LabVIEW™

System Identification Toolkit User Manual

NATIONAL INSTRUMENTS™

September 2004 Edition
Part Number 371001B-01

Worldwide Technical Support and Product Information

ni.com

National Instruments Corporate Headquarters

11500 North Mopac Expressway Austin, Texas 78759-3504 USA Tel: 512 683 0100

Worldwide Offices

Australia 1800 300 800, Austria 43 0 662 45 79 90 0, Belgium 32 0 2 757 00 20, Brazil 55 11 3262 3599,
Canada (Calgary) 403 274 9391, Canada (Ottawa) 613 233 5949, Canada (Québec) 450 510 3055,
Canada (Toronto) 905 785 0085, Canada (Vancouver) 604 685 7530, China 86 21 6555 7838,
Czech Republic 420 224 235 774, Denmark 45 45 76 26 00, Finland 385 0 9 725 725 11,
France 33 0 1 48 14 24 24, Germany 49 0 89 741 31 30, India 91 80 51190000, Israel 972 0 3 6393737,
Italy 39 02 413091, Japan 81 3 5472 2970, Korea 82 02 3451 3400, Malaysia 603 9131 0918,
Mexico 01 800 010 0793, Netherlands 31 0 348 433 466, New Zealand 0800 553 322, Norway 47 0 66 90 76 60,
Poland 48 22 3390150, Portugal 351 210 311 210, Russia 7 095 783 68 51, Singapore 65 6226 5886,
Slovenia 386 3 425 4200, South Africa 27 0 11 805 8197, Spain 34 91 640 0085, Sweden 46 0 8 587 895 00,
Switzerland 41 56 200 51 51, Taiwan 886 2 2528 7227, Thailand 662 992 7519,
United Kingdom 44 0 1635 523545

For further support information, refer to the *Technical Support and Professional Services* appendix. To comment on National Instruments documentation, refer to the National Instruments Web site at ni.com/info and enter the info code feedback.

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 document 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

Trademarks

LabVIEW™, MATRIXx™, National Instruments™, National Instruments Alliance Partner™, NI™, ni.com™, and NI-DAQ™ are trademarks of National Instruments Corporation.

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

Members of the National Instruments Alliance Partner Program are business entities independent from National Instruments and have no agency, partnership, or joint-venture relationship with National Instruments.

Patents

For patents covering National Instruments products, refer to the appropriate location: **Help»Patents** in your software, the `patents.txt` file on your CD, or `ni.com/patents`.

WARNING REGARDING USE OF NATIONAL INSTRUMENTS PRODUCTS

(1) NATIONAL INSTRUMENTS PRODUCTS ARE NOT DESIGNED WITH COMPONENTS AND TESTING FOR A LEVEL OF RELIABILITY SUITABLE FOR USE IN OR IN CONNECTION WITH SURGICAL IMPLANTS OR AS CRITICAL COMPONENTS IN ANY LIFE SUPPORT SYSTEMS WHOSE FAILURE TO PERFORM CAN REASONABLY BE EXPECTED TO CAUSE SIGNIFICANT INJURY TO A HUMAN.

