquist plot, and root-locus plot
- Design of linear state feedback for arbitrary pole placement

- Elements and control structures in VI form with user-controlled, runtime-adjustable parameters that are important for applications such as autotuning PID parameters
- GSim Manager VI and GSim Synchronizer VI watch real-time behavior and provide warnings
- Large systems can be decomposed based on the subVI technique

Control and Simulation Problems

With GSIM, you can formulate many control and simulation problems in graphical form. In fact, this graphical form offers the most appropriate representation of these tasks. Figure 1-1 shows a differential equation with a given initial condition. The time-dependent function *y* is unknown, indicating a physical, technical, biological, or related quantity, whereas the function *f* and the initial value are known.

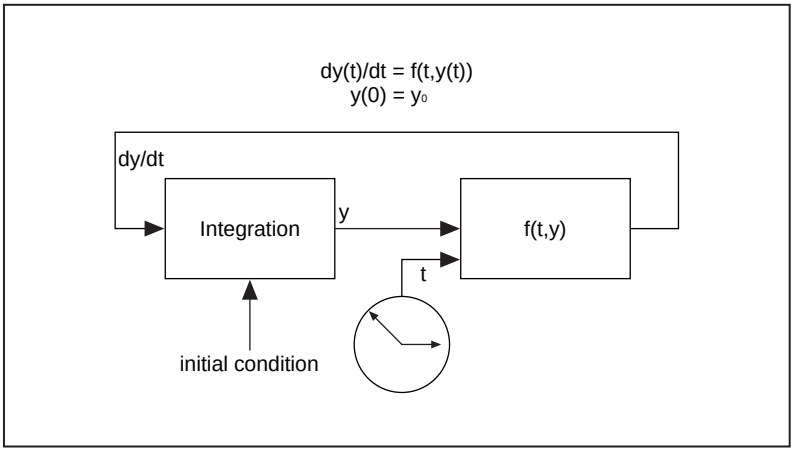


Figure 1-1. Differential Equations (Simulations) Realized as Feedback Structures

The formula at the top of Figure 1-1 offers a compressed description of the problem. There is a one-to-one relation between the formula and the graphical depiction of the initial value problem. This relation is based on the feedback structure shown in Figure 1-1. The box labeled *Integration* outputs *y* if the input is the derivative of *y*. The other box calculates *f* as a function of *y* and time *t*. If the derivative of *y* and f(*t*,*y*) are the same, you can interpret the feedback structure as an equation. The *initial condition* is part of the *Integration* box.

Consider the three main advantages of graphical representation:

- Many control and simulation problems are more complex than the one depicted in Figure 1-1. Often, you need to consider nonlinearities, numerous variables, and special cases, which graphical forms show best. Furthermore, graphical representations can describe problems that formulae cannot.
- Because the graphical concept is extremely intuitive, you can add new features, produce immediate results by adding a new branch (wire), and check and test the entire model or individual parts without influencing or destroying the existing system.
- Because many control and simulation problems are described graphically, you do not need to reformulate the system.

Figure 1-2 shows a pure control system in a graphical representation. Because most real-world control systems receive feedback through hardwire inputs, you do not need to build a feedback structure.

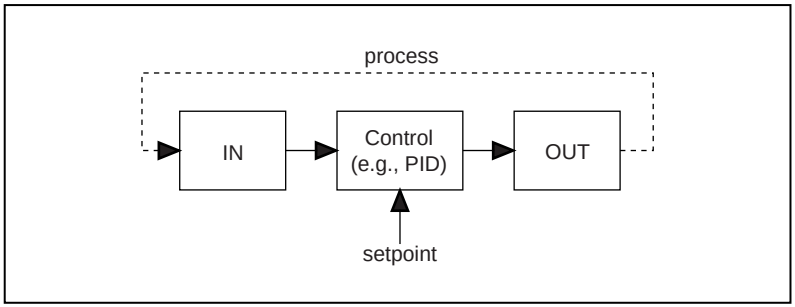


Figure 1-2. Organization of Control Systems Where the Process Provides Feedback

The *Control* block can have any degree of complexity, and an appropriate combination of GSIM routines constructs it. Both the *IN* and *OUT* boxes are part of GSIM.

Control and Simulation Applications

Control and Simulation applications fall into many domains. The following list explains some of the possible applications and provides examples of those applications.

Analysis and Optimization of Linear Time-Invariant Systems— Calculate the overall transfer function of two systems in series, parallel, or feedback connection. Compute the poles, zeros, and other useful parameters such as damping ratios, natural frequency, and settling time. Convert one type of representation to another. Design linear state feedback for arbitrary pole placement. Analyze a control system using graphical methods such as the root-locus plot, the Bode plot, or the Nyquist plot.

Real-Time Control of Deployed Systems —Control inverted pendulums, PID, heating with relays or more sophisticated strategies, cascaded water tanks, and chemical reactors. These applications are based on DAQ boards.

Development of Real-Time Control Systems—You can accomplish the same tasks as those listed for real-time control of deployed systems. Developers can optimize PID parameters, relay thresholds, and limited integrators. With GSIM, you can alter values during runtime or partly implement and partly simulate a new system under development.

Simulation and Simulation with Data Acquisition—Simulate situations in fields such as engineering, physics, chemistry, biology, astronomy, and math. Many simulations are based on input data (for example, the current position of a space craft or the current flow into a tank) and you can predict the behavior of the system in the near future (for example, the position of that space craft in one hour or the water level of that tank in one day with given inflow).

Education—Simulate linear and nonlinear systems of any desired complexity. For example, you can simulate mechanics, analog circuits, biology (predator-prey model), celestial bodies, physics, inverted pendulum, study of numerical algorithms, and animations.

Control and Simulation Libraries and VIs

Table 1-1 lists all the libraries and VIs in the Control and Simulation Software for G.

Table 1-1. Control and Simulation Libraries and VIs

Library	Library VIs
BASIC	GSim Series GSim Parallel GSim Feedback GSim Compute Zeros GSim Compute Poles GSim Compute Parameters GSim Normalize
CONVERT	GSim SS2RP GSim SS2TF GSim SS2ZP GSim TF2SS GSim ZP2TF GSim Modal Form
FEEDBACK	GSim Design from SS GSim Design from TF GSim Design from ZP
FREQRESP	GSim Bode Plot from SS GSim Bode Plot from TF GSim Root Locus GSim Root Locus Plot Utility GSim Nyquist Plot
CONNECT	GSim Initial Condition GSim Input Selector GSim Trigger

Table 1-1. Control and Simulation Libraries and VIs (Continued)

Library	Library VIs
DISCRETE	GSim Discrete Filter GSim Discrete State-Space GSim Discrete Transfer Function GSim Discrete Zero-Pole GSim Discrete Time-Integrator GSim First-Order Hold GSim Unit Delay GSim Zero-Order Hold
LINEAR	GSim Derivative GSim Integrator GSim Limited Integrator GSim PID Parallel GSim PID Serial GSim State-Space GSim Transfer Function IC GSim Transfer Function GSim Zero-Pole
NONLIN	GSim Dead Zone GSim Friction GSim Memory GSim Quantizer GSim Rate Limiter GSim Relay GSim Saturation GSim Switch GSim Transport Delay GSim Zero Crossing
SHARED	GSim Initialize GSim Manager GSim Synchronizer GSim Timer
SINKS	GSim Data Out GSim Scope 1d GSim Stop

Table 1-1. Control and Simulation Libraries and VIs (Continued)

Library	Library VIs
SOURCES	GSim Chirp Signal GSim Collect Data In GSim Data In GSim Pulse Generator GSim Ramp GSim Realtime Clock GSim Signal Generator GSim Simulation Clock GSim Sine Wave GSim Step
XMPLCONT	Bode Application Bode Plot Wizard Electronic Pacemaker Nyquist Plot Wizard Quarter Car Dynamics Robot Joint Design Root Locus Wizard Suspended Ball
XMPLSIM1	Child Toy Example Gun and Moving Target Example Lorenz Attractor Example PID Example Relay Feedback Example RLC Circuit Example Water Tank Example

Table 1-1. Control and Simulation Libraries and VIs (Continued)

Library	Library VIs
XMPLSIM2	Control Example Curtis-Hirschfelder Example F14 Example H(s) Example IC H(s) Example House Temperature Control Example House Thermostat Example Integration Example Inverted Pendulum Example Limited Integration Example Mathieu Example Parameter Estimation Example Population Growth Example Predator-Prey Example Van der Pol Example

Installation Procedure

The following sections contain instructions for installing the Control and Simulation Software for G on the Windows 95/NT, Windows 3.*x*, and Power Macintosh platforms.

Windows 95/NT

Complete the following steps to install the Control and Simulation Software for G.

1. Launch Windows 95 or NT.
2. Insert the Control and Simulation Software for G CD into the CD-ROM drive.

3. From the **Start** menu, choose **Run** and type `A:\setup.exe`.
4. Follow the instructions on your screen.

Once you have completed the on-screen installation instructions, you are ready to run the Control and Simulation Software for G.

Windows 3.*x*

Complete the following steps to install the Control and Simulation Software for G.

1. Launch Windows.
2. Insert the Control and Simulation Software for G CD into the CD-ROM drive.
3. From the File Manager, run `SETUP.EXE`.
4. Follow the instructions on your screen.

Once you have completed the on-screen installation instructions, you are ready to run the Control and Simulation Software for G.

Power Macintosh

Complete the following steps to install the Control and Simulation Software for G.

1. Insert the Control and Simulation Software for G CD into the CD-ROM drive.
2. Follow the instructions on your screen.

Once you have completed the on-screen installation instructions, you are ready to run the Control and Simulation Software for G.

Getting Help

You can access the online reference by selecting **Help»Online Reference** or pressing `<Ctrl-H>`.

2

Control and Simulation Problems

This chapter presents control and simulation examples, prescribes a step-by-step approach, and offers some restrictions and hints for developing control and simulation programs.

Example: House Temperature Control

Figure 2-1 depicts the front panel of the House Temperature Control Example VI. The upper-left section of the front panel contains the input parameters. You must fix the values of **dt** (sampling time), **end time** (simulation time, not real time), and **continuous integrator** at the beginning of the calculation. You can alter all other parameters—**Model/DAQ** switch (switches between the simulated and actual process), **setpoint**, **controller** switch, and all parameters that determine the control strategies in detail—during runtime. The chart illustrates the behavior of the control system. The boolean **timing** indicates a violation of the real-time behavior of the realized control system and provides an important warning (that is, the LED turns on when the system is running slower than real time).

Figure 2-2 is the main diagram for the House Temperature Control Example VI. All control and simulation tasks in this software are based on a While Loop. Feedback structures are realized by the use of local variables. **temperature** is the only feedback variable. The GSim Manager VI controls the simulation task (see the lower part of the While Loop).

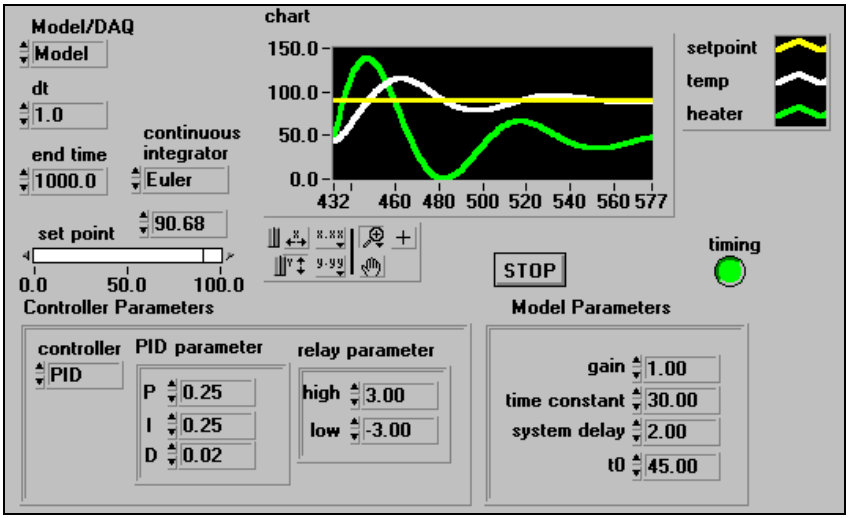


Figure 2-1. House Temperature Control Example VI Front Panel

The VI to the left of the While Loop initializes **dt** (step rate**)**, **end time**, and **continuous integrator**. All other parameters are part of the While Loop, and you can alter them during runtime of the control or simulation task. The setpoint can be modified by the user or as a result of a direct measurement. In this example, only the user can modify **setpoint**.

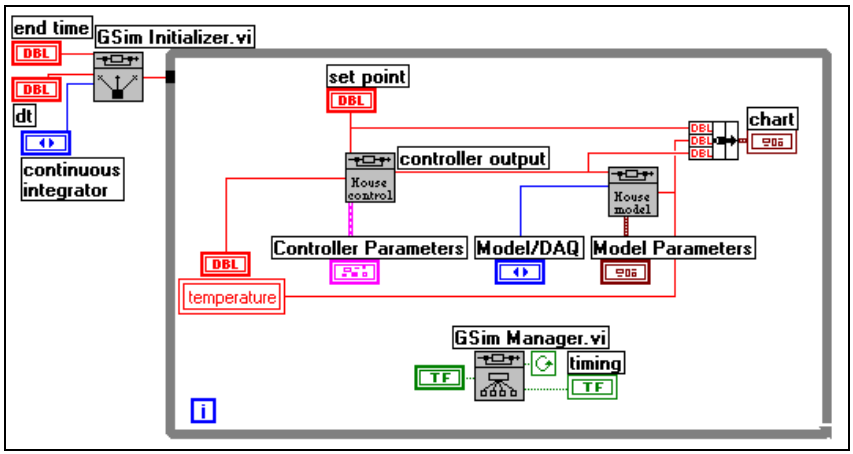


Figure 2-2. House Temperature Control Example VI Diagram

The graphical presentation of the calculated results is displayed in the upper right section of the While Loop. You can use all LabVIEW graph handling elements. For this example, the values for **setpoint**, **temperature**, and **heater** were chosen to display in a chart.

Example: PID Controllers in GSIM

GSim PID Parallel and GSim PID Serial are part of `LINEAR.llb` and represent a general PID controller in GSIM notation. Figure 2-3 shows that the underlying GSIM program is a combination of a proportional factor, an integrator, and a differentiator. All necessary parameters are controls of this VI. GSIM can perform approximately 5000 PID control loops per second on a 166 MHz, Pentium-based PC. A PID Benchmark is part of the example library delivered with the Control and Simulation Software for G.

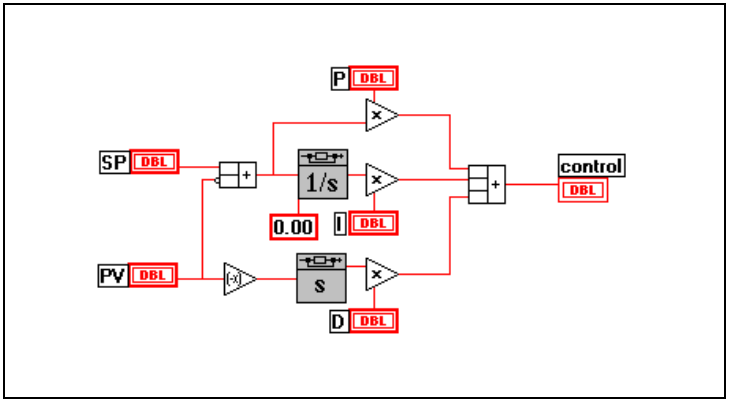


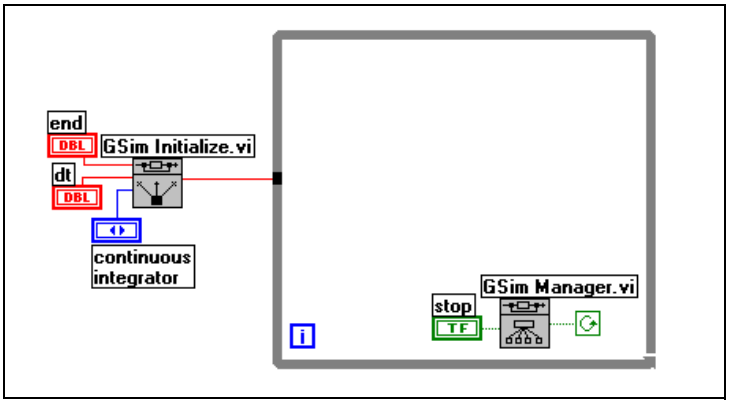
Figure 2-3. PID Controller in GSIM Notation

A Step-By-Step Approach to GSIM Programs

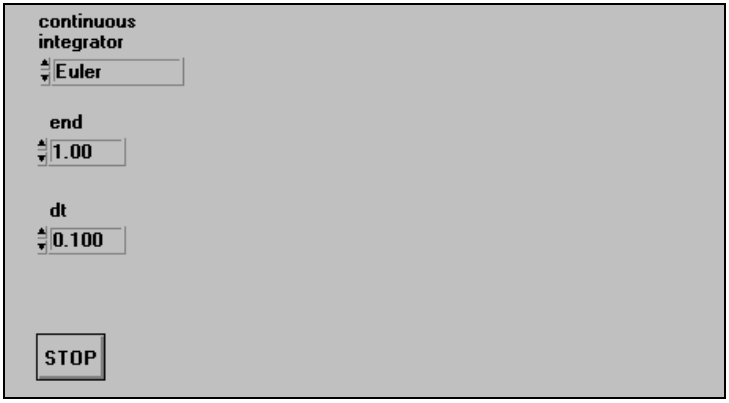
All GSIM programs have a similar structure, as demonstrated in the following step-by-step activity.

Step 1: Generate the General GSIM Frame

1. Open a new VI by selecting **File»New**. If you have closed all VIs, select **New VI** from the dialog box.
2. Add a While Loop to the diagram.
3. Choose `GSim Manager.vi` and `GSim Initializer.vi` from `SHARE.llb` and place both as shown in the following block diagram.

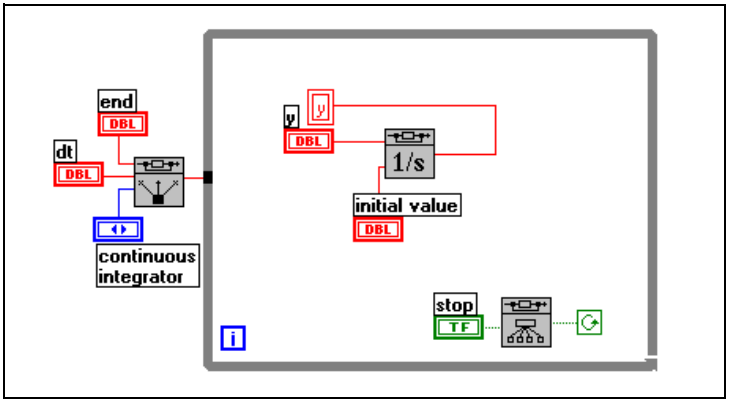


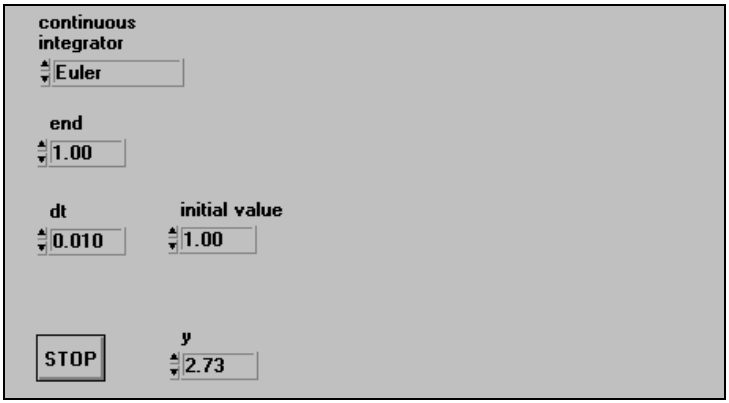
4. Connect the dummy output of `GSim Initializer.vi` with the While Loop to guarantee that the sequence executes correctly.
5. Connect the upper output of `GSim Manager.vi`, **stop**, with the conditional terminal of the While Loop. The GSim Manager VI controls all data flows and handles error conditions. Usually, a control or simulation task stops when the end time is reached or if an error occurs.
6. Connect a STOP button to `GSim Manager.vi` (optional). The GSim Manager VI accepts a boolean true as an explicit stop command.
7. Create controls for **end time**, **dt**, and **continuous integrator** to `GSim Initializer.vi` on the front panel, as shown in the following figure.



Step 2: Create the Feedback Structure

1. Add a numerical control representing the feedback to the front panel and name it (in this example, **y**).
2. Create a local variable with the same name (**y**).
3. Pop up on the local variable **y** and change to **Read Local Variable**.
4. Create a second control with the name **initial value**.
5. Open the `LINEAR.llb` library and choose `GSim Integrator.vi`.
6. Connect all controls, the local variable, and `GSim Integrator.vi` as shown in the following block diagram. Notice that the numerical control **y** and its local variable form the two parts of a feedback structure.