(2) IN ANY APPLICATION, INCLUDING THE ABOVE, RELIABILITY OF OPERATION OF THE SOFTWARE PRODUCTS CAN BE IMPAIRED BY ADVERSE FACTORS, INCLUDING BUT NOT LIMITED TO FLUCTUATIONS IN ELECTRICAL POWER SUPPLY, COMPUTER HARDWARE MALFUNCTIONS, COMPUTER OPERATING SYSTEM SOFTWARE FITNESS, FITNESS OF COMPILERS AND DEVELOPMENT SOFTWARE USED TO DEVELOP AN APPLICATION, INSTALLATION ERRORS, SOFTWARE AND HARDWARE COMPATIBILITY PROBLEMS, MALFUNCTIONS OR FAILURES OF ELECTRONIC MONITORING OR CONTROL DEVICES, TRANSIENT FAILURES OF ELECTRONIC SYSTEMS (HARDWARE AND/OR SOFTWARE), UNANTICIPATED USES OR MISUSES, OR ERRORS ON THE PART OF THE USER OR APPLICATIONS DESIGNER (ADVERSE FACTORS SUCH AS THESE ARE HEREAFTER COLLECTIVELY TERMED "SYSTEM FAILURES"). ANY APPLICATION WHERE A SYSTEM FAILURE WOULD CREATE A RISK OF HARM TO PROPERTY OR PERSONS (INCLUDING THE RISK OF BODILY INJURY AND DEATH) SHOULD NOT BE RELIANT SOLELY UPON ONE FORM OF ELECTRONIC SYSTEM DUE TO THE RISK OF SYSTEM FAILURE. TO AVOID DAMAGE, INJURY, OR DEATH, THE USER OR APPLICATION DESIGNER MUST TAKE REASONABLY PRUDENT STEPS TO PROTECT AGAINST SYSTEM FAILURES, INCLUDING BUT NOT LIMITED TO BACK-UP OR SHUT DOWN MECHANISMS. BECAUSE EACH END-USER SYSTEM IS CUSTOMIZED AND DIFFERS FROM NATIONAL INSTRUMENTS' TESTING PLATFORMS AND BECAUSE A USER OR APPLICATION DESIGNER MAY USE NATIONAL INSTRUMENTS PRODUCTS IN COMBINATION WITH OTHER PRODUCTS IN A MANNER NOT EVALUATED OR CONTEMPLATED BY NATIONAL INSTRUMENTS, THE USER OR APPLICATION DESIGNER IS ULTIMATELY RESPONSIBLE FOR VERIFYING AND VALIDATING THE SUITABILITY OF NATIONAL INSTRUMENTS PRODUCTS WHENEVER NATIONAL INSTRUMENTS PRODUCTS ARE INCORPORATED IN A SYSTEM OR APPLICATION, INCLUDING, WITHOUT LIMITATION, THE APPROPRIATE DESIGN, PROCESS AND SAFETY LEVEL OF SUCH SYSTEM OR APPLICATION.

Contents

About This Manual

Conventions ix
Related Documentation x

Chapter 1
Introduction to System Identification

Model-Based Control Design Process 1-2
Analyzing Data and Creating a Dynamic System 1-2
Designing a Controller 1-4
Simulating the Dynamic System 1-4
Deploying the Controller 1-4
Developing Models with the Toolkit 1-5
System Identification Assistant 1-5
System Identification VIs 1-5

Chapter 2
Acquiring and Preprocessing Data

Acquiring Data from a System 2-1
Accounting for Factors that Influence a System 2-2
Choosing a Stimulus Signal 2-2
Common Stimulus Signals 2-3
Filtered Gaussian White Noise 2-4
Random Binary Signal 2-4
Pseudo-Random Binary Sequence 2-5
Chirp Waveform 2-6
Selecting a Sampling Rate 2-7
Applying an Anti-Aliasing Filter 2-7
Preprocessing Data from a System 2-8
Visually Inspecting the Data 2-8
Removing Offsets and Trends 2-8
Removing Offsets 2-9
Removing Trends 2-9
Outliers 2-10
Filtering and Downsampling 2-10
Data Scaling 2-11

Chapter 3
Nonparametric Model Estimation Methods

Impulse Response 3-2
Correlation Analysis 3-3
Prewhitening 3-5
Accuracy of the Impulse Response 3-5
Selecting Impulse Response Length 3-8
Applications of the Impulse Response 3-8
Frequency Response 3-10
Spectral Analysis Method 3-11
Accuracy of the Lag Window 3-12
Applications of the Frequency Response 3-16

Chapter 4
Parametric Model Estimation Methods

Parametric Model Estimation 4-1
General-Linear Polynomial Model 4-1
ARX Model 4-3
ARMAX Model 4-4
Output-Error Model 4-5
Box-Jenkins Model 4-6
AR Model 4-7
State-Space Model 4-8
Polynomial Models versus State-Space Models 4-8
Determining Parameters for the Prediction Error Method 4-9
Akaike's Information Criterion 4-10
Akaike's Final Prediction Error Criterion 4-11
Minimum Data Length Criterion 4-11
Converting Models 4-11
Validating Models 4-12
Validation Methods 4-13
Simulation 4-13
Prediction 4-13
Residual Analysis 4-14
Autocorrelation 4-14
Cross Correlation 4-15
Model Order Reduction 4-15

Chapter 5
Recursive Model Estimation Methods

Defining Recursive Model Estimation 5-1
Adaptive Algorithms 5-3
Least Mean Square 5-3
Normalized Least Mean Square 5-5
Recursive Least Squares 5-6
Kalman Filter 5-7

Chapter 6
System Identification Case Study

Data Preprocessing 6-1
Examining the Time Response Data 6-2
Examining the Frequency Response Data 6-3
Applying a Filter to the Raw Data 6-5
Downsampling the Raw Data 6-6
Estimating the Model 6-7
Akaike's Information Criterion 6-8
Verifying the Results 6-9
Minimum Data Length Criterion 6-10
User-Defined Criterion 6-11
ARX Model Validation 6-13
Simulation and Prediction 6-13
Residual Analysis 6-14
Estimating a State-Space Model 6-17
Finding the Singular Values 6-17
Validating the Estimated State-Space Model 6-18
Additional Examples 6-19

Appendix A
References

Appendix B
Technical Support and Professional Services

About This Manual

The LabVIEW System Identification Toolkit provides a library of VIs and an assistant for developing models of a system based on a large set of raw data. Both tools enable you to complete the entire system identification process from analyzing the raw data to validating the identified model.

This manual discusses the main steps in the system identification process and how to use the System Identification VIs to create applications that can accomplish the various tasks in the process. Refer to the LabVIEW Help, available in LabVIEW by selecting **Help»VI, Function, & How-To Help**, for information about the steps in the assistant and a tutorial about how to use the assistant.

The System Identification Assistant uses the same system identification concepts described in this manual. However, this manual does not include information about how to use the assistant. Refer to the *NI Express Workbench Help*, available in the NI Express Workbench environment by selecting **Help»Express Workbench Help**, for information about the steps in the assistant and a tutorial about how to use the assistant.

Conventions

The following conventions are used in this manual:

» The **»** symbol leads you through nested menu items and dialog box options to a final action. The sequence **File»Page Setup»Options** directs you to pull down the **File** menu, select the **Page Setup** item, and select **Options** from the last dialog box.

This icon denotes a note, which alerts you to important information.

bold Bold text denotes items that you must select or click in the software, such as menu items and dialog box options. Bold text also denotes parameter names.

italic Italic text denotes variables, emphasis, a cross reference, or an introduction to a key concept. This font also denotes text that is a placeholder for a word or value that you must supply.

`monospace`	Text in this font denotes text or characters that you should enter from the keyboard, sections of code, programming examples, and syntax examples. This font is also used for the proper names of disk drives, paths, directories, programs, subprograms, subroutines, device names, functions, operations, variables, filenames, and extensions.
`monospace italic`	Italic text in this font denotes text that is a placeholder for a word or value that you must supply.

Related Documentation

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

- *LabVIEW Help*
- *LabVIEW User Manual*
- *Getting Started with LabVIEW*
- *NI Express Workbench Help*
- *Signal Processing Toolset User Manual*

Refer to Appendix A, *References*, for a list of textbooks and technical papers that National Instruments used to develop the System Identification Toolkit.

1

Introduction to System Identification

System identification involves building mathematical models of a dynamic system based on a set of measured stimulus and response data samples. You can use system identification in a wide range of applications, including mechanical engineering, biology, physiology, meteorology, economics, and model-based control design. For example, engineers use a system model of the relationship between the fuel flow and the shaft speed of turbojet engine to optimize the efficiency and operational stability of the jet engine. Biologists and physiologists use system identification techniques in areas such as eye pupil response and heart rate control. Meteorologists and economists build mathematical models based on historical data for use in forecasting.

This manual focuses on how to use system identification in the model-based control design process, which involves identifying a model of a plant, analyzing and synthesizing a controller for the plant, simulating the plant and controller, and deploying the controller. A plant is the real-world, physical system that you want to control.