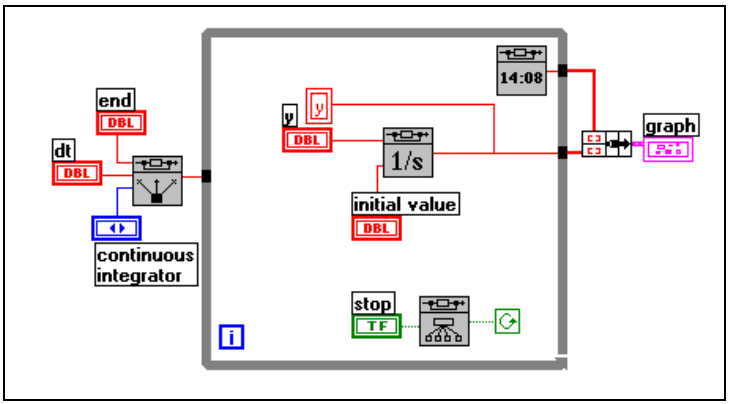


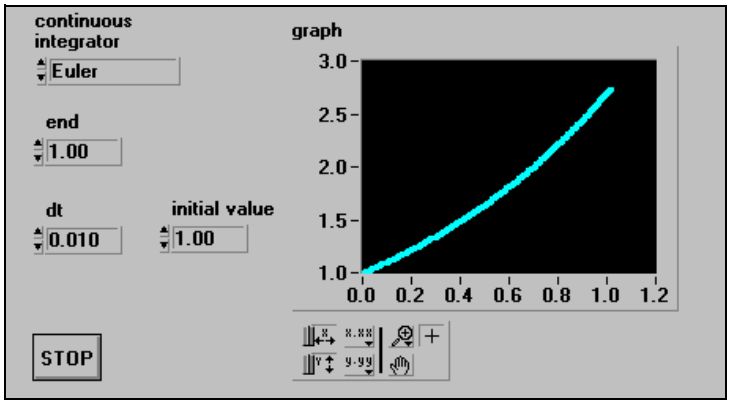


7. Run the VI under various conditions and watch the behavior of the numerical control as you alter the initial conditions.

Step 3: Adjust the User Interface

1. Add the user interface as shown in the following figures. You can find `GSim Simulation Clock.vi` in `SOURCE.llb`.





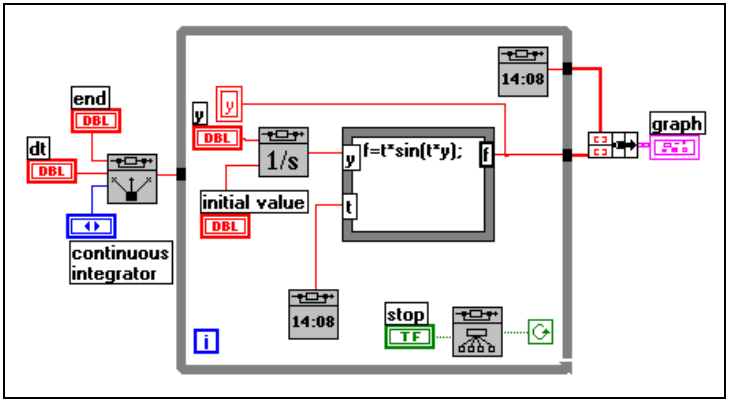
2. Hide the numerical control **y**. Although not necessary, hiding this control results in an essential increase of speed.
3. Fix the following values:
   - **end time**=1
   - **dt**=0.01
   - **continuous integrator**=Euler
   - **initial value** =1.0.

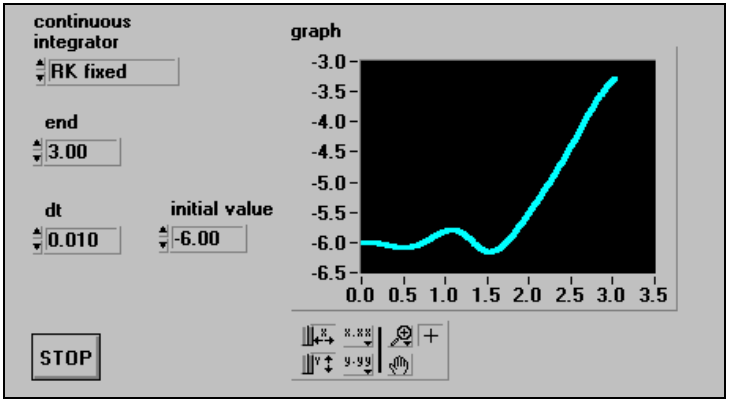
   The calculated graph depicts the function exp(t) in the interval (0,1).

The central point of this GSIM program is the integrator. You can interpret the input as the derivative $dy(t)/dt$ of an unknown function $y(t)$, and the output is the function $y(t)$ itself. Because of the feedback structure, both $dy(t)/dt$ and $y(t)$ must be the same. In other words, the differential equation $dy(t)/dt=y(t)$ is solved where **initial value** is given (in our case, **initial value** is equal to 1). This differential equation has the unique solution $y(t)=\exp(t)$.

Step 4: Modify the Program to Solve First-Order Differential Equations

1. Modify the developed GSIM program as shown in the following figures. Refer to Figure 1-1, *Differential Equations (Simulations) Realized as Feedback Structures*, for the underlying theory. This VI solves all first-order, ordinary differential equations with given initial conditions.

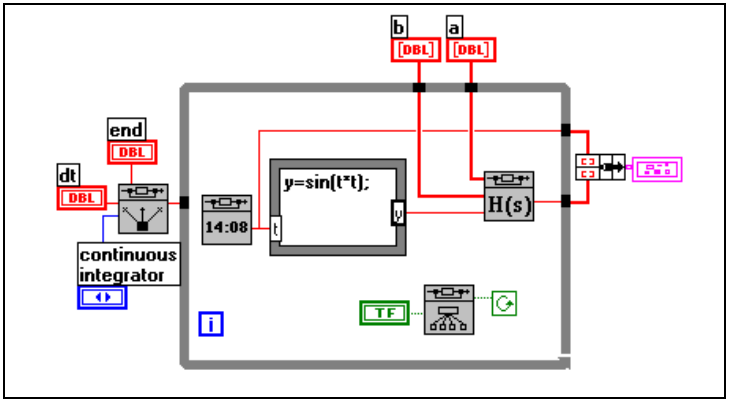


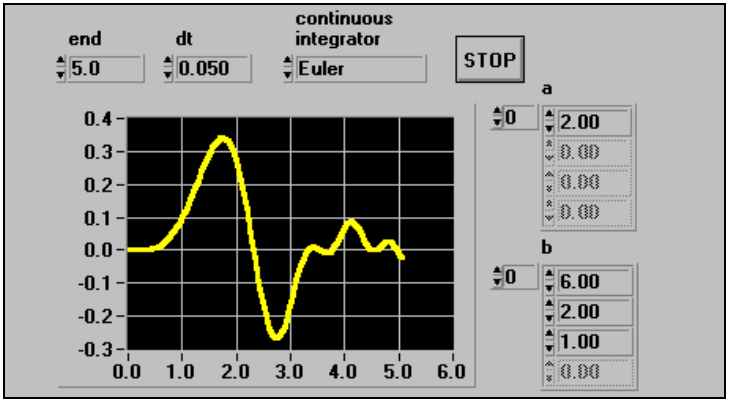


2. Alter all parameters and the contents of the formula node and observe the calculated results.
3. Run some tests in the debugging mode (choose **end time**=1 and **dt**=0.01).

Step 5: Study Other GSIM Programs

Study the GSIM program shown in the following figures. This VI is part of the example section delivered with GSIM (`H(s) Example.vi`). The program realizes a general transfer function where the input is given by a formula.





Some Restrictions and Hints

Although you have a lot of design freedom with GSIM, during the development of new control and simulation projects, National Instruments strongly recommends that developers adhere to the following guidelines. These guidelines apply to Simulation VIs only.

Guideline 1: Do Not Change Execution Options on Reentrant VIs

Most GSIM VIs are reentrant. Because changing the execution options of any of these VIs can result in an unexpected behavior of the solution, National Instruments recommends that you do not change any execution options for these VIs. Do not place any of the GSIM VIs in a nested loop (a loop within a loop).

If you need an integrator for an array of variables, choose the array version of GSim Integrator, which is `GSim Integrator (array).vi`.

Guideline 2: Include GSim Initialize and GSim Manager

You must include the GSim Initialize VI and GSim Manager VI in any and all GSim programs.

Guideline 3: Run Only One GSIM Program at a Time

Do not run two or more GSIM programs concurrently. You can build up programs of any complexity based on an arbitrarily deep hierarchy, but two or more concurrently running VIs interfere with each other.

Guideline 4: Hide Control/Indicator Feedback Structures

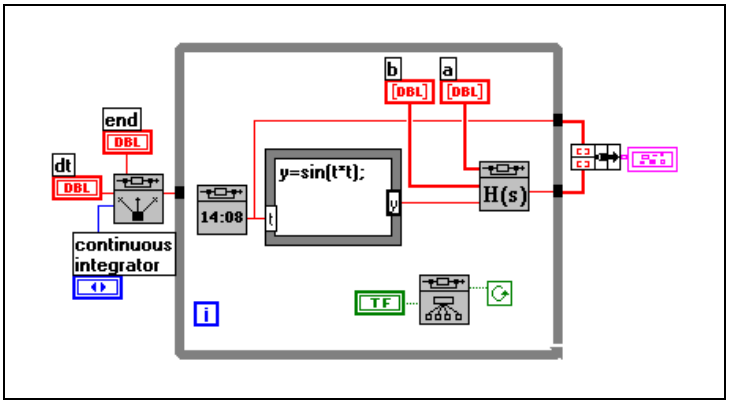
Hide all controls and indicators that represent feedback structures to significantly improve the runtime behavior of the solution.

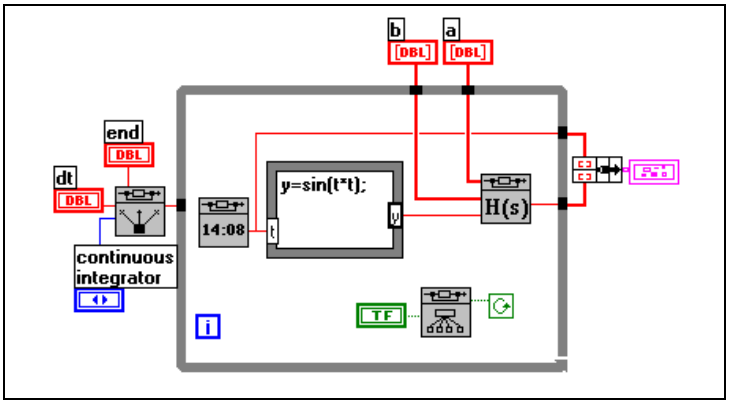
Guideline 5: Evaluate Results Carefully

Evaluate all results very carefully. No mathematical method can guarantee its correctness in all cases. Try different integrators and choose the sampling rate value **dt** carefully, which is a tradeoff between speed and correctness. Although a general rule does not exist, we recommend you use the Euler method for control routines; use the Adams and Runge-Kutta methods for simulation tasks, as they are more sophisticated than the Euler method.

Guideline 6: Position Controls Carefully on the Diagram

The position of control parameters on the diagram is important. Controls that are part of the While Loop can directly and immediately determine the behavior of GSIM VIs. A control parameter that is not part of the While Loop delivers a valid value only at the very beginning of the program. Choose the appropriate method depending on your given problem. Although they are only slightly different programs, the following two diagrams explain both situations. Notice the position of controls **a** and **b,** which are important to the runtime behavior of the GSIM program.





3

Basic VIs

This chapter describes the VIs that perform basic operations required for the analysis and design of linear, time-invariant systems.

When two systems are connected in a series or parallel connection, as shown in Figure 3-1, you often must compute the representation of the overall system. Similarly, when one system is connected in the feedback loop of another system, you might be interested in the representation of the combined system. In many applications, you might want to compute the poles and zeros of the system given the transfer function of the system. Information pertaining to poles and zeros of the system is useful in determining the stability of the system. Parameters such as natural frequencies, damping ratios, settling time, and maximum percent overshoot also might offer insight. Use the VIs included in the `BASIC.llb` library to perform these basic operations.

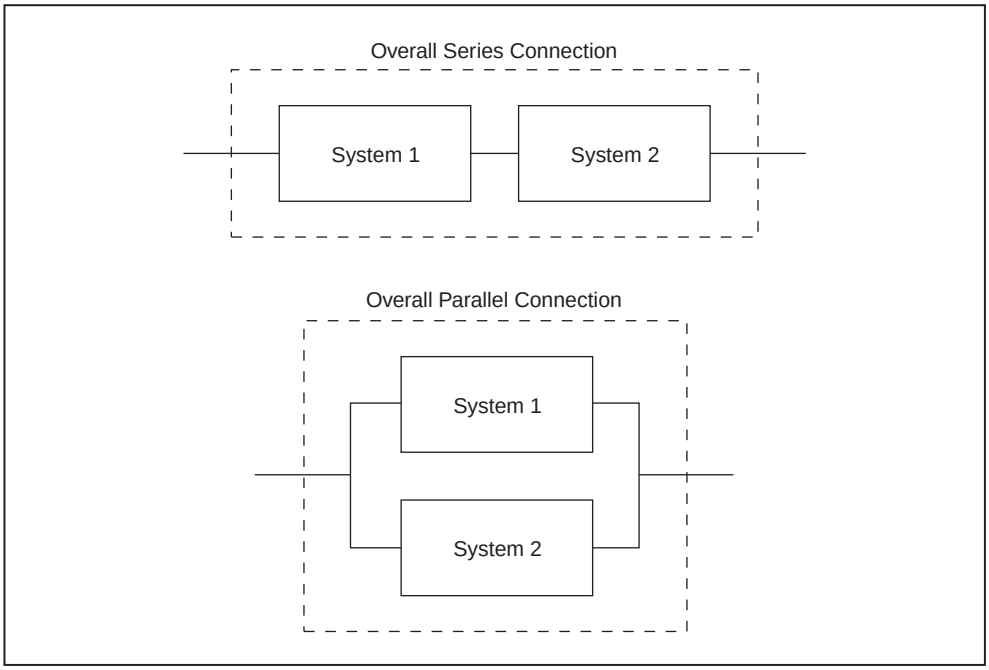
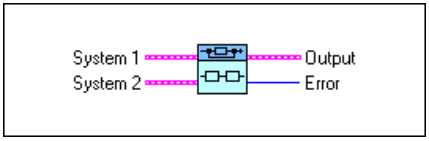


Figure 3-1. Two Systems, System 1 and System 2, in Series and Parallel Connection

GSim Series

Computes the overall transfer function of two systems connected in series.



System 1 is the transfer function $G_1(s)$ of the first system in the series connection. **System 1** contains the following two elements:

Betas1 specifies the numerator coefficients of the transfer function. Refer to Appendix B, *Description of the Betas and Alphas Parameters*, for information about entering coefficients.

Alphas1 specifies the denominator coefficients of the transfer function. Refer to Appendix B, *Description of the Betas and Alphas Parameters*, for information about entering coefficients.

System 2 is the transfer function $G_2(s)$ of the second system in the series connection. **System 2** contains the following two elements:

Betas2 specifies the numerator coefficients of the transfer function. Refer to Appendix B, *Description of the Betas and Alphas Parameters*, for information about entering coefficients.

Alphas2 specifies the denominator coefficients of the transfer function. Refer to Appendix B, *Description of the Betas and Alphas Parameters*, for information about entering coefficients.

Output is the overall transfer function $G(s)$ where $G(s) = G_1(s) \times G_2(s)$. **Output** contains the following two elements:

Betas specifies the numerator coefficients of the overall transfer function. Refer to Appendix B, *Description of the Betas and Alphas Parameters*, for information on interpreting coefficients.

Alphas specifies the denominator coefficients of the overall transfer function. Refer to Appendix B, *Description of the Betas and Alphas Parameters*, for information on interpreting coefficients.

Error. See Appendix A, *Error Codes*, for a description of the error.

If $G_1(s)$ represents the transfer function of the first system and $G_2(s)$ represents the transfer function of the second system, the overall transfer function $G(s)$ of the two systems connected in series is given by

$$G(s) = G_1(s) \times G_2(s)$$

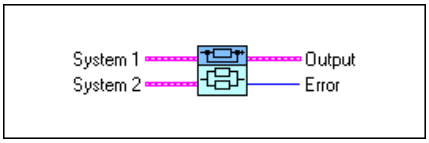
If $G_1(s) = (s-1)/(s+1)$ is the transfer function of the first system, and the transfer function of the second system is $G_2(s) = s/(s^2-1)$, the overall transfer function of the series connection is

$$G(s) = G_1(s) \times G_2(s) = \frac{s^2 - s}{s^3 + s^2 - s - 1}$$

If n systems are connected in series, you can execute this VI ($n-1$) times. First, compute the transfer function of the series connection of the first two systems. Then, compute the transfer function of the series connection of the resulting system and the third system. You can continue this process in a loop ($n-1$) times. These operations are associative.

GSim Parallel

Computes the overall transfer function of two systems connected in parallel.



System 1 is the transfer function $G_1(s)$ of the first system in the parallel connection. **System 1** contains the following two elements:

Betas1 specifies the numerator coefficients of the transfer function. Refer to Appendix B, *Description of the Betas and Alphas Parameters*, for information about entering coefficients.

Alphas1 specifies the denominator coefficients of the transfer function. Refer to Appendix B, *Description of the Betas and Alphas Parameters*, for information about entering coefficients.

System 2 is the transfer function $G_2(s)$ of the second system in the parallel connection. **System 2** contains the following two elements:

Betas2 specifies the numerator coefficients of the transfer function. Refer to Appendix B, *Description of the Betas and Alphas Parameters*, for information about entering coefficients.

Alphas2 specifies the denominator coefficients of the transfer function. Refer to Appendix B, *Description of the Betas and Alphas Parameters*, for information about entering coefficients.

Output is the overall transfer function $G(s)$ where

$$G(s) = G_1(s) + G_2(s)$$

Betas specifies the numerator coefficients of the overall transfer function. Refer to Appendix B, *Description of the Betas and Alphas Parameters*, for information on interpreting coefficients.

Alphas specifies the denominator coefficients of the overall transfer function. Refer to Appendix B, *Description of the Betas and Alphas Parameters*, for information on interpreting coefficients.

Error. See Appendix A, *Error Codes*, for a description of the error.

If $G_1(s)$ represents the transfer function of the first system and $G_2(s)$ represents the transfer function of the second system, the overall transfer function $G(s)$ of the two systems connected in parallel is given by

$$G(s) = G_1(s) + G_2(s)$$

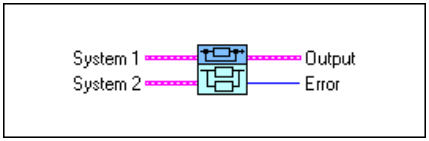
If the transfer function of the first system is $G_1(s) = 1/s$ and the transfer function of the second system is $G_2(s) = 1/s^2$, the overall transfer function is

$$G(s) = G_1(s) + G_2(s) = \frac{s^2 + s}{s^3}$$

If *n* systems are connected in parallel, you can execute this VI (*n*−1) times. First, compute the transfer function of the parallel connection of the first two systems. Then, compute the transfer function of the parallel connection of the resulting system and the third system. You can continue this process in a loop (*n*−1) times. These operations are associative.

GSim Feedback

Computes the overall transfer function of a main system connected in a closed-loop feedback connection. This VI assumes a negative feedback loop.



System 1 is the transfer function $G_1(s)$ of the main system in the closed-loop connection. **System 1** contains the following two elements:

Betas1 specifies the numerator coefficients of the transfer function. Refer to Appendix B, *Description of the Betas and Alphas Parameters*, for information about entering coefficients.

Alphas1 specifies the denominator coefficients of the transfer function. Refer to Appendix B, *Description of the Betas and Alphas Parameters*, for information about entering coefficients.

System 2 is the transfer function $G_2(s)$ of the feedback system in the closed-loop connection. **System 2** contains the following two elements:

Betas2 specifies the numerator coefficients of the transfer function. Refer to Appendix B, *Description of the Betas and Alphas Parameters*, for information about entering coefficients.

Alphas2 specifies the denominator coefficients of the transfer function. Refer to Appendix B, *Description of the Betas and Alphas Parameters*, for information about entering coefficients.

Output is the overall transfer function $G(s)$. If $G_1(s)$ represents the transfer function of the main system and $G_2(s)$ represents the transfer function of the feedback system, the overall transfer function $G(s)$ of the system connected in the feedback loop is given by

$$G(s) = \begin{cases} \dfrac{G_1(s)}{1 + G_1(s)G_2(s)} & \text{for negative feedback loops} \\ \dfrac{G_1(s)}{1 - G_1(s)G_2(s)} & \text{for positive feedback loops} \end{cases}$$

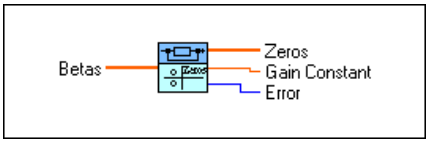
Betas specifies the numerator coefficients of the transfer function. Refer to Appendix B, *Description of the Betas and Alphas Parameters*, for information on interpreting coefficients.

Alphas specifies the denominator coefficients of the overall transfer function. Refer to Appendix B, *Description of the Betas and Alphas Parameters*, for information on interpreting coefficients.

Error. See Appendix A, *Error Codes*, for a description of the error.

GSim Compute Zeros

Computes the zeros of a system, given the numerator coefficients of its transfer function.



[DBL] **Betas** specifies the numerator coefficients of the system transfer function for which the zeros are being computed. Refer to Appendix B, *Description of the Betas and Alphas Parameters*, for information about entering coefficients.

[CDB] **Zeros** are the zeros of the system.

[DBL] **Gain Constant** is the gain constant of the system.

[I32] **Error**. See Appendix A, *Error Codes*, for a description of the error.

If the numerator of the transfer function is given by

$$H(s) = \beta_n s^n + \beta_{n-1} s^{n-1} + \ldots + \beta_1 s + \beta_0$$

then it can be written in terms of zeros z_i and gain constant K as

$$H(s) = K \prod_{i=0}^{n-1} (s - z_i)$$

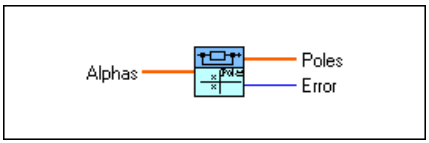
The zeros of the transfer function $G(s)$ are the values of s for which the transfer function equals zero. As an example, consider the following equation:

$$G(s) = \frac{s^2 + 3s + 2}{s^3 + s^2 + s + 1}$$

The zeros of the system are at –1 and –2.

GSim Compute Poles

Computes the poles of a system, given the denominator coefficients of its transfer function.



[DBL] **Alphas** specifies the denominator coefficients of the system transfer function for which the poles are being computed. See Appendix B, *Description of the Betas and Alphas Parameters*, for information about entering coefficients.

[CDB] **Poles** denotes the poles of the system.

I32 **Error**. See Appendix A, *Error Codes*, for a description of the error.

If the denominator of the transfer function is given by

$$H(s) = \alpha_n s^n + \alpha_{n-1} s^{n-1} + \ldots + \alpha_1 s + \alpha_0$$

it can be written in terms of poles p_i as

$$H(s) = \prod_{i=0}^{n-1} (s - p_i) \qquad \text{if } \alpha_n = 1$$

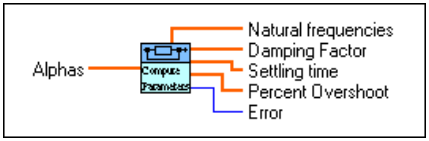
The poles of the transfer function $G(s)$ are the values of s for which the transfer function equals infinity. As an example, consider the following equation:

$$G(s) = \frac{s^2 + 3s + 2}{s^3 - 3s^2 + 2s + 0}$$

The poles of the system are at 0, 1, and 2.

GSim Compute Parameters

Computes the parameters natural frequencies, damping factor, settling time, and percent overshoot for a system, given the denominator coefficients of its transfer function.



[DBL] **Alphas** specifies the denominator coefficients of the system transfer function for which the parameters are computed. See Appendix B, *Description of the Betas and Alphas Parameters*, for information about entering coefficients.

[DBL] **Natural frequencies** denotes the natural frequencies of the system.

[DBL] **Damping Factor** is the damping factor of the system.

[DBL] **Settling time** is the settling time of the system.

[DBL] **Percent Overshoot** is the percent overshoot of the system.

[I32] **Error**. See Appendix A, *Error Codes*, for a description of the error.

Given a pair of conjugate complex poles $\alpha + j\beta$, the natural frequencies ω_n and damping ratio ξ are defined by the following equations:

$$\omega_n = \sqrt{\alpha^2 + \beta^2}$$

$$\xi = \frac{-\alpha}{\sqrt{\alpha^2 + \beta^2}}$$

The settling time t_s is given by

$$t_s = \frac{4}{\xi\omega_n}$$

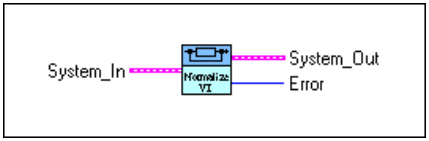
The maximum percent overshoot m is given by

$$m = e^{\frac{-\xi\pi}{\sqrt{1-\xi^2}}} \times 100$$

For a pair of conjugate complex poles, these parameters appear twice in the output array—once for each pole. For real poles, the VI gives the absolute value of the pole as natural frequency and as damping ratio ± 1, depending on the negative or positive sign of the eigenvalue.

GSim Normalize

Normalizes the transfer function of a system; that is, it normalizes the coefficient of the highest order term in the denominator equal to one.



System_In is the transfer function of the system that needs to be normalized. **System_In** contains the following two elements:

Betas_in specifies the numerator coefficients of the transfer function of the system. Refer to Appendix B, *Description of the Betas and Alphas Parameters*, for information about entering coefficients.

Alphas_in specifies the denominator coefficients of the transfer function of the system. Refer to Appendix B, *Description of the Betas and Alphas Parameters*, for information about entering coefficients.

System_Out is the transfer function of the normalized system. This cluster contains the following two elements:

Betas_out specifies the numerator coefficients of the transfer function of the normalized system. Refer to Appendix B, *Description of the Betas and Alphas Parameters*, for information on interpreting coefficients.

Alphas_out specifies the denominator coefficients of the transfer function of the normalized system. The coefficient of the highest order term is equal to one. Refer to Appendix B, *Description of the Betas and Alphas Parameters*, for information on interpreting coefficients.

Error. See Appendix A, *Error Codes*, for a description of the error.

If the transfer function G(s) of a system is given by

$$G(s) = \frac{3s + 9}{6s^2 + 9s + 3}$$

the normalized transfer function is given by

$$G(s) = \frac{0.5s + 1.5}{s^2 + 1.5s + 0.5}$$

4

Conversion VIs

This chapter explains different forms of system representation and the VIs that you can use to convert one form to another.

State-Space (SS) Representation

In this representation, the system is described in terms of its state variables. Let x_1 and x_2 represent two state variables of the system. Let **u** represent the input and **y** represent the output vector. Then the state-space representation is given by1

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} a_{00} & a_{01} \\ a_{10} & a_{11} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} \mathbf{u}$$

$$\mathbf{y} = \begin{bmatrix} c_1 & c_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} d \end{bmatrix} \mathbf{u}$$

In a condensed form, these two equations usually are written as

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}$$

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

If n represents the number of states, m the number of inputs, and p the number of outputs, matrix **A** is n-by-n, matrix **B** is n-by-m, matrix **C** is p-by-n, and matrix **D** is p-by-m.

The matrices **A**, **B**, **C**, and **D** completely determine the state-space representation. This type of representation is important because it describes the behavior of the internal variables (states) of the system.

Modal-Form (MF) Representation

The modal-form representation is a special form of the state-space representation where matrix $\bar{\mathbf{A}}$ is diagonal. The diagonal elements of $\bar{\mathbf{A}}$ are the eigenvalues of the system. The modal-form representation of a system is given by

$$\dot{\mathbf{x}} = \bar{\mathbf{A}}\mathbf{x} + \bar{\mathbf{B}}\mathbf{u}$$

$$\mathbf{y} = \bar{\mathbf{C}}\mathbf{x} + \bar{\mathbf{D}}\mathbf{u}$$

Transfer-Function (TF) Representation

A transfer function relates the output of the system to its input. It does not describe the internal behavior of the system. In the frequency domain, the transfer function is usually written in terms of the Laplace transform as

$$H(s) = \frac{Y(s)}{U(s)} = \frac{\beta_n s^n + \beta_{n-1} s^{n-1} + \ldots + \beta_1 s + \beta_0}{\alpha_m s^m + \alpha_{m-1} s^{m-1} + \ldots + \alpha_1 s + \alpha_0}$$

Thus, the system is represented in terms of the coefficients of the numerator and denominator of the system transfer function.

Zero-Pole (ZP) Representation

A system also can be represented in terms of its zeros and poles. The zeros of the system are the roots of the numerator of the transfer function. At these values, the transfer function of the system becomes equal to zero. The poles of the system are the roots of the denominator of the transfer function. At these values, the transfer function of the system tends towards infinity. In the frequency domain, the zero-pole representation of the system is written as

$$H(s) = \frac{K\prod_{i=0}^{n-1}(s - z_i)}{\prod_{i=0}^{m-1}(s - p_i)}$$

Thus, the system is represented in terms of the zeros, poles, and the gain constant K.

Residual-Pole (RP) Representation

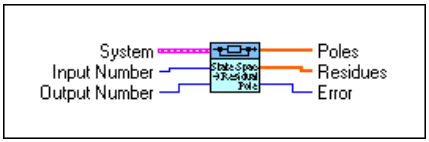
In the residual-pole representation, the system is described in terms of its poles and the residuals at the poles of the system. In the frequency domain, the residual-pole representation of the system is written as

$$H(s) = \sum_{i=0}^{m-1} \frac{K_i}{s - p_i}$$

In many situations, you must convert one form of representation to another. Use the VIs in the `CONVERT.llb` library to carry out these conversions.

GSim SS2RP

Converts the state-space representation to the residual-pole representation.



System is the state-space representation. **System** contains the following four elements:

A is matrix **A** of the state-space representation.

B is matrix **B** of the state-space representation.

C is matrix **C** of the state-space representation.

D is matrix **D** of the state-space representation.

Input Number is the input number for multiple-input systems. This parameter defaults to 1.

Output Number is the output number for multiple-output systems. This parameter defaults to 1.

Poles denotes the poles of the system in the residual-pole representation.

Residues specifies the residues in the residual-pole representation.

Error. See Appendix A, *Error Codes*, for a description of the error.

The state-space representation is given by

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

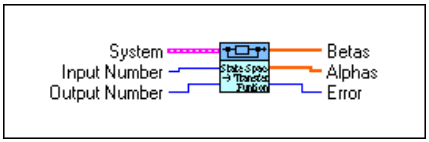
$$\mathbf{y} = \mathbf{Cx} + \mathbf{Du}$$

and the residual-pole representation is given by

$$H(s) = \sum_{i=0}^{m-1} \frac{K_i}{s - p_i}$$

GSim SS2TF

Converts the state-space representation to the transfer-function representation.



System is the state-space representation. **System** contains the following four elements:

A is matrix **A** of the state-space representation.

B is matrix **B** of the state-space representation.

C is matrix **C** of the state-space representation.

D is matrix **D** of the state-space representation.

Input Number is the input number for multiple-input systems. This parameter defaults to 1.

Output Number is the output number for multiple-output systems. This parameter defaults to 1.

Betas specifies the numerator coefficients of the transfer function. Refer to Appendix B, *Description of the Betas and Alphas Parameters*, for information on interpreting coefficients.

Alphas specifies the denominator coefficients of the transfer function. Refer to Appendix B, *Description of the Betas and Alphas Parameters*, for information on interpreting coefficients.

Error. See Appendix A, *Error Codes*, for a description of the error.

The state-space representation is given by

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}$$

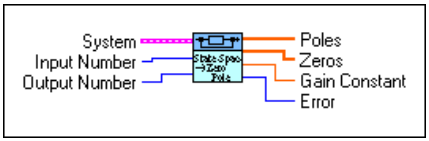
$$\mathbf{y} = \mathbf{C}\mathbf{x} + \mathbf{D}\mathbf{u}$$

and the transfer-function representation is given by

$$H(s) = \frac{\beta_n s^n + \beta_{n-1} s^{n-1} + \ldots + \beta_1 s + \beta_0}{\alpha_m s^m + \alpha_{m-1} s^{m-1} + \ldots + \alpha_1 s + \alpha_0}$$

GSim SS2ZP

Converts the state-space representation to the zero-pole representation.



System is the state-space representation. **System** contains the following four elements:

A is matrix **A** of the state-space representation.

B is matrix **B** of the state-space representation.

C is matrix **C** of the state-space representation.

D is matrix **D** of the state-space representation.

Input Number is the input number for multiple-input systems. This parameter defaults to 1.

Output Number is the output number for multiple-output systems. This parameter defaults to 1.

Poles specifies the poles of the system.

Zeros specifies the zeros of the system.

Gain Constant is the gain constant of the system.

Error. See Appendix A, *Error Codes*, for a description of the error.

The state-space representation is given by

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

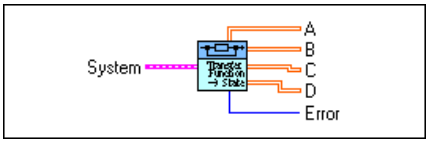
$$\mathbf{y} = \mathbf{Cx} + \mathbf{Du}$$

and the zero-pole representation is given by

$$H(s) = \frac{K\prod_{i=0}^{n-1} s - z_i}{\prod_{i=0}^{m-1} s - p_i}$$

GSim TF2SS

Converts the transfer-function representation to the state-space representation.



System is the transfer-function representation. **System** contains the following two elements:

Betas specifies the numerator coefficients of the transfer function. Refer to Appendix B, *Description of the Betas and Alphas Parameters*, for information about entering coefficients.

Alphas specifies the denominator coefficients of the transfer function. Refer to Appendix B, *Description of the Betas and Alphas Parameters*, for information about entering coefficients.

A is matrix **A** of the state-space representation.

B is matrix **B** of the state-space representation.

C is matrix **C** of the state-space representation.

D is matrix **D** of the state-space representation.

Error. See Appendix A, *Error Codes*, for a description of the error.

The transfer-function representation is given by

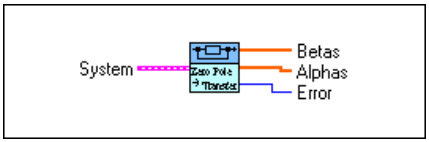
$$H(s) = \frac{\beta_n s^n + \beta_{n-1} s^{n-1} + \ldots + \beta_0}{\alpha_m s^m + \alpha_{m-1} s^{m-1} + \ldots + \alpha_0}$$

and the SISO state-space representation is given by

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}u$$
$$y = \mathbf{C}\mathbf{x} + \mathbf{D}u$$

GSim ZP2TF

Converts the zero-pole representation to the transfer-function representation.



System is the zero-pole representation of the system with a gain constant equal to 1. **System** contains the following two elements:

Zeros specifies the zeros of the system.

Poles specifies the poles of the system.

Betas specifies the numerator coefficients of the transfer function. Refer to Appendix B, *Description of the Betas and Alphas Parameters*, for information on interpreting coefficients.

Alphas specifies the denominator coefficients of the transfer function. Refer to Appendix B, *Description of the Betas and Alphas Parameters*, for information on interpreting coefficients.

Error. See Appendix A, *Error Codes*, for a description of the error.

The zero-pole representation is given by

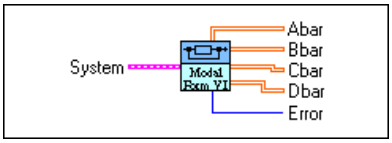
$$H(s) = \frac{\prod_{i=0}^{n-1} s - z_i}{\prod_{i=0}^{m-1} s - p_i}$$

where the gain constant is equal to one. The transfer-function representation is given by

$$H(s) = \frac{\beta_n s^n + \beta_{n-1} s^{n-1} + \ldots + \beta_0}{\alpha_m s^m + \alpha_{m-1} s^{m-1} + \ldots + \alpha_0}$$

GSim Modal Form

Converts the state-space representation to the modal-form representation.



System is the state-space representation. **System** contains the following four elements:

A is matrix **A** of the state-space representation.

B is matrix **B** of the state-space representation.

C is matrix **C** of the state-space representation.

D is matrix **D** of the state-space representation.

Abar is matrix **Abar** of the modal-form representation.

Bbar is matrix **Bbar** of the modal-form representation.

Cbar is matrix **Cbar** of the modal-form representation.

Dbar is matrix **Dbar** of the modal-form representation.

Error. See Appendix A, *Error Codes*, for a description of the error.

5

Feedback VIs

This chapter describes VIs you can use to design linear state feedback systems from state-space, transfer-function, and residual-pole representations.

The actual poles of the system are determined based on the given representation of the system. The position of these main poles determines many properties of the system. For example, if one of the poles of the system has non-negative real parts, the system is unstable. In many applications you need to move these poles to a specified location, and you can do this with linear state feedback.

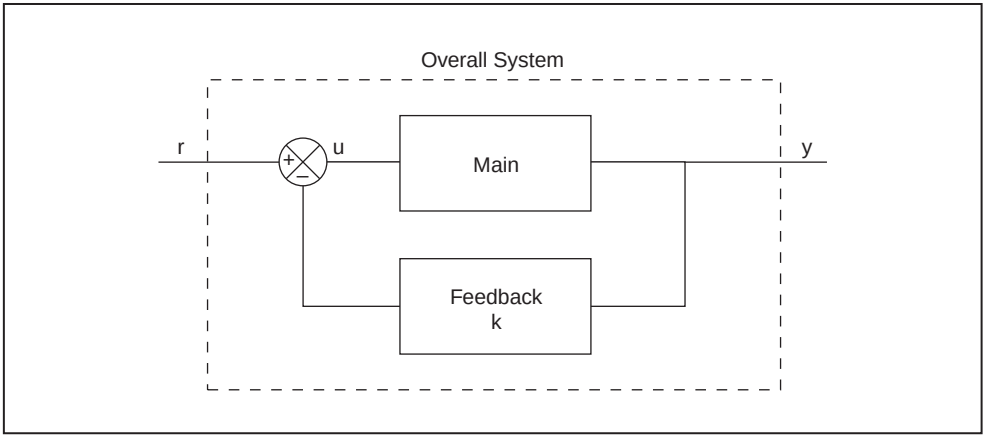


Figure 5-1. Closed-Loop Feedback Structure

All VIs in the `FEEDBACK.llb` library are for single-input, single-output case. Consider the state-space representation of a single-input, single-output, linear time-invariant system:

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}u \tag{5-1}$$

$$y = \mathbf{C}\mathbf{x} + \mathbf{D}u \tag{5-2}$$

where **x** is an *n*-by-1 state vector
u is scalar input
y is scalar output
A is an *n*-by-*n* real constant matrix
B is an *n*-by-1 real constant column vector
C is a 1-by-*n* real row vector
D is a scalar

In state feedback, every state variable is multiplied by a gain and fed back into the input terminal. If the gain between the ith state variable and the input is k_i, the state feedback vector is given by

$$\mathbf{k} = \begin{bmatrix} k_1 & k_2 & \ldots & k_n \end{bmatrix}$$

The state-space representation of the overall system is obtained by replacing the input *u* by $u = r - \mathbf{k}\mathbf{x}$, where *r* is the reference input:

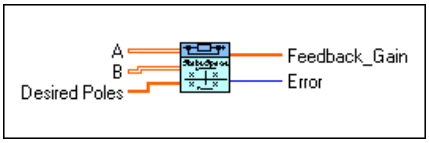
$$\dot{\mathbf{x}} = (\mathbf{A} - \mathbf{B}\mathbf{k})\mathbf{x} + \mathbf{B}r$$

$$y = (\mathbf{C} - \mathbf{D}\mathbf{k})\mathbf{x} + \mathbf{D}r$$

Notice that these equations have the same dimension and same state space as Equation (5-1) and Equation (5-2). If the main system is controllable and its complex conjugate eigenvalues appear in pair, then, by the use of state feedback $u = -\mathbf{k}\mathbf{x}$, the eigenvalues of the feedback connected system can be arbitrarily placed at the desired location on the zero-pole plot.

GSim Design from SS

Designs linear state feedback, given the state-space representation of the system.



[DBL] **A** is matrix **A** of the state-space representation of the open-loop system.

[DBL] **B** is matrix **B** of the state-space representation of the open-loop system.

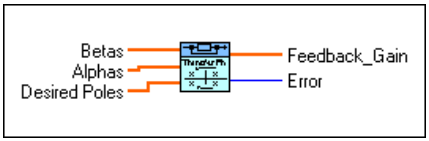
[CDB] **Desired Poles** specifies the desired pole locations in the closed-loop system.

[CDB] **Feedback_Gain** is the feedback gain needed for the desired pole placement.

I32 **Error**. See Appendix A, *Error Codes*, for a description of the error.

GSim Design from TF

Designs linear state feedback, given the transfer-function representation of the system.



[DBL] **Betas** specifies the numerator coefficients of the transfer-function of the open-loop system. Refer to Appendix B, *Description of the Betas and Alphas Parameters*, for information about entering coefficients.