System identification is the initial step—identifying a model of a plant—in the model-based control design process. System identification is an iterative process. You first acquire raw data from a real-world system, then format and process the data as necessary, and finally select a mathematical algorithm that you can use to identify a mathematical model of the system. You then can use the resulting mathematical model to analyze the dynamic characteristics of and simulate the time response of the system. You also can use the mathematical model to design a model-based controller.

The LabVIEW System Identification Toolkit assists you in identifying large multivariable models of high-order systems from large amounts of data. The System Identification Toolkit provides two tools, an assistant and a library of VIs, for identifying these discrete single-input single-output (SISO) and multiple-input and multiple-output (MIMO) linear systems, respectively. Both tools enable you to complete the entire system identification process from analyzing the raw data to validating the identified model.

This chapter provides an overview of the model-based control design process and the steps in the process where you can use National Instruments software and hardware. This chapter also provides an overview of the system identification process you use to analyze a plant and identify a model that describes the plant. Finally, this chapter provides information about the two tools in the System Identification Toolkit that enable you to estimate system models.

Model-Based Control Design Process

The model-based design process involves modeling a plant, analyzing and synthesizing a controller for the plant, simulating the plant and controller, and deploying the controller. While the System Identification Toolkit provides solutions for analyzing raw data and creating plant models, National Instruments also provides solutions for the other three components in the process, as shown in Figure 1-1.

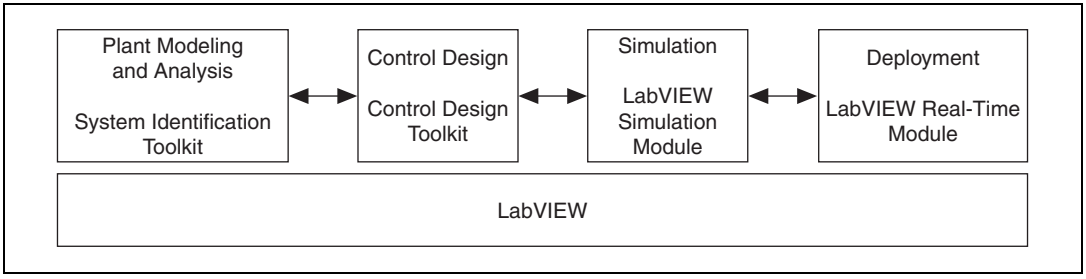

Figure 1-1. Plant Modeling, Control Design, Simulation, and Deployment

Analyzing Data and Creating a Dynamic System

In the initial phase of the design process, you must obtain a mathematical model of the plant you want to control. One way to obtain a model is by using a numerical process known as system identification. This process involves acquiring data from a plant and then numerically analyzing stimulus and response data to estimate the parameters of the plant.

National Instruments provides data acquisition (DAQ) and modular instrumentation software and hardware that you can use to stimulate and measure the response of the plant. You then can use the System Identification Toolkit to estimate and create accurate mathematical models of the plant. You can use the toolkit to create discrete, linear models of systems based on measured stimulus and response data.

System identification is a process that includes acquiring, formatting, processing, and identifying mathematical models based on raw data from a real-world system. You then validate that the resulting model fits the observed system behavior. If the results are unsatisfactory, you revise the parameters and iterate through the process. Figure 1-2 shows a typical system identification flowchart.