[DBL] **Alphas** specifies the denominator coefficients of the transfer-function of the open-loop system. Refer to Appendix B, *Description of the Betas and Alphas Parameters*, for information about entering coefficients.

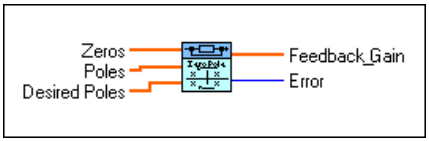
[CDB] **Desired Poles** specifies the desired pole locations in the closed-loop system.

[CDB] **Feedback_Gain** is the feedback gain needed for the desired pole placement.

[I32] **Error**. See Appendix A, *Error Codes*, for a description of the error.

GSim Design from ZP

Designs linear state feedback, given the zero-pole representation of the system.



[CDB] **Zeros** specifies the zeros of the open-loop system.

[CDB] **Poles** specifies the poles of the open-loop system.

[CDB] **Desired Poles** specifies the desired pole locations in the closed-loop system.

[CDB] **Feedback_Gain** is the feedback gain needed.

[I32] **Error**. See Appendix A, *Error Codes*, for a description of the error.

6

Frequency Response VIs

This chapter describes VIs you can use for the frequency domain analysis of linear, time-invariant control systems. The commonly used tools for frequency domain analysis are the Nyquist plot, the Bode plot, and the root-locus plot.

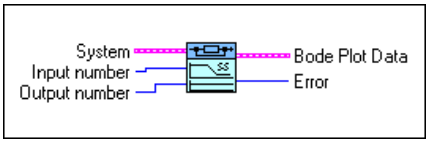
The Nyquist plot is a polar plot of the amplitude and phase of the open-loop transfer function $G(s)H(s)$ as $s = \sigma + j\omega$ traverses a contour that encloses the right-half plane. You often need to determine the stability of a system using the Nyquist stability criterion.

As *s* traverses the positive imaginary axis, it has the value of real frequency ω, and the plot corresponds to $G(j\omega)H(j\omega)$. In this case, two diagrams that both have ω as a common axis can convey the same amount of information as the Nyquist plot. These two diagrams usually are referred to as Bode diagrams. Because the Bode diagrams provide information about *s* only corresponding to the positive imaginary axis, they represent the frequency response.

The root-locus method is a technique for determining the roots of the closed-loop characteristic equation of a system as a function of the static gain. This method is based on the relationship that exists between the roots of the closed-loop transfer function and the poles and zeros of the open-loop transfer function. The root-locus method has several distinct advantages. You can accurately determine the relative stability of the control system using the knowledge about closed-loop roots that you obtain from the root-locus plot. Alternatively, users can apply this information to obtain approximate solutions to their control system problems. The VIs described in this chapter provide a graphical method of constructing the root-locus plot.

GSim Bode Plot from SS

Computes data for the Bode plot, given the state-space representation of a system.



System is the state-space representation of the system. **System** is a cluster containing the following four elements:

A is matrix **A** of the state-space representation.

B is matrix **B** of the state-space representation.

C is matrix **C** of the state-space representation.

D is matrix **D** of the state-space representation.

Input number is used with a multi-input system. This parameter defaults to `1`. Enter the default value for a single-input system.

Output number is used with a multi-output system. This parameter defaults to `1`. Enter the default value for a single-output system.

Bode Plot Data contains information for the Gain and Phase Bode diagram. This cluster contains the following three elements:

Gain specifies gain values in dB for different frequency values in the **Frequency** array.

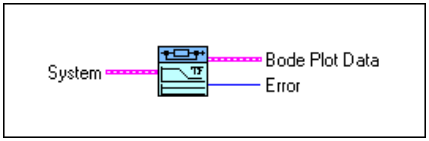
Phase specifies phase values in degrees for different frequency values in the **Frequency** array.

Frequency specifies frequency values for which the gain and phase are computed.

Error. See Appendix A, *Error Codes*, for a description of the error.

GSim Bode Plot from TF

Computes data for the Bode plot, given the transfer-function representation of a system.



System is the transfer-function representation of the system. **System** is a cluster containing the following two elements:

Betas specifies the numerator coefficients of the transfer function. Refer to Appendix B, *Description of the Betas and Alphas Parameters*, for information about entering coefficients.

Alphas specifies the denominator coefficients of the transfer function. Refer to Appendix B, *Description of the Betas and Alphas Parameters*, for information about entering coefficients.

Bode Plot Data contains information for the Gain and Phase Bode diagram. **Bode Plot Data** is a cluster containing three elements:

Gain specifies gain values in dB for different frequency values in the **Frequency** array.

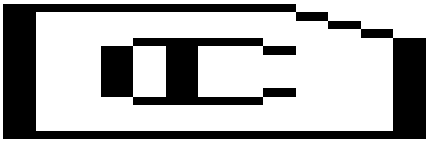
Phase specifies phase values in degrees for different frequency values in the **Frequency** array.

Frequency specifies frequency values for which the gain and phase are computed.

Error. See Appendix A, *Error Codes*, for a description of the error.

GSim Root Locus

Computes data for the root-locus plot, given the transfer-function representation of a system. The VI generates a text file that contains root-locus plot information, such as the number of branches in the root-locus plot, the start and end points for each branch, and the breakpoints.



System is the transfer-function representation of the system. **System** is a cluster containing the following two elements:

Betas specifies the numerator coefficients of the transfer function. Refer to Appendix B, *Description of the Betas and Alphas Parameters*, for information about entering coefficients.

Alphas specifies the denominator coefficients of the transfer function. Refer to Appendix B, *Description of the Betas and Alphas Parameters*, for information about entering coefficients.

Input Gain is the single gain value for which you want to calculate the value on the root-locus. If you want to calculate the data for the root-locus plot for different gain values, enter `0.0` (default value) and the VI internally generates the gain values.

Resolution is the resolution value used internally to compute the gain values. This value is used only if the **Input Gain** equals `0.0`. **Resolution** defaults to `0.10`.

Write to File is a Boolean that specifies whether to write to a file information related to the root-locus plot. If you set this input to TRUE (the default value), the VI opens a dialog box so you can select the file to which the information is written. If you set this input to FALSE, no information is written.

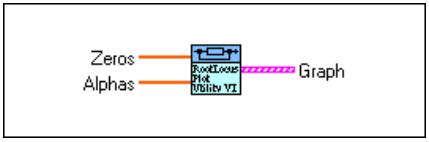
Zeros specifies the zeros of the closed-loop system for different gain values in the gain array.

Gain specifies gain values for which zeros are computed.

Error. See Appendix A, *Error Codes*, for a description of the error.

GSim Root Locus Plot Utility

Draws the root-locus plot. After computing the data for the root-locus plot using the GSim Root Locus VI, use this VI to plot the root-locus on an XY Graph.

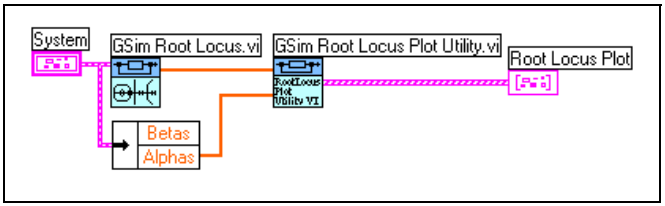


Zeros specifies the zeros output from the GSim Root Locus Plot VI.

Alphas specifies the denominator coefficients of the transfer function of the system. Refer to Appendix B, *Description of the Betas and Alphas Parameters*, for information about entering coefficients.

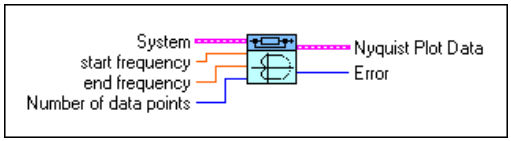
Graph is a graph output that you can directly connect to an XY Graph.

The following figure shows how to use the GSim Root Locus VI with the GSim Root Locus Plot Utility VI.



GSim Nyquist Plot

Computes data for the Nyquist plot, given the transfer-function representation of a system.



System is the transfer-function representation of the system. **System** is a cluster containing the following two elements:

Betas specifies the numerator coefficients of the transfer function. Refer to Appendix B, *Description of the Betas and Alphas Parameters*, for information about entering coefficients.

Alphas specifies the denominator coefficients of the transfer function. Refer to Appendix B, *Description of the Betas and Alphas Parameters*, for information about entering coefficients.

start frequency defaults to `0.0`. Use this parameter if you want to compute the Nyquist plot between two frequencies.

end frequency defaults to `0.0`. Use this parameter if you want to compute the Nyquist plot between two frequencies.

If both the start and end frequencies equal zero, the VI internally computes the frequency range based on the input **Number of data points**.

Number of data points is the number of data points for which you want the Nyquist plot data to be computed. **Number of data points** defaults to `800`.

Nyquist Plot Data contains information for the Nyquist plot. **Nyquist Plot Data** is a cluster containing the following two elements:

X_axis is data for the x-axis of the Nyquist plot (Real part of the Nyquist plot).

Y_axis is data for the y-axis of the Nyquist plot (Imaginary part of the Nyquist plot).

Error. See Appendix A, *Error Codes*, for a description of the error.

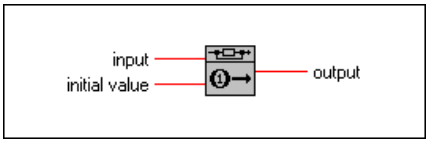
7

Connections

This chapter describes the VIs that allow multiplexing, realize triggers, and produce initial values explicitly.

GSim Initial Condition

Produces a predefined initial condition only during the first iteration of the control or simulation task.



input is the real valued current input.

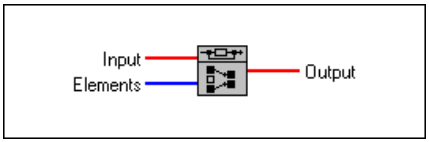
initial value is the predefined initial condition.

output is the real valued output.

During the first iteration, the **output** parameter of this VI is set to **initial value**. During subsequent iterations, **output** is set to **input**. In other words, this VI sets the **initial value** for a wire or connector. An alteration of **initial value** during runtime of the simulation does not affect the behavior of the system.

GSim Input Selector

Collects some special channels from a given list of channels.



Input is a one-dimensional array of input signals of any length.

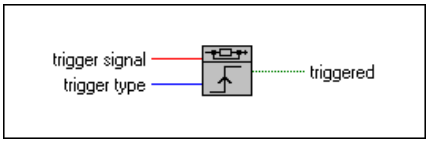
Elements is a one-dimensional array of non-negative integers specifying some indexes.

Output is a one-dimensional array of selected values.

This VI selects an array of all elements from a given array with indexes described by the array **Elements**. For Example, for **Input**={2.3,–1.9, .89, 11.9} and **Elements**={1,2} the result **Output**={–1.9, .89} is determined, where the element 2.3 occupies the index 0 of the input array.

GSim Trigger

Detects changes of the input signal during the control or simulation process.



trigger signal is the real valued input.

trigger type consists of three valid choices: 0 (rising), 1 (falling), or 2 (either).

triggered is a boolean value that is TRUE if and only if the trigger event has occurred.

If *trigger signal*$_n$ is the new input signal and *trigger signal*$_{n-1}$ is the last input signal, the following conditions apply:

$$trigger\ type = \begin{cases} rising & triggered \text{ is TRUE if } trigger\ signal_n > trigger\ signal_{n-1} \\ falling & triggered \text{ is TRUE if } trigger\ signal_n < trigger\ signal_{n-1} \\ either & triggered \text{ is TRUE if } trigger\ signal_n \neq trigger\ signal_{n-1} \end{cases}$$

8

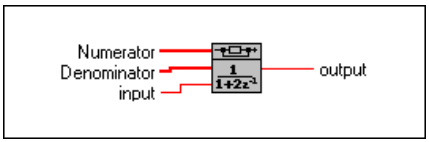
Discrete Systems

This chapter describes the routines in GSIM that handle discrete-time linear systems.

You can represent a linear, time-invariant discrete system in a state-space representation, transfer-function representation, or zero-pole representation. The integrator is a special transfer function. You can implement three different integration methods: backward, forward, and trapezoidal.

GSim Discrete Filter

Realizes an IIR (infinite impulse response) filter with forward and backward coefficients defined by numerator and denominator polynomials. The GSim Discrete Filter VI is based completely on the GSim Discrete Transfer Function VI in this control and simulation software.



Numerator is a one-dimensional array containing the coefficients of the numerator polynomial in ascending powers of *z*.

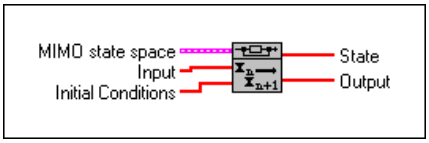
Denominator is a one-dimensional array containing the coefficients of the denominator polynomial in ascending powers of *z*. The order of the **Denominator** must be greater than or equal to the order of the **Numerator**, or you receive an error. See Appendix A, *Error Codes*, for a list of errors and their descriptions.

input is a real valued input.

output is a real valued output.

GSim Discrete State-Space

Calculates a set of new values in **Output**, where both **Input** and **Initial Conditions** are given.



MIMO state space (multiple-input, multiple-output) is a cluster consisting of four different matrices describing the state-space representation of the MIMO system.

A is the system matrix of the given MIMO system.

B is the input matrix of the given MIMO system.

C is the output matrix of the given MIMO system.

D is the straight-way matrix of the given MIMO system.

Internally, a MIMO test verifies that **MIMO state space** is a valid MIMO system.

Input is a one-dimensional array of input signals (see function *u* in the description). The VI verifies that the dimension of **Input** is valid.

Initial Conditions is a one-dimensional array of initial conditions. The VI verifies that the dimension of **Input** is valid. **Input** and **Initial Conditions** must have the same dimensions.

State is a one-dimensional array specifying the internal state.

Output is a one-dimensional array of calculated values.

The following equations give the state-space description of a MIMO system

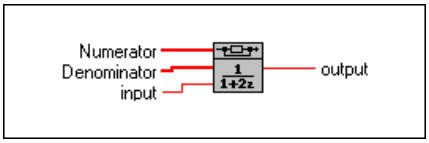
$$\mathbf{x}_{n+1} = \mathbf{A}\mathbf{x}_n + \mathbf{B}\mathbf{u}_n$$
$$\mathbf{y}_n = \mathbf{C}\mathbf{x}_n + \mathbf{D}\mathbf{u}_n$$

where **u** is input
y is output
x is the state
A is an n_x by n_x matrix
B is an n_x by n_u matrix
C is an n_y by n_x matrix
D is an n_y by n_u matrix

Furthermore, the exact initial conditions for **x** are known. Internally, a one-step calculation is performed during each While Loop.

GSim Discrete Transfer Function

Calculates a new value **output**, where both **input** and a transfer function are given. The SISO (single-input, single-output) transfer function consists of a numerator polynomial and a denominator polynomial.



Numerator is a one-dimensional array containing the coefficients of the numerator polynomial in ascending powers of *z*.

Denominator is a one-dimensional array containing the coefficients of the denominator polynomial in ascending powers of *z*. The order of the **Denominator** must be greater than or equal to the order of the **Numerator**, or you receive an error. See Appendix A, *Error Codes*, for a list of errors and their descriptions.

input is a real valued input.

output is a real valued output.

The SISO form of a discrete transfer function has the following structure:

$$H(z) = \frac{Numerator\ (z)}{Denominator\ (z)}$$

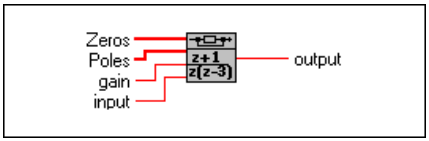
where *Numerator* (*z*) and *Denominator* (*z*) are polynomials that completely describe the SISO (single-input, single-output) system. An equivalent representation is given by

$$H(z) = \frac{a_n z^n + a_{n-1} z^{n-1} + \ldots + a_0 z^{n-m}}{b_n z^n + b_{n-1} z^{n-1} + \ldots + b_0}$$

This VI uses the GSim Discrete Filter VI to calculate the discrete transfer function.

GSim Discrete Zero-Pole

Calculates a new value **output**, where **input** and the **Zeros** and **Poles** of a transfer function are given. The SISO (single-input, single-output) transfer function consists of a numerator polynomial and a denominator polynomial. This VI solves a linear difference equation.



Zeros is a one-dimensional complex array representing the numerator polynomial of the transfer function of the given system. For all zeros with an imaginary part, the conjugate complex number also must belong to this array. If it does not, an error occurs. See Appendix A, *Error Codes*, for a list of errors and their descriptions.

Poles is a one-dimensional complex array representing the denominator polynomial of the transfer function of the given system. For all poles with an imaginary part, the conjugate complex number also must belong to this array. If it does not, an error occurs. See Appendix A, *Error Codes*, for a list of errors and their descriptions.

gain is a real value representing the common gain of both polynomials.

input is a real valued input.

output is a real valued output.

The SISO form of a transfer function of a discrete system has the following structure:

$$H(z) = \frac{Numerator\ (z)}{Denominator(z)}$$

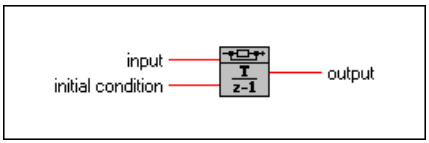
where *Numerator* (z) and *Denominator* (z) are polynomials that completely describe the SISO (single-input, single-output) discrete system in terms of the delay operator z. An equivalent representation is given by the following factored form:

$$H(z) = K\frac{(z-z_1)\ldots(z-z_n)}{(z-p_1)\ldots(z-p_m)}$$

where z_1, ..., z_n represent the zeros, p_1, ..., p_m represent the poles, and K denotes the gain.

GSim Discrete-Time Integrator

Integrates an input signal.



input is a real valued input.

initial condition specifies the initial output of the integrator.

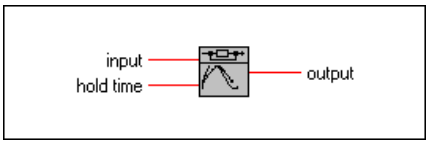
output is the real valued output consisting of the integral of **input**.

At the start of the control or simulation task, **output** is equal to **initial condition** regardless of the value of the current **input**. For all other loops, the results of **output** depend on your choice of integrator: forward, backward, or trapezoidal.

This choice is part of the initialization procedure based on the GSim Initialize VI. Because there is not a general rule for selecting the best method, check all three methods to determine which is the best for a specific problem.

GSim First-Order Hold

Realizes a first-order sample and hold.



input is the current real valued input.

hold time represents a period of time during which the output is first-order holding (sample and hold).

output is the calculated output value.

Figures 8-1 and 8-2 depict the difference between a zero-order and a first-order sample and hold. Both examples are based on a sine signal where the sampling rate is 0.1 and **hold time** is 0.6.

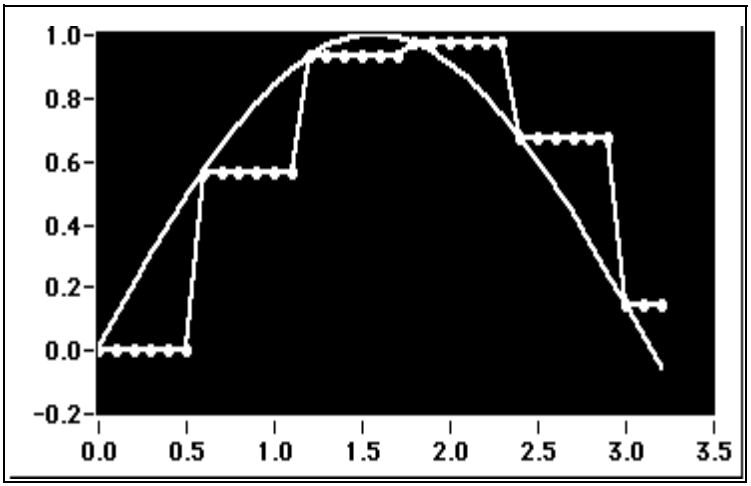


Figure 8-1. Zero-Order Sample and Hold of a Sine Signal

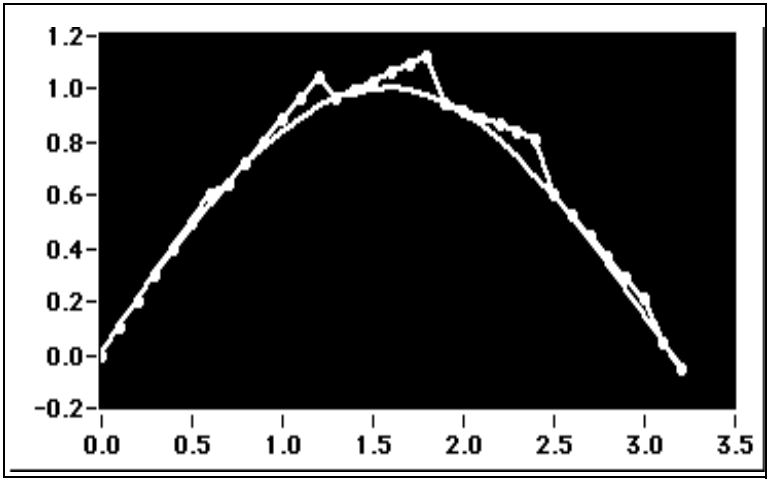
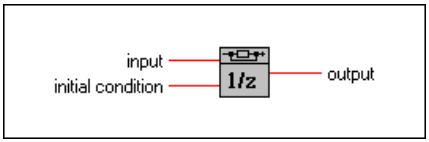


Figure 8-2. First-Order Sample and Hold of a Sine Signal

The zero-order sample and hold retains the current value during the entire **hold time**, which results in a step function like signal. The first-order sample and hold retains the derivative of the function during the entire **hold time**. Usually, this signal is a much better approximation of the given signal. On the other hand, zero-order sampling and hold elements are more popular.