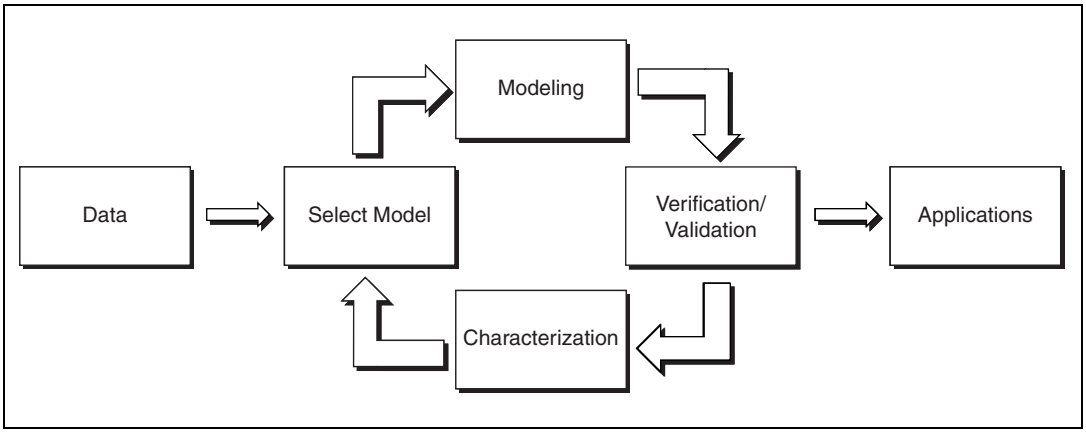

Figure 1-2. System Identification Application Flowchart

Real-world systems seldom have one model that perfectly describes all the observed behaviors of the system. Because system identification involves many variables—such as sampling frequency, type of mathematical model, model order, and so on—you usually have a number of models you can use. Each model describes the behavior of the system to some extent or in a particular mode of operation.

Furthermore, multiple applicable algorithms might be available for the same model. The algorithms you select depend on the model structure, stochastic assumptions, and numerical properties of the algorithm. The System Identification Toolkit includes different adaptive schemes for recursive system identification and different algorithms for auto-regression (AR) model estimation. Refer to Chapter 3, *Nonparametric Model Estimation Methods*, and Chapter 4, *Parametric Model Estimation Methods*, for information about the various estimation methods that the System Identification Toolkit supports.

Designing a Controller

In the second phase of the design process, you synthesize and analyze a controller. The LabVIEW Control Design Toolkit provides a set of VIs for classical and modern linear control analysis and design techniques. With these VIs you can create and analyze linear time-invariant system models and design automatic control systems.

You can analyze the plant model you identified using the System Identification Toolkit using the Control Design Toolkit. The Control Design VIs help you determine an appropriate controller structure. You then can synthesize a controller to achieve the desired performance criteria of the closed-loop system based on the dynamic behavior of the plant and/or control system. Finally, you can analyze the overall closed-loop system by combining the controller with the identified plant model.

Simulating the Dynamic System

In the third phase, you want to simulate the dynamic system. The LabVIEW Simulation Module allows you to simulate dynamic systems in LabVIEW. You can investigate the time response of the dynamic system to complex, time-varying inputs before deploying a controller. For this process, you can use a simple linear time-invariant model, a higher order model, or a nonlinear model of the plant.

Deploying the Controller

The last stage of the design process is to deploy the controller to a real-time target. LabVIEW and the LabVIEW Real-Time Module provide a common platform that you can use to implement or prototype the embedded control system. You also can use the LabVIEW Simulation Module and the LabVIEW Real-Time Module as the platform for implementing the control system.

National Instruments also provides products for I/O and signal conditioning that you can use to gather and process data. Using these tools, which are built on the LabVIEW platform, you can experiment with different approaches at each stage in the design process and quickly identify the optimal design solution for an embedded control system.

Refer to the National Instruments Web site at `ni.com` for information about these National Instruments products.

Developing Models with the Toolkit

The System Identification Toolkit provides a library of VIs and an assistant for developing models of a system based on a large set of raw data. Both tools enable you to complete the entire system identification process from analyzing the raw data to validating the identified model.

System Identification Assistant

Without prior knowledge about programming in LabVIEW, you can use the System Identification Assistant to develop a model that reflects the behavior of a certain dynamic system. You access the System Identification Assistant through the NI Express Workbench. The Express Workbench is a new framework that can host multiple interactive National Instruments tools and assistants.

Using the System Identification Assistant, you can create a project that encompasses the whole system identification process. In one project, you can load or acquire raw data into the System Identification Assistant, preprocess the data, estimate a model that describes the system, and then validate the accuracy of the model. The Express Workbench provides windows in which you can immediately see the raw data, the response data, the estimated model, the validation results, and the mathematical equations that describe the model.