GSim Unit Delay

Delays the input exactly one time step.



input is the current real valued input.

initial condition represents the initial value.

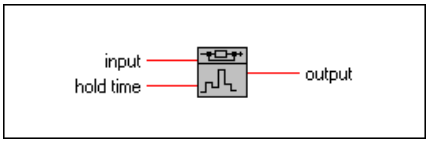
output is the calculated output value:

$$output_0 = initial\ condition$$

$$output_n = input_{n-1} \quad \text{for all } n > 0$$

GSim Zero-Order Hold

Holds the signal value for a definite period of time.



input is the current real valued input.

hold time represents a period of time during which the output is holding (sample and hold).

output is the calculated output value:

$$output_n = \begin{cases} input_0 & \text{if } n \times dt \le hold\ time \\ input_m & \text{if } last\ sample\ time < m \times dt \le last\ sample\ time + hold\ time, \\ & \text{for all } n > 0 \end{cases}$$

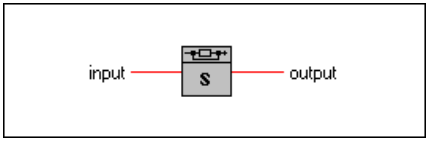
9

Linear Systems

This chapter describes the routines that handle linear continuous systems, which can be represented as state-space, transfer-function, or zero-pole representations and implemented with three different integration methods: Euler, Adams, or Runge-Kutta. The integrator is a special transfer function.

GSim Derivative

Calculates the derivative of a signal.



DBL **input** in is a real input signal.

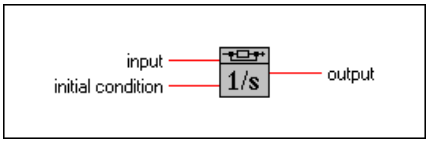
DBL **output** is the real output signal consisting of the derivative of **input**.

At the start (first iteration) of the control or simulation task, **output** equals 0 regardless of the current **input** value. For all other iterations, the following formula applies:

$$output_n = \frac{input_n - input_{n-1}}{dt}$$

GSim Integrator

Integrates an input signal.



input is a real input signal.

initial condition is the initial output of the integrator.

output is the real output signal consisting of the integral of **input**.

At the start of the control or simulation task, **output** equals **initial condition** regardless of the current **input** value. For all subsequent loops, the results depend on the chosen integrator: Euler, Adams, or Runge-Kutta. You choose an integrator as part of the initialization procedure based on the GSim Initialize VI. Because there is not a general rule for selecting the best method, check all three methods to determine which is the best for a specific problem. In general, the Euler method is fast but only applicable to relatively simple systems. You can use the Euler method for many control tasks, but base complicated simulations on the Adams or Runge-Kutta method. Figure 9-1 illustrates a typical integration process, where you choose the integrator with the GSim Initialize VI to the left of the diagram.

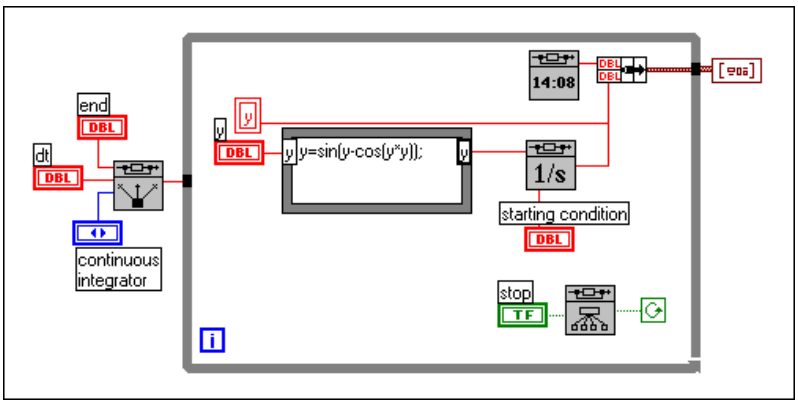
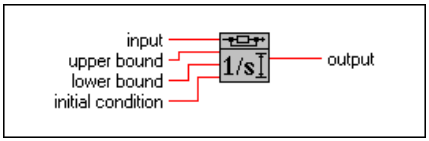


Figure 9-1. A Typical Integration Process with GSim Integrator

GSim Limited Integrator

Integrates an input signal where some restrictions apply.



input is a real input signal.

upper bound is the upper limit of the output.

lower bound is the lower bound of the output.

initial condition is the initial output of the integrator.

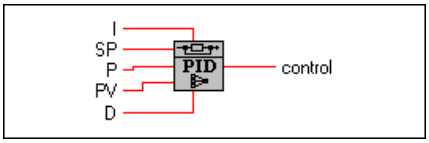
output is the real output signal consisting of the integral of **input**.

This VI is based on the GSim Integrator VI. The **output** value is defined as follows:

$$output = \begin{cases} lower\ bound & \text{if } input < lower\ bound \\ & \quad \text{and } input \text{ is negative} \\ upper\ bound & \text{if } input > upper\ bound \\ & \quad \text{and } input \text{ is positive} \\ input & \text{otherwise} \end{cases}$$

GSim PID Parallel

Realizes a PID controller in parallel notation.



I is the I-part of the PID controller. It must be a non-negative real number.

SP is the current setpoint of the system under control.

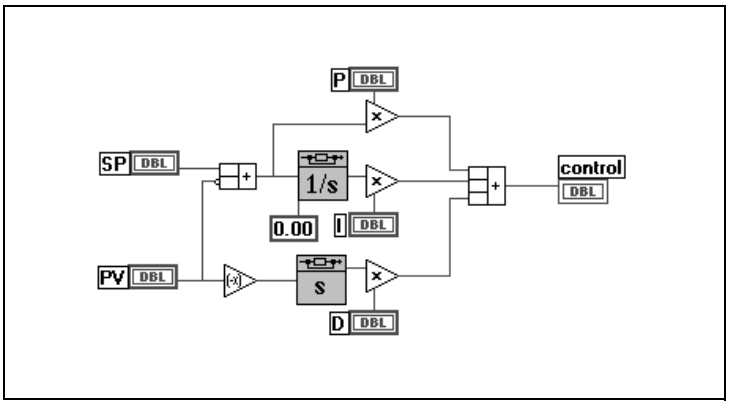
P is the P-part of the PID controller. It must be a non-negative real number.

PV is the current value of the process variable.

D is the D-part of the PID controller. It must be a non-negative real number.

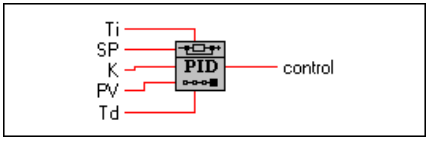
control is the output of the PID controller at the given setpoint and value of process variable, where the parameters of the PID controller are **P**, **I**, and **D**.

This VI realizes the classical PID controller according to the following scheme:



GSim PID Serial

Realizes a PID controller in serial notation.



Ti is the dominant time constant of the PID controller, which must be a non-negative, real number.

SP is the current setpoint of the system under control.

K is the static gain of the PID controller, which must be a non-negative, real number.

PV is the current value of the process variable.

Td is the dominant deadtime of the PID controller, which must be a non-negative, real number.

control is the output of the PID controller at a given setpoint and value of process variable, where the parameters of the PID controller are **Ti**, **K**, and **Td**.

This VI is based on the GSim PID Parallel VI. The following formulas are used:

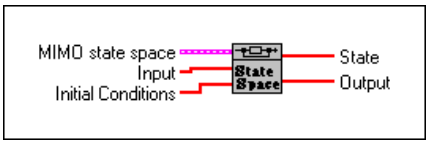
$$P = K\left(\frac{1 + T_d}{T_i}\right)$$

$$I = \frac{K}{T_i}$$

$$D = K \times T_d$$

GSim State-Space

Calculates a set of new **Output** values, where both **Input** and **Initial Conditions** are given.



MIMO state space is a cluster containing four different matrices that describe the state-space representation of the MIMO (multiple-input, multiple-output) system.

A is the system matrix of the given MIMO system.

B is the input matrix of the given MIMO system.

C is the output matrix of the given MIMO system.

D is the straight-way matrix of the given MIMO system.

Internally a MIMO test checks whether the **MIMO state space** is a valid MIMO system. An invalid MIMO system results in an error.

Input is a one-dimensional array of input signals (see the following state-space description of a MIMO system and the function of **u**). The VI verifies that the dimension of **Input** is valid.

Initial Conditions is a one-dimensional array of initial conditions. The VI verifies that the dimension of **Input** is valid. **Input** and **Initial Conditions** must have the same dimension.

State is a one-dimensional array specifying the internal state.

Output is a one-dimensional array of calculated values.

The following equations give the state-space description of a MIMO system:

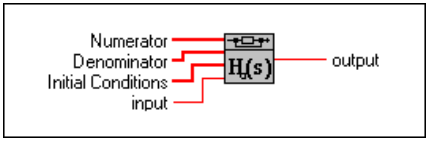
$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t)$$
$$\mathbf{y}(t) = \mathbf{C}\mathbf{x}(t) + \mathbf{D}\mathbf{u}(t)$$

where **u** is input
y is output
x is the state
A is an n_x by n_x matrix
B is an n_x by n_u matrix
C is an n_y by n_x matrix
D is an n_y by n_u matrix

Furthermore, the exact initial conditions for **x** are known. Internally, a one-step calculation is performed during each While Loop.

GSim Transfer Function IC

Calculates a new value **output**, where **input**, **Initial Conditions**, and a transfer function are given. The SISO transfer function consists of a numerator polynomial and a denominator polynomial.



Numerator is a one-dimensional array representing a polynomial in ascending powers of *s*.

Denominator is a one-dimensional array representing the denominator polynomial in ascending powers of *s*.

Initial Conditions is an one-dimensional array representing the initial conditions of the underlying differential equation.

input is a real input signal.

output is a real output signal.

The SISO form of a transfer function has the following structure:

$$H(s) = \frac{Numerator\ (s)}{Denominator\ (s)}$$

where *Numerator* (*s*) and *Denominator* (*s*) are polynomials that completely describe the SISO (single-input, single-output) system. An equivalent representation is given by

$$H(s) = K\frac{(s-z_1)\ldots(s-z_n)}{(s-p_1)\ldots(s-p_m)}$$

where z_1, ..., z_n represent the zeros, and p_1, ..., p_m represent the poles of the system.

This VI solves the linear differential equation

$$a_m y^{(m)} + a_{m-1} y^{(m-1)} + \ldots + a_1 y^{(1)} + a_0 y = b_n x^{(n)} + b_{n-1} x^{(n-1)} + \ldots + b_1 x^{(1)} + b_0 x$$

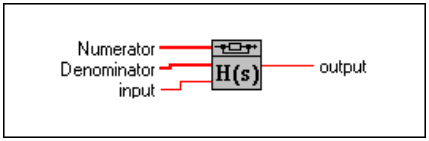
$$y^{(m-1)}(0) = ic_{m-1}, y^{(m-2)}(0) = ic_{m-2}, \ldots, y^{(1)}(0) = ic_1, y(0) = ic_0$$

where x is the given input and the components of *ic* stand for the array of **Initial Conditions**.

Note ***This VI deletes leading zeros in the representation of the polynomial. For example,*** $1 + 2s - s^2 + 0s^3$ ***reduces to*** $1 + 2s - s^2$.

GSim Transfer Function

Calculates a new value **output**, where both **input** and a transfer function are given. The SISO (single-input, single-output) transfer function consists of a numerator polynomial and a denominator polynomial.



Numerator is a one-dimensional array representing a polynomial.

Denominator is a one-dimensional array representing the denominator polynomial.

input is a real input signal.

output is a real output signal.

The SISO form of a transfer function has the following structure:

$$H(s) = \frac{Numerator\ (s)}{Denominator\ (s)}$$

where *Numerator* (*s*) and *Denominator* (*s*) are polynomials that completely describe the SISO (single-input, single-output) system. An equivalent representation is given by

$$H(s) = K\frac{(s-z_1)\ldots(s-z_n)}{(s-p_1)\ldots(s-p_m)}$$

This VI solves the linear differential equation

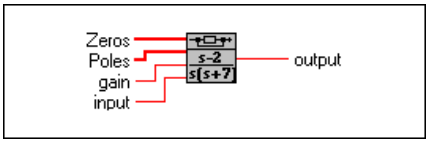
$$a_m y^{(m)} + a_{m-1}y^{(m-1)} + \ldots + a_1 y^{(1)} + a_0 y = b_n x^{(n)} + b_{n-1}x^{(n-1)} + \ldots + b_1 x^{(1)} + b_0 x$$
$$y^{(m-1)}(0) = 0, y^{(m-2)}(0) = 0, \ldots, y^{(1)}(0) = 0, y(0) = 0$$

where *x* is the given input.

Note ***This VI deletes leading zeros in the representation of the polynomial. For example,*** $1 + 2s - s^2 + 0s^3$ ***reduces to*** $1 + 2s - s^2$**.**

GSim Zero-Pole

Calculates a new value **output**, where **input**, **Zeros**, and **Poles** of a transfer function are given. The SISO transfer function consists of a numerator polynomial and a denominator polynomial.



Zeros is a one-dimensional complex array representing the numerator polynomial of the transfer function of the given system. For all zeros with an imaginary part, the conjugate complex number also must belong to this array. If it does not, an error occurs. See Appendix A, *Error Codes*, for a list of errors and their descriptions.

Poles is a one-dimensional complex array representing the denominator polynomial of the transfer function of the given system. For all poles with an imaginary part, the conjugate complex number also must belong to this array. If it does not, an error occurs. See Appendix A, *Error Codes*, for a list of errors and their descriptions.

gain is a real value representing the common gain of both polynomials.

input is a real input signal.

output is a real output signal.

The SISO form of a transfer function has the following structure:

$$H(s) = \frac{Numerator\ (s)}{Denominator\ (s)}$$

where *Numerator* (*s*) and *Denominator* (*s*) are polynomials that completely describe the SISO (single-input, single-output) system. An equivalent representation is given by

$$H(s) = K\frac{(s-z_1)\ldots(s-z_n)}{(s-p_1)\ldots(s-p_m)}$$

where z_1, ..., z_n represent the zeros, and p_1, ..., p_m represent the poles, and K denotes the gain.

This VI solves the linear differential equation

$$a_m y^{(m)} + a_{m-1} y^{(m-1)} + \ldots + a_1 y^{(1)} + a_0 y = b_n x^{(n)} + b_{n-1} x^{(n-1)} + \ldots + b_1 x^{(1)} + b_0 x$$

$$y^{(m-1)}(0) = 0, y^{(m-2)}(0) = 0, \ldots, y^{(1)}(0) = 0, y(0) = 0$$

where x is the given input. The coefficients a and b are real values.

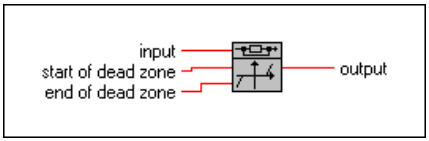
10

Nonlinear Systems

This chapter describes the nonlinear elements you can combine with other routines to build complex nonlinear systems.

GSim Dead Zone

Realizes a region of zero output.



input is the real valued current input.

start of dead zone is the lower limit of the dead zone.

end of dead zone is the upper limit of the dead zone.

output is a real number representing the output.

$$output = \begin{cases} 0 & \text{if } start\ of\ dead\ zone < input < end\ of\ dead\ zone \\ input - start\ of\ dead\ zone & \text{if } start\ of\ dead\ zone \geq input \\ input - end\ of\ dead\ zone & \text{if } end\ of\ dead\ zone \leq input \end{cases}$$

The bold lines in Figure 10-1 represent the output of GSim Dead Zone.

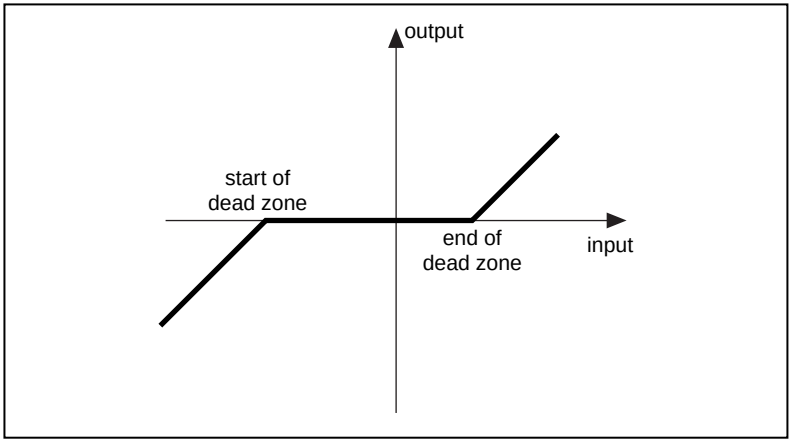
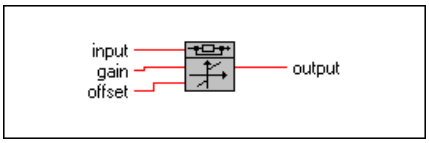


Figure 10-1. Output of the GSim Dead Zone VI

GSim Friction

Realizes a discontinuity at zero but guarantees the linear behavior elsewhere.



DBL **input** is the real-valued current input.

DBL **gain** is the slew rate of the linear function.

DBL **offset** is the offset of the linear function.

DBL **output** is a real number representing the output. The **output** value of this VI is calculated according to the following formula:

$$output = \text{sign}(input) \times (gain \times \text{abs}(input) + offset)$$

The bold lines in Figure 10-2 represent the output of GSim Friction.

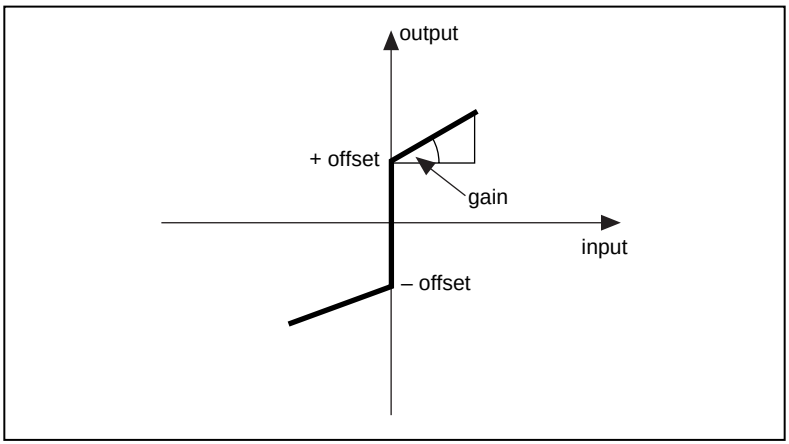
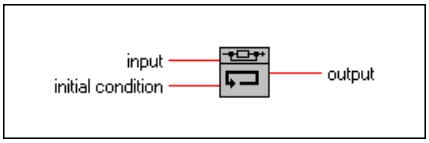


Figure 10-2. Output of the GSim Friction VI

GSim Memory

Stores the input value of the last control or simulation loop.



input is the real-valued current input.

initial condition is the predefined initial condition.

output is the real valued output.

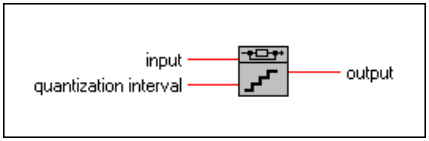
The value **output** of this VI depends on the index counter of the underlying While Loop. At the start of the simulation, **output** equals **initial condition**. For any other index, the following relationship applies:

$$output_n = input_{n-1}$$

Here $input_n$ denotes the current input value and $input_{n-1}$ the predecessor of this value. In other words, this VI stores the last **input** value. An alteration of **initial condition** during runtime of the calculation does not influence the behavior of the system.

GSim Quantizer

Quantizes an input signal.



input is the real-valued current input.

quantization interval is a positive number that describes the quantization steps. If **quantization interval** is less than or equal to 0, an error generates and execution stops.