After creating this project in the Express Workbench, you can convert the project to a LabVIEW block diagram and customize the block diagram in LabVIEW. This conversion enables you to enhance the capabilities of the application. Refer to the *NI Express Workbench Help*, available in the NI Express Workbench environment by selecting **Help»Express Workbench Help**, for more information about using the assistant to develop models.

System Identification VIs

The System Identification Toolkit also provides VIs that you can use to preprocess raw data from a dynamic system and develop a model that reflects the behavior of that system. The Data Preprocess VIs enable you to analyze the response of a plant or dynamic system to a certain stimulus. After analyzing the data, you can use the Parametric Model Estimation, Nonparametric Model Estimation, or the Recursive Model Estimation VIs to estimate a model for the plant or dynamic system. Finally, you can use the Model Validation or Model Presentation VIs to determine whether the model accurately describes dynamics of the identified system.

These VIs enable you to customize a LabVIEW block diagram to achieve specific goals. You also can use other LabVIEW VIs and functions to enhance the functionality of the application. Unlike creating a project with the System Identification Assistant, creating a LabVIEW application using these VIs requires basic knowledge about programming in LabVIEW. Refer to the *LabVIEW User Manual* and the *Getting Started with LabVIEW* manual for more information about the LabVIEW programming environment.

2

Acquiring and Preprocessing Data

The first step in identifying an unknown system is data acquisition. You can acquire data from NI data acquisition hardware and software or you can use data from a pre-stored file. For verification and validation reasons, you need to acquire two sets of input-output data samples or split the data into two sets. You use one set of samples to estimate the mathematical model of the system. You use the second set of samples to validate the resulting model. If the resulting model does not meet the predefined specifications, such as the mean square error (MSE), modify the settings and re-verify the resulting model with the data sets.

After acquiring the data, you need to preprocess the raw data samples. Preprocessing involves steps such as removing trends, filtering noise, and so on. The LabVIEW System Identification Toolkit provides Data Preprocess VIs that enable you to analyze the raw data and determine whether the data accurately reflects the response of the system you want to identify.

This chapter briefly describes the data acquisition process and the assumptions the System Identification Toolkit makes. This chapter also describes how to preprocess raw data using the Data Preprocess VIs. Refer to the *LabVIEW Help*, available by selecting **Help»VI, Function, & How-To Help**, for more information about the Data Preprocess VIs.

Acquiring Data from a System

One of the biggest advantages of using the System Identification Toolkit is the integration with LabVIEW, NI data acquisition hardware, and NI-DAQ. Refer to the *LabVIEW Measurements Manual* for information about setting up and configuring a data acquisition system, and how to use LabVIEW to acquire data samples.

If the data samples are in a pre-stored data file, you can use LabVIEW to import the data from the file. Refer to the *LabVIEW User Manual* about information on the file formats LabVIEW supports.

Identifying a system involves a number of choices with regard to the system output signals you want to measure and the input signals you want to manipulate. The choices you make about how to manipulate system inputs, types of signal conditioning, signal ranges, and sampling behavior affect the validity of the model you obtain. You can use different modeling techniques on the same experimental data set, but if the data set does not reflect the behavior of interest then you need to acquire a more descriptive data set.

Because the system identification process is often an experimental process, it is often time consuming and possibly costly. Therefore, you must think about the design of process prior to experimenting with various identification techniques. The following sections describe the various data acquisition and system stimulation assumptions you must consider before identifying a system model. These sections also provide information about the trade-offs associated with each choice.

Accounting for Factors that Influence a System

The key to the system identification process is having some knowledge of the system for which you want to identify a model. This knowledge provides the basis for determining which signals are outputs for determining sensor placement and which signals are inputs that you can use to excite the system. Simple tests might be necessary to determine influences, coupling, time delays, and time constants to aid in the modeling effort.

Also you need to consider signals that are not directly capable of being manipulated but still affect the system. You need to include those signals as inputs to the system model. For example, consider the effect of wind gusts on the pitch dynamics of an airplane. The airplane responds in pitch to the elevator angle as a direct input. A wind gust affects the pitch of an airplane, which in turn influences the dynamics of the airplane, but the wind gust is not directly adjustable. To create an accurate model of the airplane, you might want to include wind gusts as an input variable.