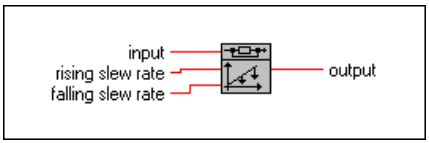
output is a real number representing the output. The output value of this VI is calculated by the following formula:

$$output = \left(\frac{input}{quantization\ interval}\right) \times quantization\ interval$$

Only discrete points are produced; all values must be multiples of the **quantization interval**.

GSim Rate Limiter

Limits the rate of change of a given signal.



input is the real-valued current input.

rising slew rate is the upper limit of the derivative calculated by two consecutive inputs.

falling slew rate is the lower limit of the derivative calculated by two consecutive inputs.

output is a real number representing the output.

The VI calculates the following derivative based on two consecutive inputs:

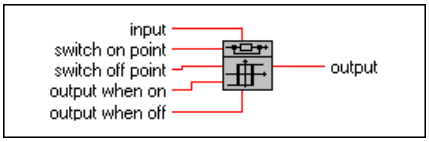
$$D = \frac{input_n - input_{n-1}}{dt}$$

Depending on the value of the ratio *D*, **output** is generated according to the following formulas:

$$output = \begin{cases} rising\ slew\ rate \times dt + output_{n-1} & \text{if } D > rising\ slew\ rate \\ falling\ slew\ rate \times dt + output_{n-1} & \text{if } D < falling\ slew\ rate \\ input_n - input_{n-1} + output_{n-1} & \text{otherwise} \end{cases}$$

GSim Relay

Switches between two constants. When the relay is on, it remains on until **input** drops below the value of **switch off point**. When the relay is off, it remains off until **input** exceeds the value of **switch on point**. The relay is predefined in the on state.



DBL **input** is the real-valued current input value.

DBL **switch on point** is the first threshold of the relay.

DBL **switch off point** is the other threshold of the relay. The **switch on point** must be greater than or equal to the **switch off point**. If not, an error occurs and the calculation stops immediately. See Appendix A, *Error Codes*, for a list of errors and their descriptions.

DBL **output when on** is the output when the relay is in the on state.

DBL **output when off** is the output when the relay is in the off state.

DBL **output** is a real number representing the output. The bold lines in Figure 10-3 represent the output of GSim Relay.

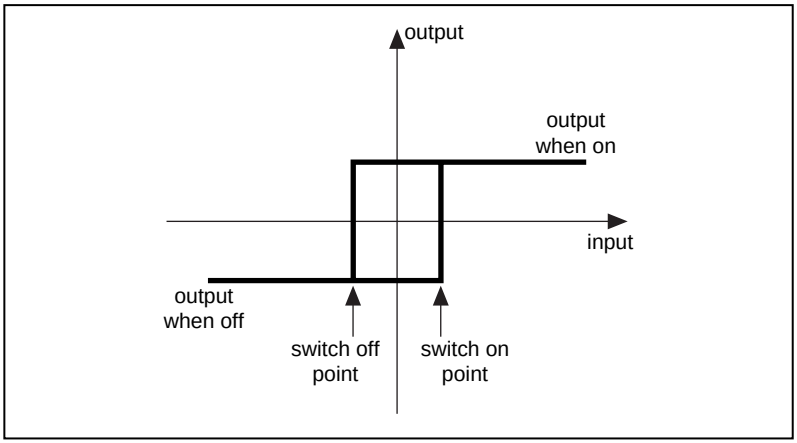
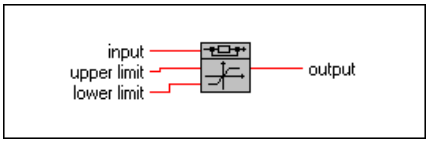


Figure 10-3. Output of the GSim Relay VI

GSim Saturation

Limits the valid range of a signal.



DBL **input** is the real-valued current input.

DBL **upper limit** is the upper limit of the valid region of the signal.

DBL **lower limit** is the lower limit of the valid region of the signal. The **upper limit** must be greater than or equal to the **lower limit**. A violation of this condition results in an error. See Appendix A, *Error Codes*, for a list of errors and their descriptions.

DBL **output** is a real number representing the output. The **output** value of this VI is calculated according to the following formulas:

$$output = \begin{cases} input & \text{if } upper\ limit \geq input > lower\ limit \\ upper\ limit & \text{if } input > upper\ limit \\ lower\ limit & \text{if } input \leq lower\ limit \end{cases}$$

The bold lines in Figure 10-4 represent the output of GSim Saturation.

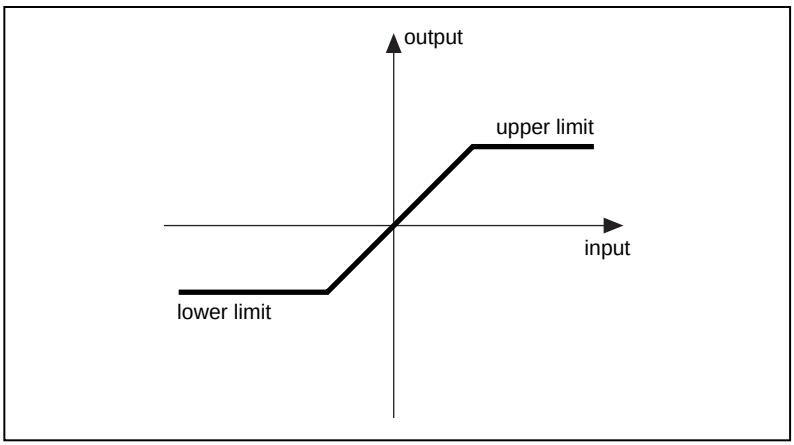
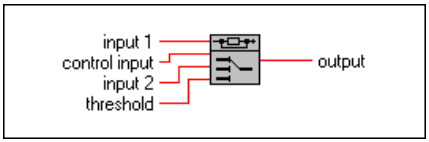


Figure 10-4. Output of the GSim Saturation VI

GSim Switch

Switches between two values.



input 1 is the first potential output value of the switcher.

control input is the input value.

input 2 is the second potential output value of the switcher.

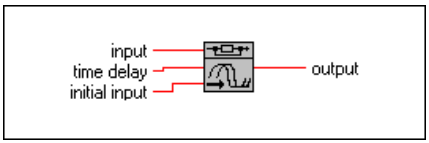
threshold denotes the switch position.

output is a real number representing the output. The **output** value of this VI is calculated according to the following formula:

$$output = \begin{cases} input\ 1 & \text{if } control\ input > threshold \\ input\ 2 & \text{otherwise} \end{cases}$$

GSim Transport Delay

Stores input values with an arbitrarily long memory.



input is the real-valued current input.

time delay is the delay in the timing structure. The VI calculates an appropriate multiple of *dt* to realize the transport delay. **time delay** must be greater than 0. A violation of this conditions leads to an error. See Appendix A, *Error Codes*, for a list of errors and their descriptions.

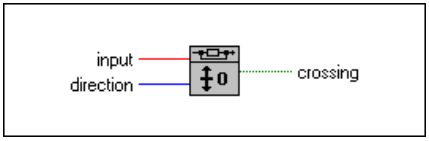
initial input is the output of this VI during the initial phase.

output is a real number representing the output. The **output** value of this VI is calculated according to the following formulas, where *dt* is the step rate initialized by the GSim Initialize VI:

$$output_n = \begin{cases} initial\ input & \text{if } \dfrac{time\ delay}{dt} > n \\ input_{n - \frac{time\ delay}{dt}} & \text{otherwise} \end{cases}$$

GSim Zero Crossing

Detects zero crossings of the input signal.



DBL **input** is the real-valued current input.

U16 **direction** consists of three valid choices: 0 (either), 1 (minus-plus), or 2 (plus-minus).

TF **crossing** is a Boolean value that is TRUE if and only if a zero crossing event has occurred.

If $input_n$ is the new input signal and $input_{n-1}$ is the previous input signal, the following cases exist when **crossing** is TRUE:

$$\text{case } direction = \begin{cases} either & \text{if } input_n \times input_{n-1} < 0 \\ minus-plus & \text{if } input_n > 0 \text{ and } input_{n-1} < 0 \\ plus-minus & \text{if } input_n < 0 \text{ and } input_{n-1} > 0 \end{cases}$$

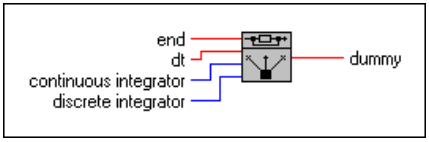
11

Shared VIs

This chapter describes the VIs that form the backbone of all control and simulation tasks based on this control and simulation software. These VIs organize and control the data flow of all calculations.

GSim Initialize

Prepares the simulation through a collection of parameters. You must include GSim Initialize in every simulation and control task.



end is the endtime of the control or simulation task. The start value is predefined as $t=0$. A non-positive **end** results in an error. See Appendix A, *Error Codes*, for a list of errors and their descriptions.

dt is the fixed step rate of the simulation. A non-positive **dt** results in an error. See Appendix A, *Error Codes*, for a list of errors and their descriptions.

continuous integrator is the method chosen for integrating the underlying differential equations. Valid values include Euler, Adams, and RK fixed (Runge-Kutta with fixed step rates).

discrete integrator is the chosen method integrating the underlying difference equations. Valid values include forward, backward, and trapezoidal.

dummy is the dummy output of this VI. You must wire this connector to the While Loop representing the control or simulation task. By ensuring that this VI is executed before entering the While Loop, you guarantee the correct order of executions.

You can determine all parameters of the control or simulation task with this VI. You must execute this VI at the start of a simulation. Figure 11-1 illustrates the correct use of this VI. There is a hidden connection between this VI and the global variable `GSIM Global.vi`,

which is part of `SHARE.llb`. Because many other VIs use this global variable at the same time, do not alter the structure of it or open it, even though opening `GSIM Global.vi` only slows the calculation.

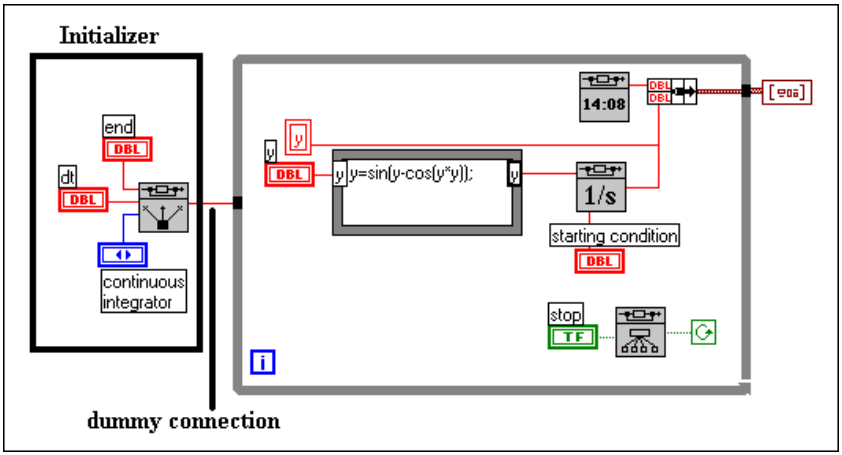
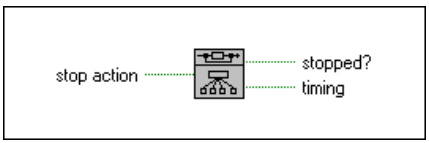


Figure 11-1. Include GSim Initializer in All Control and Simulation Tasks

GSim Manager

Controls the whole simulation and manages error handling.



stop action is usually connected to a user defined Boolean (stop button) to allow the user to stop the simulation. The simulation stops immediately after this parameter becomes TRUE, provided you choose the correct construction of the control or simulation task.

stopped? is a Boolean output that is TRUE if and only if the stop action is TRUE, the simulation is finished, or an error occurred.

timing is TRUE if the simulation is not fast enough to guarantee realtime behavior; that is, the **dt** of the system global (usually defined by the user) is too small.

As shown in Figure 11-2, the GSim Manager VI is an essential part of any simulation. This VI manages most activities of all implemented VIs. During the execution of every While Loop of a control or simulation task, the GSim Manager executes before all other routines. It is important that you adhere to the following rules. There is no automatic detection for the violation of the following conditions.

- Connect the **stopped?** Boolean directly to the loop controller of the underlying While Loop.
- If a **stop action** button on the front panel is part of the solution, connect the terminal corresponding to this button on the block diagram directly to the GSim Manager VI.
- Never use the GSim Manager VI as part of another data flow structure (with the exception of the underlying While Loop). For example, avoid the construction of sequences containing the GSim Manager VI.
- Only one GSim Manager can be part of a control or simulation loop.

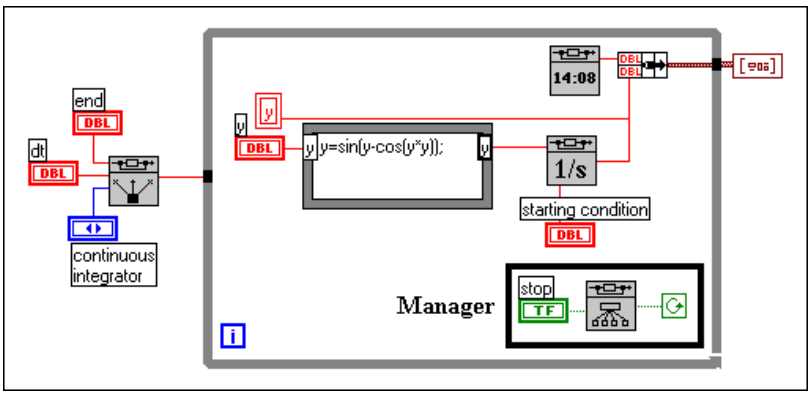


Figure 11-2. GSim Manager VI Is Essential to All Simulations

GSim Synchronizer

Synchronizes the execution order of all control and simulation VIs.

event is a TRUE Boolean output if and only if the iteration counter of the control or simulation task is exactly 0 or the VI containing this GSim Synchronizer VI is executed for the first time during the control or simulation task under consideration.

This VI is part of all other GSIM VIs and guarantees the correct order of all calculations during runtime of the control or simulation task. If you want to build an additional VI and incorporate it into this control and simulation software, you must include GSim Synchronizer as part of the solution. The correct structure of such a VI is shown in Figure 11-3.

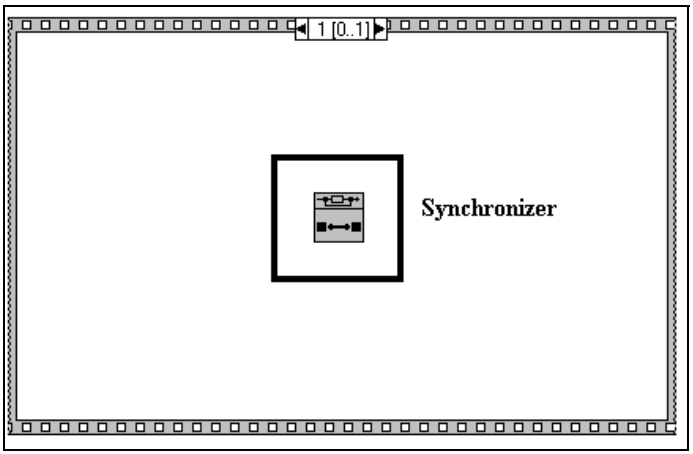


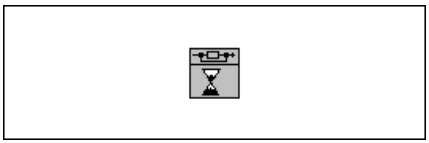
Figure 11-3. GSim Synchronizer in the Zeroth Frame

All GSIM VIs are organized as sequences. The zeroth frame consists of the synchronizer. You can use all other frames to realize the functionality of the VI under development.

You might use GSim Synchronizer to construct new elements (linear, discrete, or nonlinear). In all other cases, you can use the common programming techniques of G. In particular, you do not need to explicitly implement GSim Synchronizer if you only want to add, subtract, multiply, or divide some results.

GSim Timer

Synchronizes the time behavior of a control or simulation task.



Use this VI if one or more of the following VIs are part of a diagram: GSim Data In, GSim Collect Data In, or GSim Data Out. The GSim Timer VI guarantees a precise waiting period defined by the global variable **dt** between consecutive loops of the control or simulation task.

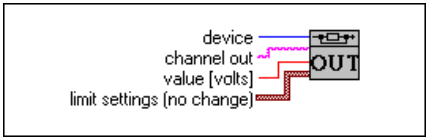
12

Sinks

This chapter describes the three VIs in the `SINKS.llb` library. The first one enables the output of a voltage using DAQ boards. The second VI simplifies the output of graphs produced by a control or simulation task. The third VI explicitly stops a task.

GSim Data Out

Outputs a voltage using a DAQ board.



device is the number of the plug-in data acquisition board. You must specify **device**.

channel out is a string that specifies the analog output channel.

value [**volts**] is the data to be written to the DAQ board.

limit settings [**no change**] is an array of clusters. Only the first array element is important (the array structure was chosen to maintain the general structure of DAQ operations). The default for **limit settings** [**no change**] is an empty array, which means the limits do not change from their original default.

upper limit is equal to your reference voltage and is the maximum voltage the DAQ board can produce.

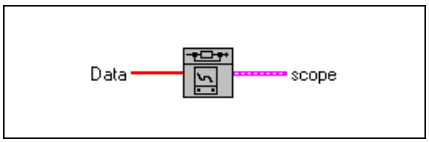
lower limit is either 0.0 V (which implies a unipolar setting) or a value equal (but opposite in sign) to the upper limit (bipolar).

reference source can have the following settings:
0: Do not change the reference source setting (default input)
1: Internal (default setting)
2: External

For more details on limit settings, devices, and channels, consult the *LabVIEW Function and VI Reference Manual*.

GSim Scope 1D

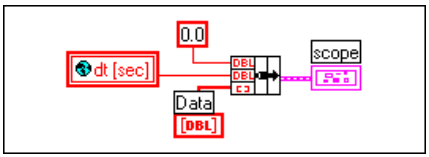
Simplifies the output of a graph determined by a control or simulation task.



Data is a one-dimensional array of real values determined at the time steps 0, 1dt, 2dt, ..., *n*dt where *dt* is the step size or time increment.

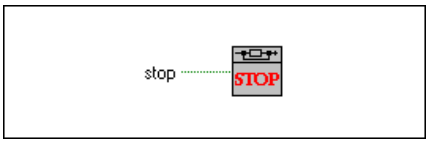
scope is the resulting waveform graph.

The following diagram shows the GSim Scope 1D program. The VI is a predefined scalarized graph.



GSim Stop

Stops the control or simulation task.



stop is a Boolean value.

If **stop** is FALSE, nothing happens. If **stop** is TRUE, this VI sends a message to the GSim Manager VI, which immediately stops the execution. The general error handler treats this condition. All values calculated prior to **stop** becoming TRUE are valid and accessible.

Users and specific situations that forbid further calculations can alter the **stop** input. Typical triggers include calculations that are too slow to realize real-time behavior and calculations in which physically unrealistic values were determined. As a developer, you must integrate GSim Stop VI into the project explicitly.

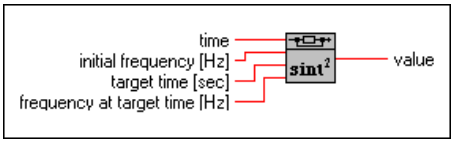
13

Data Sources

The chapter describes the signal-producing sources in the `SOURCES.llb` library. You can use these VIs as inputs for other routines in GSIM. Furthermore, two timing VIs are described: GSim Simulation Clock and GSim Realtime Clock.

GSim Chirp Signal

Generates a chirp signal.



DBL **time** is the current time value.

DBL **initial frequency** [**Hz**] is the frequency at **time**=0.

DBL **target time** [**sec**] is the upper limit for the time. **target time** [**sec**] must be greater than 0. A value of 0 results in an error. See Appendix A, *Error Codes*, for a list of errors and their descriptions.

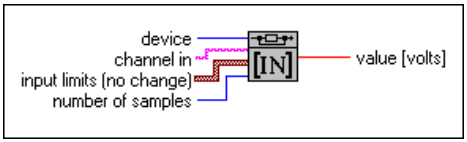
DBL **frequency at target time** [**Hz**] is the frequency at the value of target time.

DBL **value** is the calculated chirp signal, which depends on the value of the other parameters in this VI. **value** is calculated according to the following formula:

$$value = \sin\left(\frac{time(frequency\ at\ target\ time[Hz] - initial\ frequency[Hz])}{target\ time}\right)$$

GSim Collect Data In

Collects *n* data points and calculates the mean value of these data points using a DAQ board.



device is the number of the plug-in data acquisition board. You must specify **device**.

channel in is a string that specifies the analog input channel.

input limits [**no change**] is an array of clusters. Only the first array element is important (the array structure was chosen to maintain the general structure of DAQ operations). The default for **input limits** [**no change**] is an empty array, so the limits do not change from their original default.

high limit specifies the maximum voltage the board measures at a particular channel.

low limit specifies the minimum voltage the board measures at a particular channel.