Choosing a Stimulus Signal

The choice of stimulus signals has an important role in the observed system behavior and the accuracy of the estimated model. These signals determine the operating points of the system. While the system under test often limits the choice of signals, you want an input signal to exhibit certain characteristics to produce a model that provides the information needed for developing an accurate model. The following sections summarize these characteristics.

- To obtain meaningful dynamic behavior, you must test the system under conditions similar to the actual operating conditions. When you complete experiments in these conditions, you identify the system in the same conditions under which you will implement the resulting model. This criterion is extremely important for nonlinear systems.
- You want the inputs to the system under test to excite the system. Exciting the system is dependent on the spectrum of the input signal. Specifically, you must excite the system with an input frequency similar to the frequency at which such inputs change during normal operations.
- You want the amplitude of the step input to cover a wide range of variations. Therefore, in the data you use for model estimation, you need to cover the normal operation range of system inputs, especially when you use the calculated model for model-based control. To cover the normal operation range, you can combine the positive and negative step changes of different magnitudes in the system inputs.
- You want the input signal to deliver as much input power to the system as possible. However, in the real-world, you must ensure that this input power stays within the limits of the physical system. The crest factor C_f, defined by the following equation, describes this property.

$$C_f^2 \equiv \frac{\max_t u^2(t)}{\lim_{N\to\infty} \frac{1}{N} \sum_{t=1}^{N} u^2(t)}$$

The smaller the crest factor the better the signal excitation resulting in larger total energy delivery and enhanced signal-to-noise ratio. The theoretical lower bound for crest factor is 1.

Common Stimulus Signals

The system response data is dependent on the physics of the system you want to study. Some systems tend to respond faster than others, and never reach steady state. Other systems have large time constants and delays. For these reason, it is important to define a stimulus signal that provides enough excitation to the system such that the response captures the important features of the system dynamics. The following sections describe common stimulus signals you can use in different process applications.

Filtered Gaussian White Noise

Filtered Gaussian white noise is a simple signal that can generate virtually any signal spectra in conjunction with the proper linear filtering. The theoretical crest factor C_f for a Gaussian is infinite, but clipping the Gaussian amplitude to the input signal limits results in a corresponding reduction in crest factor while minimally affecting the generated spectrum.

Figure 2-1 shows an example of a Filtered Gaussian white noise.

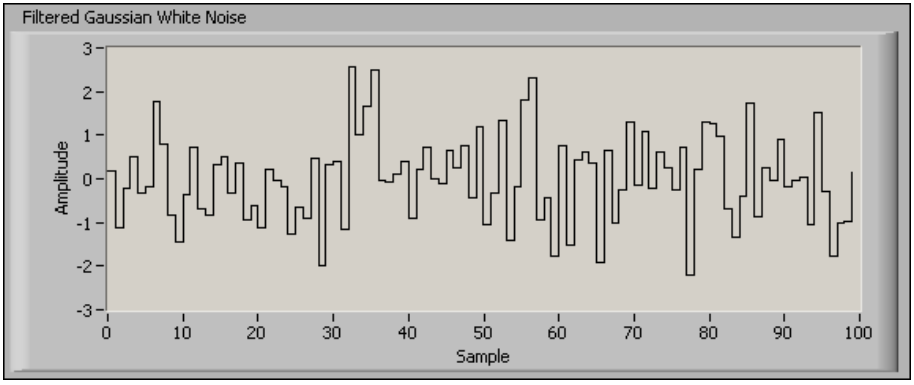

Figure 2-1. Filtered Gaussian White Noise

Random Binary Signal

A random binary signal is a random process that can assume one of two possible values at any time. A simple method of generating a random binary signal is to take Gaussian white noise, filter it for the desired spectra and then convert it to a binary signal by taking the sign of the filtered signal. The desired spectra is a function of the system time constraints. The appropriate scaling must provide a meaningful response to the system, well above the noise level.

You can scale the signal to any desired amplitude. The resulting signal has a minimum crest factor C_f of 1. Some differences in the resulting spectra are expected so off-line analysis of the signal should be performed.