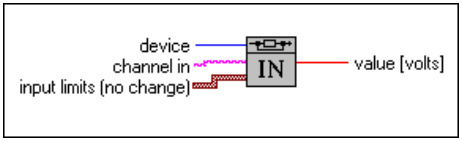
number of samples is the number of data points that have to be read.

value [**volts**] is the mean value in volts of **number of samples** of data in.

Consult the *LabVIEW Function and VI Reference Manual* for more information on devices, channels, and input limits.

GSim Data In

Performs a single measurement of a channel using a DAQ board.



device is the number of the plug-in data acquisition board. You must specify **device**.

channel in is a string that specifies the analog input channel.

input limits [**no change**] is an array of clusters. Only the first array element is important (the array structure was chosen to maintain the general structure of DAQ operations). The default for **input limits** [**no change**] is an empty array, so the limits do not change from their original default.

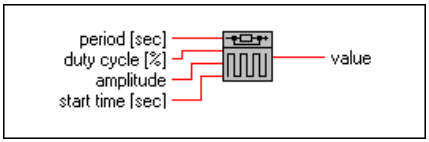
- **high limit** specifies the maximum voltage the board measures at a particular channel.
- **low limit** specifies the minimum voltage the board measures at a particular channel.

value [**volts**] is the value in volts of data in.

Consult the *LabVIEW Function and VI Reference Manual* for more information on devices, channels, and input settings.

GSim Pulse Generator

Generates a pulse signal.



DBL **period** [**sec**] is the period of the pulse signal in seconds. The value of **period** [**sec**] must be positive. A non-positive **period** [**sec**] produces an error. See Appendix A, *Error Codes*, for a list of errors and their descriptions.

DBL **duty cycle** [**%**] is the ratio between the high and low levels of the pulse. **duty cycle** [**%**] must be in the range of 0% to 100%. A violation of this condition produces an error. See Appendix A, *Error Codes*, for a list of errors and their descriptions.

DBL **amplitude** is the maximum amplitude of the pulse where `0` sets the output to 0 amplitude.

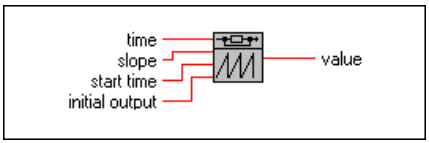
DBL **start time** [**sec**] is the trigger for the pulse signal.

DBL **value** is the generated pulse signal, which depends on the other parameters in this VI. **value** is calculated according to the following formulas, where *clock* is the current simulation time:

$$value = \begin{cases} 0 & \text{if } \dfrac{clock - start\ time}{period} - \text{int}\left(\dfrac{clock - start\ time}{period}\right) \geq \dfrac{duty\ cycle[\%]}{100} \\ amplitude & \text{otherwise} \end{cases}$$

GSim Ramp

Generates a ramp signal.



time is the current time value.

slope is the slew rate of the ramp.

start time marks the start of the rising edge of the signal.

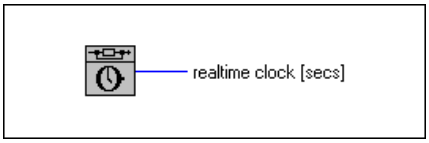
initial output is the offset of the signal.

value is the calculated ramp chirp signal, which depends on the other parameters in this VI. **value** is calculated according to the following formulas:

$$value = \begin{cases} slope \times time + initial\ output & \text{if } time > start\ time \\ initial\ output & \text{otherwise} \end{cases}$$

GSim Realtime Clock

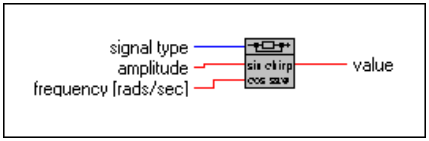
Generates the duration in seconds of the control or simulation task. This VI is based on the Get Date/Time in Seconds function, which you can access from the **Functions»Time & Date** palette in LabVIEW or BridgeVIEW.



realtime clock [**secs**] is the time in seconds since the start of the control or simulation task.

GSim Signal Generator

Generates a user-defined signal.



U32 **signal type** specifies a signal type: `sin` (sine signal), `square` (square signal), `sawtooth` (sawtooth signal), or `random` (randomly chosen signal, uniformly distributed).

DBL **amplitude** is the maximum amplitude of the signal.

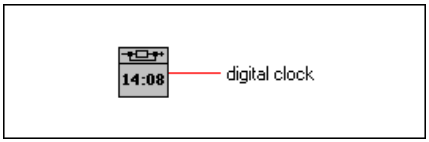
DBL **frequency** [**rads**/**sec**] is the frequency of the signal. With random signals, this parameter is ignored. **frequency** [**rads**/**sec**] must be a non-negative value. A violation of this condition produces an error. See Appendix A, *Error Codes*, for a list of errors and their descriptions.

DBL **value** is the output signal, which depends on the other parameters in this VI. **value** is calculated according to the following formulas, where *simulation time* is produced by the GSim Simulation Clock VI.

Signal Type	Formula to Determine Output Signal
sin	$value = amplitude \times \sin\left(simulation\ time \times frequency\left[\frac{rads}{sec}\right]\right)$
square	$value = amplitude \times \mathrm{sign}\left(\sin\left(simulation\ time \times frequency\left[\frac{rads}{sec}\right]\right)\right)$
sawtooth	$value = amplitude \times \mathrm{mod}(simulation\ time \times frequency, frequency)$
random	$value = amplitude \times \mathrm{random}(\ )$

GSim Simulation Clock

Outputs the simulated time of the control or simulation task.

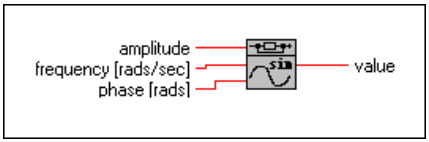


digital clock outputs the simulated time, which is measured in multiples of the global variable *dt*, which is fixed at the start of a control or simulation task.

digital clock is equal to the value of $index \times dt$, where *index* is the counter of the underlying While Loop.

GSim Sine Wave

Generates a sine signal.



amplitude is the maximum amplitude of the sine signal.

frequency [**rads**/**sec**] is the frequency of the sine signal in rads/sec. The value of **frequency** [**rads**/**sec**] must be non-negative. A negative value produces an error. See Appendix A, *Error Codes*, for a list of errors and their descriptions.

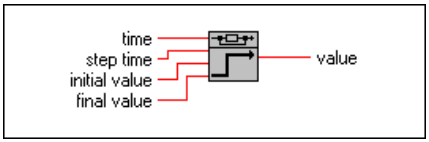
phase [**rads**] is the phase of the sine signal in rads.

value is the output of the sine signal as determined by the other parameters in this VI. **value** is calculated according to the following formula, where *simulation time* denotes the output of the GSim Simulation Time VI:

$$value = amplitude \times \sin((simulation\ time \times frequency[rads/sec]) + phase[rads])$$

GSim Step

Generates a step signal.



time is the current time value.

step time is the point of discontinuity of the step function.

initial value is the lower limit of the given step function.

final value is the upper limit of the given step function.

value is the calculated step signal, which depends on the other parameters in this VI. **value** is calculated according to the following formulas:

$$value = \begin{cases} final\ value & \text{if } time > step\ time \\ initial\ value & \text{otherwise} \end{cases}$$

A

Error Codes

This appendix lists and describes error codes returned by the control and simulation VIs.

Table A-1. Error Codes

Error Code	Error	Description
0	NoErr	No Error.
–24001	DiscreteFiltErr	Numerator or denominator conflict.
–24002	DiscreteSSErr	Starting conditions conflict—MIMO problem or signal in.
–24003	DiscreteZPErr	Needs pairwise conjugate complex numbers.
–24004	ContSSErr	Starting conditions conflict—MIMO problem or signal in.
–24005	TFNormalErr	Numerator or denominator conflict.
–24006	TFSIMOErr	Numerator or denominator conflict.
–24007	TFSISOErr	Numerator or denominator conflict.
–24008	QuantizerErr	Quantizer interval is less than zero.
–24009	RelayErr	Switch on is less than switch off.
–24010	SaturationErr	Upper limit is less than lower limit.
–24011	DelayErr	Time delay is less than zero.
–24012	StopErr	User-canceled simulation.
–24013	ChirpErr	Target time is equal to zero.
–24014	PulseErr	Period is non-positive or invalid duty cycle was specified.
–24015	GeneratorErr	Frequency is less than zero.
–24016	SineWaveErr	Frequency is less than zero.
–24017	MIMOErr	Not a valid MIMO system.
–24018	DataInErr	DAQ problems (input).

Table A-1. Error Codes (Continued)

Error Code	Error	Description
–24019	DataOutErr	DAQ problems (output).
–24020	InValidInputOutputErr	Invalid input or output value.
–24021	MatrixASizeErr	Matrix **A** needs to be a square matrix.
–24022	MatrixABSizeErr	Number of rows in matrix **A** must equal the number of rows in matrix **B**.
–24023	MatrixCDSizeErr	Number of rows in matrix **C** must equal the number of rows in matrix **D**.
–24024	MatrixACSizeErr	Number of columns in matrix **A** must equal the number of columns in matrix **C**.
–24025	MatrixBDSizeErr	Number of columns in matrix **B** must equal the number of columns in matrix **D**.
–24026	DenominatorSizeErr	Number of elements in the denominator must be greater than the number of elements in the numerator.
–24027	DesiredPolesSizeErr	Number of desired poles must equal the number of poles in the main system.

B

Description of the Betas and Alphas Parameters

This appendix provides a general description of the **Betas** and **Alphas** parameters. You should understand how these parameters work before implementing them in functions.

Betas is an array containing the numerator coefficients of the transfer function. If the numerator of the transfer function is given by

$$H(s) = \beta_n s^n + \beta_{n-1} s^{n-1} + \ldots + \beta_1 s + \beta_0$$

the first element of the array **Betas** is β_0, the second element is β_1, and the nth element of the array is β_n.

Alphas is an array containing the denominator coefficients of the transfer function. If the denominator of the transfer function is given by

$$H(s) = \alpha_n s^n + \alpha_{n-1} s^{n-1} + \ldots + \alpha_1 s + \alpha_0$$

the first element of the array **Alphas** is α_0, the second element is α_1, and the nth element of the array is α_n.

The following two examples show transfer functions as they should appear on the front panel.

$$H(s) = 3s^3 + s + 2.70$$

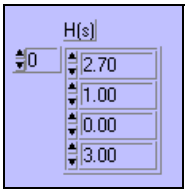


Figure B-1. An Example of H(s) in a 1D-Array Input on the Front Panel

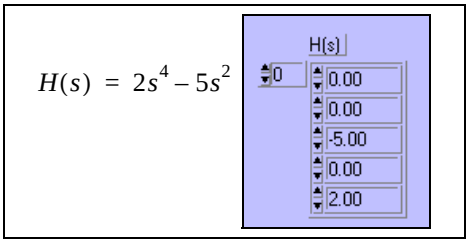


Figure B-2. Another Example of H(s) in a 1D-Array Input on the Front Panel

C

Customer Communication

For your convenience, this appendix contains forms to help you gather the information necessary to help us solve your technical problems and a form you can use to comment on the product documentation. When you contact us, we need the information on the Technical Support Form and the configuration form, if your manual contains one, about your system configuration to answer your questions as quickly as possible.

National Instruments has technical assistance through electronic, fax, and telephone systems to quickly provide the information you need. Our electronic services include a bulletin board service, an FTP site, a fax-on-demand system, and e-mail support. If you have a hardware or software problem, first try the electronic support systems. If the information available on these systems does not answer your questions, we offer fax and telephone support through our technical support centers, which are staffed by applications engineers.

Electronic Services

Bulletin Board Support

National Instruments has BBS and FTP sites dedicated for 24-hour support with a collection of files and documents to answer most common customer questions. From these sites, you can also download the latest instrument drivers, updates, and example programs. For recorded instructions on how to use the bulletin board and FTP services and for BBS automated information, call (512) 795-6990. You can access these services at:

United States: 512 794 5422
Up to 14,400 baud, 8 data bits, 1 stop bit, no parity

United Kingdom: 01635 551422
Up to 9,600 baud, 8 data bits, 1 stop bit, no parity

France: 01 48 65 15 59
Up to 9,600 baud, 8 data bits, 1 stop bit, no parity

FTP Support

To access our FTP site, log on to our Internet host, `ftp.natinst.com`, as `anonymous` and use your Internet address, such as `joesmith@anywhere.com`, as your password. The support files and documents are located in the `/support` directories.

Fax-on-Demand Support

Fax-on-Demand is a 24-hour information retrieval system containing a library of documents on a wide range of technical information. You can access Fax-on-Demand from a touch-tone telephone at (512) 418-1111.

E-Mail Support (Currently USA Only)

You can submit technical support questions to the applications engineering team through e-mail at the Internet address listed below. Remember to include your name, address, and phone number so we can contact you with solutions and suggestions.

`support@natinst.com`

Telephone and Fax Support

National Instruments has branch offices all over the world. Use the list below to find the technical support number for your country. If there is no National Instruments office in your country, contact the source from which you purchased your software to obtain support.

Country	Telephone	Fax
Australia	03 9879 5166	03 9879 6277
Austria	0662 45 79 90 0	0662 45 79 90 19
Belgium	02 757 00 20	02 757 03 11
Brazil	011 288 3336	011 288 8528
Canada (Ontario)	905 785 0085	905 785 0086
Canada (Quebec)	514 694 8521	514 694 4399
Denmark	45 76 26 00	45 76 26 02
Finland	09 725 725 11	09 725 725 55
France	01 48 14 24 24	01 48 14 24 14
Germany	089 741 31 30	089 714 60 35
Hong Kong	2645 3186	2686 8505
Israel	03 6120092	03 6120095
Italy	02 413091	02 41309215
Japan	03 5472 2970	03 5472 2977
Korea	02 596 7456	02 596 7455
Mexico	5 520 2635	5 520 3282
Netherlands	0348 433466	0348 430673
Norway	32 84 84 00	32 84 86 00
Singapore	2265886	2265887
Spain	91 640 0085	91 640 0533
Sweden	08 730 49 70	08 730 43 70
Switzerland	056 200 51 51	056 200 51 55
Taiwan	02 377 1200	02 737 4644
United Kingdom	01635 523545	01635 523154
United States	512 795 8248	512 794 5678

Technical Support Form

Photocopy this form and update it each time you make changes to your software or hardware, and use the completed copy of this form as a reference for your current configuration. Completing this form accurately before contacting National Instruments for technical support helps our applications engineers answer your questions more efficiently.

If you are using any National Instruments hardware or software products related to this problem, include the configuration forms from their user manuals. Include additional pages if necessary.

Name _____

Company _____

Address _____

_____

Fax (___) _____Phone (___) _____

Computer brand_____ Model _____Processor_____

Operating system (include version number) _____

Clock speed _____MHz RAM _____MB Display adapter _____

Mouse ___yes ___no Other adapters installed_____

Hard disk capacity _____MB Brand_____

Instruments used _____

_____

National Instruments hardware product model _____ Revision _____

Configuration _____

National Instruments software product _____ Version _____

Configuration _____

The problem is: _____

_____

_____

_____

_____

List any error messages: _____

_____

_____

The following steps reproduce the problem: _____

_____

_____

_____

_____

Control and Simulation Software for G Hardware and Software Configuration Form

Record the settings and revisions of your hardware and software on the line to the right of each item. Complete a new copy of this form each time you revise your software or hardware configuration, and use this form as a reference for your current configuration. Completing this form accurately before contacting National Instruments for technical support helps our applications engineers answer your questions more efficiently.

National Instruments Products

Hardware revision _____

Interrupt level of hardware _____

DMA channels of hardware _____

Base I/O address of hardware _____

Programming choice _____

National Instruments software _____

Other boards in system _____

Base I/O address of other boards _____

DMA channels of other boards _____

Interrupt level of other boards _____

Other Products

Computer make and model _____

Microprocessor _____

Clock frequency or speed _____

Type of video board installed _____

Operating system version _____

Operating system mode _____

Programming language _____

Programming language version _____

Other boards in system _____

Base I/O address of other boards _____

DMA channels of other boards _____

Interrupt level of other boards _____

Documentation Comment Form

National Instruments encourages you to comment on the documentation supplied with our products. This information helps us provide quality products to meet your needs.

Title: *Control and Simulation Software for G Reference Manual*

Edition Date: February 1998

Part Number: 321878A-01

Please comment on the completeness, clarity, and organization of the manual.

_____

_____

_____

_____

_____

_____

_____

If you find errors in the manual, please record the page numbers and describe the errors.

_____

_____

_____

_____

_____

_____

_____

Thank you for your help.

Name _____

Title _____

Company _____

Address _____

_____

E-Mail Address _____

Phone (___) _____ Fax (___) _____

Mail to: Technical Publications
National Instruments Corporation
6504 Bridge Point Parkway
Austin, Texas 78730-5039

Fax to: Technical Publications
National Instruments Corporation
512 794 5678

Glossary

Prefix	Meanings	Value
n-	nano-	10^{-9}
µ-	micro-	10^{-6}
m-	milli-	10^{-3}

Numbers/Symbols

∞	Infinity.
π	Pi.
1D	One-dimensional.
2D	Two-dimensional.

A

A/D	Analog/digital.
Adams Method	An integrator that uses past values of a solution to approximate (interpolation and extrapolation) the function in the interval of integration.
ANSI	American National Standards Institute.
array	Ordered, indexed set of data elements of the same type.
array shell	Front panel object that houses an array. An array shell consists of an index display, a data object window, and an optional label. It can accept various data types.
ASCII	American Standard Code for Information Interchange.

B

block diagram	Pictorial description or representation of a program or algorithm. In G, the block diagram is the source code for the VI. It consists of executable icons called nodes as well as wires that carry data between the nodes.
Boolean controls and indicators	Front panel objects used to manipulate and display Boolean (TRUE or FALSE) data. Several styles are available, such as switches, buttons, and LEDs.
BridgeVIEW	A G-based (graphical) program development application used commonly for industrial automation purposes.
broken VI	VI that cannot be compiled or run. It is signified by a broken arrow in the Run button.

C

Case structure	Conditional branching control structure, which executes one and only one of its subdiagrams based on its input. It is the combination of the IF, THEN, ELSE, and CASE statements in control flow languages.
CIN	*See* code interface node.
cluster	Set of ordered, unindexed data elements of any data type including numeric, Boolean, string, array, or cluster. The elements must be all controls or all indicators.
cluster shell	Front panel object that contains the elements of a cluster.
code interface node	CIN. Special block diagram node through which you can link conventional, text-based code to a VI.
compile	Process that converts high-level code to machine-executable code. VIs are compiled automatically before they run for the first time after creation or alteration.
conditional terminal	Terminal of a While Loop that contains a Boolean value that determines whether the VI performs another iteration.
connector	Part of the VI or function node that contains input and output terminals. Data passes to and from the node through a connector.

constant — *See* universal constant and user-defined constant.

continuous run — Execution mode in which a VI is run repeatedly until the operator stops it. You enable it by clicking the **Continuous Run** button.

Continuous Run button — Icon that indicates the execution status of a VI.

control — Front panel object for entering data to a VI interactively or to a subVI programmatically.

conversion — Process of changing the type of a data element.

CPU — Central processing unit.

current VI — VI whose front panel, block diagram, or Icon Editor is the active window.

D

DAQ — *See* data acquisition.

data acquisition — DAQ. Process of acquiring data, typically from A/D or digital input plug-in boards.

data logging — To acquire data and simultaneously store it in a disk file. G file I/O functions can log data.

data storage formats — Arrangement and representation of data stored in memory.

data type — Format for information. In BridgeVIEW, acceptable data types for tag configuration are analog, discrete, bit array, and string. In LabVIEW, acceptable data types for most functions are numeric, array, string, and cluster.

data type descriptor — Code that identifies data types, used in data storage and representation.

datalog file — File that stores data as a sequence of records of a single, arbitrary data type that you specify when you create the file. While all the records in a datalog file must be of a single type, that type can be complex; for instance, you can specify that each record is a cluster containing a string, a number, and an array.

description box — Dialog box containing online documentation for a G object.

developer — *See* system developer.

dimension — Size and structure attribute of an array.

E

empty array — Array that has zero elements but has a defined data type. For example, an array that has a numeric control in its data display window but has no defined values for any element is an empty numeric array.

error message — Indication of a software or hardware malfunction, or an unacceptable data entry attempt.

Euler Method — A simple integrator that calculates a new value of the solution based on the last value.

F

feedback — An information stream that is fed back (control—real hardware, simulation —wire structures).

FFT — Fast Fourier transform.

file refnum — Identifier that G associates with a file when you open it. You use the file refnum to indicate that you want a function or VI to perform an operation on the open file.

For Loop — Iterative loop structure that executes its subdiagram a set number of times. Equivalent to conventional code: `For i = 0 to n – 1, do....`

frame — Subdiagram of a Sequence structure.

front panel — Interactive user interface of a VI. Modeled after the front panel of physical instruments, it is composed of switches, slides, meters, graphs, charts, gauges, LEDs, or other controls or indicators.

function — Built-in execution element, comparable to an operator, function, or statement in a conventional language.

G

G	Graphical programming language used to develop LabVIEW and BridgeVIEW applications.
global variable	Non-reentrant subVI with local memory that uses an uninitialized shift register to store data from one execution to the next. The memory of copies of these subVIs is shared and thus can be used to pass global data between them.
graph control	Front panel object that displays data in a Cartesian plane.
GSIM	Abbreviation for the Control and Simulation Software for G.

H

Help	Online instructions that explain how to use a Windows application. The **Help** menu displays specific Help topics. Pressing `<F1>` displays a list of Help topics.
Hz	Hertz. Cycles per second.

I

I/O	Input/output. Transfer of data to or from a computer system involving communications channels, operator input devices, and/or data acquisition and control interfaces.
indicator	Front panel object that displays output.
integrator	General method that solves differential equations.

L

LabVIEW	Laboratory Virtual Instrument Engineering Workbench. Program development application based on the programming language G used commonly for test and measurement purposes.
local variable	Variable that enables you to read or write to one of the controls or indicators on the front panel of your VI.

M

matrix	Two-dimensional array.
MB	Megabytes of memory.
MIMO	Multiple-Input, Multiple-Output. A system with more than one input and more than one output.

N

numeric controls and indicators	Front panel objects used to manipulate and display or input and output numeric data.

P

PID	Proportional-Integral-Derivative. A specific controller, extremely important in practice, that is based on a proportional factor, an integrator, and a differentiator.
plot	Graphical representation of an array of data shown either in a graph or a chart.

R

real-time	Pertaining to the performance of a computation during the actual time that the related physical process transpires so results of the computation can be used in guiding the physical process.
reentrant execution	Mode in which calls to multiple instances of a subVI can execute in parallel with distinct and separate data storage.
representation	Subtype of the numeric data type. Representations include signed and unsigned byte, word, and long integers, as well as single-, double-, and extended-precision floating-point numbers, both real and complex.

root locus plot	A technique to describe the roots of the closed-loop characteristic equations of a system; can be used to determine stable control systems.
Runge-Kutta Method	A sophisticated and the most common integrator in practice. Some values in between are used to improve the accuracy of the solution.

S

scalar	Number capable of being represented by a point on a scale. A single value as opposed to an array. Scalar Booleans and clusters are explicitly singular instances of their respective data types.
SISO	Single-Input, Single-Output. A system with exactly one input and exactly one output.
state space	System description based on internal states (MIMO systems).
string controls and indicators	Front panel objects used to manipulate and display or input and output text.
structure	Program control element, such as a Sequence, Case, For Loop, or While Loop.
subdiagram	Block diagram within the border of a structure.
subVI	VI used in the block diagram of another VI. It is comparable to a subroutine.
system developer	Creator of the application software to be executed.

T

top-level VI	VI at the top of the VI hierarchy. This term distinguishes the VI from its subVIs.
transfer function	System description based on polynomials (SISO systems).
two-dimensional	Having two dimensions, such as an array with both rows and columns.
type descriptor	*See* data type descriptor.

U

universal constant	Uneditable block diagram object that emits a particular ASCII character or standard numeric constant, such as pi.
user-defined constant	Block diagram object that emits a value you set.

V

VI	*See* virtual instrument.
VI library	Special file that contains a collection of related VIs for a specific use.
virtual instrument	VI. Program in the graphical programming language G; so-called because it models the appearance and function of a physical instrument.

W

While Loop	Loop structure that repeats a section of code until a condition is met. It is comparable to a Do loop or a Repeat-Until loop in conventional programming languages.
wire	Data path between nodes. *See also* data flow.
wire branch	Section of wire that contains all the wire segments from one junction to another, from a terminal to the next junction, or from one terminal to another if no junctions exist between the terminals.

Z

zero pole	System description based on the roots of the transfer function polynomials (SISO systems).

Index

A

Alphas, B-1 to B-2

B

basic system operations, 3-1
basic VIs, 3-1 to 3-13
BASIC.llb, 1-6, 3-1 to 3-13
    GSim Compute Parameters, 1-6, 3-10 to 3-11
    GSim Compute Poles, 1-6, 3-9
    GSim Compute Zeros, 1-6, 3-8
    GSim Feedback, 1-6, 3-6 to 3-7
    GSim Normalize, 1-6, 3-12 to 3-13
    GSim Parallel, 1-6, 3-4 to 3-5
    GSim Series, 1-6, 3-2 to 3-3
Betas, B-1 to B-2
Bode plot, 6-2, 6-3
bulletin board support, C-1

C

computing poles, 3-9
computing zeros, 3-8
CONNECT.llb, 1-6, 7-1 to 7-3
    GSim Initial Condition, 1-6, 7-1
    GSim Input Selector, 1-6, 7-2
    GSim Trigger, 1-6, 7-3
connections, 7-1 to 7-3
control and simulation
    applications, 1-5
        development of real-time control systems, 1-5
        education, 1-5
        PID controllers in GSIM, 2-3
        real-time control of deployed systems, 1-5
        simulations (with data acquisition), 1-5
        system analysis and optimization, 1-5
    compatibility, 1-1
    execution order, 11-4 to 11-5
    features, 1-2 to 1-3
    graphical representations, 1-3 to 1-4
    libraries
        See libraries.
    libraries and VIs, list, 1-6 to 1-9
    models, 1-1
    problems, 1-3 to 1-4
    programs, 1-2, 2-10
    software, 1-1
    step-by-step approach to GSIM problems, 2-4 to 2-9
    stopping tasks, 12-3
    timing, 11-5, 13-6, 13-8
    VIs
        See VIs.
controls in While Loops, 2-11
conversion VIs, 4-1 to 4-11

CONVERT.llb, 1-6, 4-1 to 4-11
GSim Modal Form, 1-6, 4-11
GSim SS2RP, 1-6, 4-3 to 4-4
GSim SS2TF, 1-6, 4-5 to 4-6
GSim SS2ZP, 1-6, 4-7 to 4-8
GSim TF2SS, 1-6, 4-9
GSim ZP2TF, 1-6, 4-10
converting representations, 4-1 to 4-11
creating feedback structures, 2-5 to 2-6, 2-10
customer communication, xiii, C-1 to C-7

D

DAQ
collecting data, 13-2
getting measurements, 13-3
outputting voltage, 12-1 to 12-2
data sources, 13-1 to 13-10
delaying input, 8-10
denominator coefficients, B-1 to B-2
discrete systems, 8-1 to 8-11
DISCRETE.llb, 1-7, 8-1 to 8-11
GSim Discrete Filter, 1-7, 8-1
GSim Discrete State-Space, 1-7, 8-2 to 8-3
GSim Discrete Transfer Function, 1-7, 8-4
GSim Discrete Zero-Pole, 1-7, 8-5 to 8-6
GSim Discrete-Time Integrator, 1-7, 8-7
GSim First-Order Hold, 1-7, 8-8 to 8-9
GSim Unit Delay, 1-7, 8-10
GSim Zero-Order Hold, 1-7, 8-11
documentation
conventions used in this manual, xii
organization of this manual, xi to xii
related documentation, xiii

E

electronic support services, C-1 to C-2
entering coefficients using Betas and Alphas, B-1 to B-2
error codes, A-1 to A-2
examples
adjusting user interface, 2-6 to 2-7
creating feedback structures, 2-5 to 2-6
generating general GSIM frames, 2-4 to 2-5
House Temperature Control Example, 2-1 to 2-3
modifying programs, 2-8
PID controllers in GSIM, 2-3
step-by-step approach to GSIM problems, 2-4 to 2-9
studying other GSIM programs, 2-9
See also XMPLCON.llb, XMPLSIM1.llb, and XMPLSIM2.llb.
execution order, 11-4 to 11-5

F

feedback structures, 2-5 to 2-6, 2-10
feedback VIs, 5-1 to 5-5
FEEDBACK.llb, 1-6, 5-1 to 5-5
GSim Design from SS, 1-6, 5-3
GSim Design from TF, 1-6, 5-4
GSim Design from ZP, 1-6, 5-5
FREQRESP.llb, 1-6, 6-1 to 6-6
GSim Bode Plot from SS, 1-6, 6-2
GSim Bode Plot from TF, 1-6, 6-3
GSim Nyquist Plot, 1-6, 6-6
GSim Root Locus, 1-6, 6-4
GSim Root Locus Plot Utility, 1-6, 6-5
frequency response VIs, 6-1 to 6-6
FTP support, C-1

G

getting help, 1-10
graphical representations of GSIM problems, 1-3 to 1-4
GSIM
See control and simulation.
GSIM libraries
See libraries.
GSIM programs
prescribed approach, 2-4 to 2-9
GSIM VIs
See VIs.

H

help, 1-10
holding signal values, 8-11
House Temperature Control Example, 2-1 to 2-3

I

installation
Power Macintosh, 1-10
Windows 3.*x*, 1-10
Windows 95/NT, 1-9 to 1-10
integration methods, 8-1, 9-2

L

libraries
BASIC.llb, 1-6, 3-1 to 3-13
CONNECT.llb, 1-6, 7-1 to 7-3
CONVERT.llb, 1-6, 4-1 to 4-11
DISCRETE.llb, 1-7, 8-1 to 8-11
FEEDBACK.llb, 1-6, 5-1 to 5-5
FREQRESP.llb, 1-6, 6-1 to 6-6
LINEAR.llb, 1-7, 9-1 to 9-12
NONLIN.llb, 1-7, 10-1 to 10-11
SHARED.llb, 1-7, 11-1 to 11-5
SINKS.llb, 1-7, 12-1 to 12-3
SOURCES.llb, 1-8, 13-1 to 13-10
XMPLCONT.llb, 1-8
XMPLSIM1.llb, 1-8
XMPLSIM2.llb, 1-9
linear systems, 9-1 to 9-12
LINEAR.llb, 1-7, 9-1 to 9-12
GSim Derivative, 1-7, 9-1
GSim Integrator, 1-7
GSim Limited Integrator, 1-7, 9-3
GSim PID Parallel, 1-7, 9-4
GSim PID Serial, 1-7, 9-5
GSim State-Space, 1-7, 9-6 to 9-7
GSim Transfer Function, 1-7, 9-9 to 9-10
GSim Transfer Function IC, 1-7, 9-8 to 9-9
GSim Zero-Pole, 1-7, 9-11 to 9-12

M

manual
See documentation.
modal form
from state space, 4-11
representation, 4-2

N

NONLIN.llb, 1-7, 10-1 to 10-11
GSim Dead Zone, 1-7, 10-1 to 10-2
GSim Friction, 1-7, 10-3
GSim Memory, 1-7, 10-4
GSim Quantizer, 1-7, 10-5
GSim Rate Limiter, 1-7, 10-6

GSim Relay, 1-7, 10-7
GSim Saturation, 1-7, 10-8
GSim Switch, 1-7, 10-9
GSim Transport Delay, 1-7, 10-10
GSim Zero Crossing, 1-7, 10-11
nonlinear systems, 10-1 to 10-11
normalizing transfer functions, 3-12 to 3-13
numerator coefficients, B-1 to B-2
Nyquist plot, 6-1, 6-6

O

overall system representation, 3-1

P

parameters, 3-10 to 3-11
PID, 9-4, 9-5
PID controllers in GSIM, 2-3
poles, 3-9, 5-1
Power Macintosh installation, 1-10

R

reentrant VIs, 2-10
residual pole
from state space, 4-3 to 4-4
representation, 4-3
root locus plot, 6-1, 6-4, 6-5

S

shared systems, 11-1 to 11-5
SHARED.llb, 1-7, 11-1 to 11-5
GSim Initialize, 1-7, 11-1 to 11-2
GSim Manager, 1-7, 11-2 to 11-3
GSim Synchronizer, 1-7, 11-4 to 11-5
GSim Timer, 1-7, 11-5
signals
calculating derivatives, 9-1
detecting input changes, 7-3
detecting zero crossings, 10-11
generating chirp signals, 13-1
generating pulse signals, 13-4
generating ramp signals, 13-5
generating random signals, 13-7
generating sawtooth signals, 13-7
generating sine signals, 13-7, 13-9
generating square signals, 13-7
generating step signals, 13-10
generating user-defined signals, 13-7
holding values, 8-11
integrating input signals, 8-7, 9-3
limiting change, 10-6
limiting range, 10-8
quantizing input signals, 10-5
simulated timing, 13-8
Simulation VIs
guidelines for using, 2-10 to 2-11
SINKS.llb, 1-7, 12-1 to 12-3
GSim Data Out, 1-7, 12-1 to 12-2
GSim Scope 1d, 1-7, 12-2
GSim Stop, 1-7, 12-3
SOURCES.llb, 1-8, 13-1 to 13-10
GSim Chirp Signal, 1-8, 13-1
GSim Collect Data In, 1-8, 13-2
GSim Data In, 1-8, 13-3
GSim Pulse Generator, 1-8, 13-4
GSim Ramp, 1-8, 13-5
GSim Realtime Clock, 1-8, 13-6
GSim Signal Generator, 1-8, 13-7
GSim Simulation Clock, 1-8, 13-8
GSim Sine Wave, 1-8, 13-9
GSim Step, 1-8, 13-10

state space
calculating output, 9-6 to 9-7
computing data for Bode plot, 6-2
designing linear state feedback, 5-3
discrete state space, 8-2 to 8-3
from transfer function, 4-9
representation, 4-1
to modal form, 4-11
to residual pole, 4-3 to 4-4
to transfer function, 4-5 to 4-6
to zero pole, 4-7 to 4-8
stopping GSIM tasks, 12-3
switching between constants, 10-7
switching between values, 10-9
system representation, 3-1
systems in parallel, 3-4 to 3-5
systems in series, 3-2 to 3-3

T

technical support, C-1 to C-2
time behavior, 11-5
timing, 11-5, 13-6, 13-8
transfer function
Alphas, B-1 to B-2
Betas, B-1 to B-2
calculating output, 9-8 to 9-9, 9-9 to 9-10
computing data for Bode plot, 6-3
computing data for Nyquist plot, 6-6
computing data for root locus plot, 6-4
designing linear state feedback, 5-4
discrete transfer function, 8-4
entering coefficients, B-1 to B-2
from state space, 4-5 to 4-6
from zero pole, 4-10
normalizing, 3-12 to 3-13
representation, 4-2
to state space, 4-9

V

VIs
sorted by library, 1-6 to 1-9
Bode Application, 1-8
Bode Plot Wizard, 1-8
Child Toy Example, 1-8
Control Example, 1-9
Curtis-Hirschfelder Example, 1-9
Electronic Pacemaker, 1-8
F14 Example, 1-9
GSim Bode Plot from SS, 1-6, 6-2
GSim Bode Plot from TF, 1-6, 6-3
GSim Chirp Signal, 1-8, 13-1
GSim Collect Data In, 1-8, 13-2
GSim Compute Parameters, 1-6, 3-10 to 3-11
GSim Compute Poles, 1-6, 3-9
GSim Compute Zeros, 1-6, 3-8
GSim Data In, 1-8, 13-3
GSim Data Out, 1-7, 12-1 to 12-2
GSim Dead Zone, 1-7, 10-1 to 10-2
GSim Derivative, 1-7, 9-1
GSim Design from SS, 1-6, 5-3
GSim Design from TF, 1-6, 5-4
GSim Design from ZP, 1-6, 5-5
GSim Discrete Filter, 1-7, 8-1
GSim Discrete State-Space, 1-7, 8-2 to 8-3
GSim Discrete Transfer Function, 1-7, 8-4
GSim Discrete Zero-Pole, 1-7, 8-5 to 8-6
GSim Discrete-Time Integrator, 1-7, 8-7
GSim Feedback, 1-6, 3-6 to 3-7
GSim First-Order Hold, 1-7, 8-8 to 8-9
GSim Friction, 1-7, 10-3
GSim Initial Condition, 1-6, 7-1
GSim Initialize, 1-7, 2-10, 11-1 to 11-2

GSim Input Selector, 1-6, 7-2
GSim Integrator, 1-7
GSim Limited Integrator, 1-7, 9-3
GSim Manager, 1-7, 2-10, 11-2 to 11-3
GSim Memory, 1-7, 10-4
GSim Modal Form, 1-6, 4-11
GSim Normalize, 1-6, 3-12 to 3-13
GSim Nyquist Plot, 1-6, 6-6
GSim Parallel, 1-6, 3-4 to 3-5
GSim PID Parallel, 1-7, 9-4
GSim PID Serial, 1-7, 9-5
GSim Pulse Generator, 1-8, 13-4
GSim Quantizer, 1-7, 10-5
GSim Ramp, 1-8, 13-5
GSim Rate Limiter, 1-7, 10-6
GSim Realtime Clock, 1-8, 13-6
GSim Relay, 1-7, 10-7
GSim Root Locus, 1-6, 6-4
GSim Root Locus Plot Utility, 1-6, 6-5
GSim Saturation, 1-7, 10-8
GSim Scope 1d, 1-7, 12-2
GSim Series, 1-6, 3-2 to 3-3
GSim Signal Generator, 1-8, 13-7
GSim Simulation Clock, 1-8, 13-8
GSim Sine Wave, 1-8, 13-9
GSim SS2RP, 1-6, 4-3 to 4-4
GSim SS2TF, 1-6, 4-5 to 4-6
GSim SS2ZP, 1-6, 4-7 to 4-8
GSim State-Space, 1-7, 9-6 to 9-7
GSim Step, 1-8, 13-10
GSim Stop, 1-7, 12-3
GSim Switch, 1-7, 10-9
GSim Synchronizer, 1-7, 11-4 to 11-5
GSim TF2SS, 1-6, 4-9
GSim Timer, 1-7, 11-5
GSim Transfer Function, 1-7, 9-9 to 9-10
GSim Transfer Function IC, 1-7, 9-8 to 9-9
GSim Transport Delay, 1-7, 10-10
GSim Trigger, 1-6, 7-3
GSim Unit Delay, 1-7, 8-10
GSim Zero Crossing, 1-7, 10-11
GSim Zero-Order Hold, 1-7, 8-11
GSim Zero-Pole, 1-7, 9-11 to 9-12
GSim ZP2TF, 1-6, 4-10
Gun and Moving Target Example, 1-8
H(s) Example, 1-9
H(s) Example IC, 1-9
House Temperature Control Example, 1-9, 2-1 to 2-3
House Thermostat Example, 1-9
Integration Example, 1-9
Inverted Pendulum Example, 1-9
Limited Integration Example, 1-9
Lorenz Attractor Example, 1-8
Mathieu Example, 1-9
Nyquist Plot Wizard, 1-8
Parameter Estimation Example, 1-9
PID Example, 1-8
Population Growth Example, 1-9
Predator-Prey Example, 1-9
Quarter Car Dynamics, 1-8
reentrant, 2-10
Relay Feedback Example, 1-8
RLC Circuit Example, 1-8
Robot Joint Design, 1-8
Root Locus Wizard, 1-8
Suspended Ball, 1-8
Van der Pol Example, 1-9
Water Tank Example, 1-8

W

Windows 3.*x* installation, 1-10
Windows 95/NT installation, 1-9 to 1-10

X

XMPLCONT.llb, 1-8
Bode Application, 1-8
Bode Plot Wizard, 1-8
Electronic Pacemaker, 1-8
Nyquist Plot Wizard, 1-8
Quarter Car Dynamics, 1-8
Robot Joint Design, 1-8
Root Locus Wizard, 1-8
Suspended Ball, 1-8
XMPLSIM1.llb, 1-8
Child Toy Example, 1-8
Gun and Moving Target Example, 1-8
Lorenz Attractor Example, 1-8
PID Example, 1-8
Relay Feedback Example, 1-8
RLC Circuit Example, 1-8
Water Tank Example, 1-8
XMPLSIM2.llb, 1-9
Control Example, 1-9
Curtis-Hirschfelder Example, 1-9
F14 Example, 1-9
H(s) Example, 1-9
H(s) Example IC, 1-9
House Temperature Control Example, 1-9, 2-1 to 2-3
House Thermostat Example, 1-9
Integration Example, 1-9
Inverted Pendulum Example, 1-9
Limited Integration Example, 1-9
Mathieu Example, 1-9
Parameter Estimation Example, 1-9
Population Growth Example, 1-9
Predator-Prey Example, 1-9
Van der Pol Example, 1-9

Z

zero pole
calculating output, 9-11 to 9-12
designing linear state feedback, 5-5
discrete zero pole, 8-5 to 8-6
from state space, 4-7 to 4-8
representation, 4-2
to transfer function, 4-10
zeros, 3-8