Binary signals are useful for identifying linear systems. However, the dual-level signal do not allow for validation against nonlinearities. If a system is nonlinear, you can use an interval of input corresponding to the desired operating point. You might need to work with more than two input levels in these cases. You can combine multiple binary signals of different levels to form the stimulus signal.

Figure 2-2 shows an example of a random binary signal.

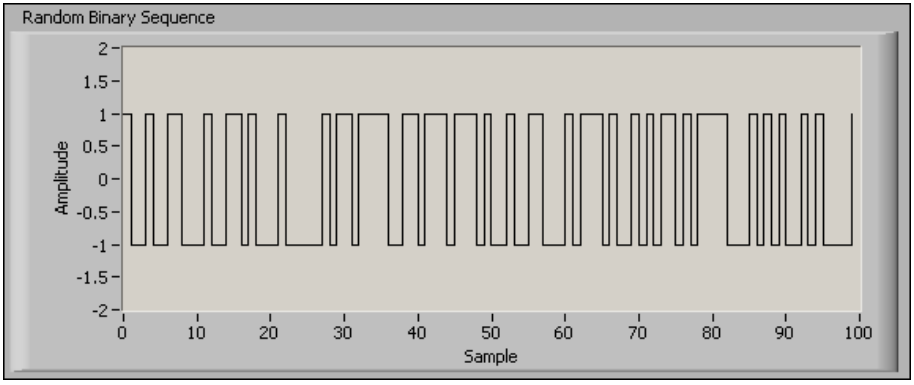

Figure 2-2. Random Binary Sequence

Pseudo-Random Binary Sequence

A Pseudo-Random Binary Sequence, also known as Maximal Length Sequence (MLS), is a periodic, deterministic signal with properties similar to white noise. You often generate a pseudo-random binary sequence using an n-bit shift register with feedback through an exclusive-OR function. While appearing random, the sequence actually repeats every $2n - 1$ values.

When using a whole period, the pseudo-random binary sequence has special mathematical advantages that make it attractive as a stimulus signal. In particular, you can attribute variations in response signals between two periods of the stimulus to noise due to the periodic nature of the signal. Also, like the white random binary noise, the pseudo-random binary sequence has a low crest factor C_f.

Figure 2-3 shows an example of a pseudo-random binary sequence.

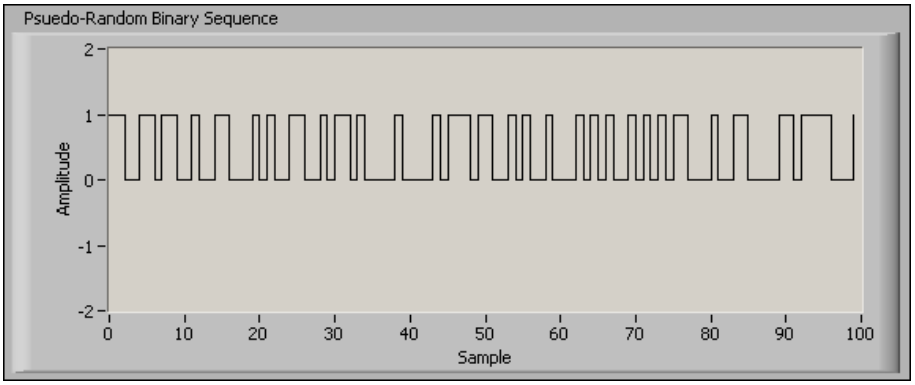

Figure 2-3. Pseudo-random Binary Sequence

Chirp Waveform

The chirp, also known as a swept sine wave, is a sinusoid waveform with a frequency that varies continuously over a certain range of values $\omega_1 \leq \omega \leq \omega_2$ for a specific period of time $0 \leq t \leq T$. The resulting signal has a crest factor C_f of $\sqrt{2}$ and is easily modified to excite specific signal spectra.

In comparison to other signals, like the white noise stimulus, a chirp waveform is easier to generate and control. Figure 2-4 shows an example of a chirp waveform.

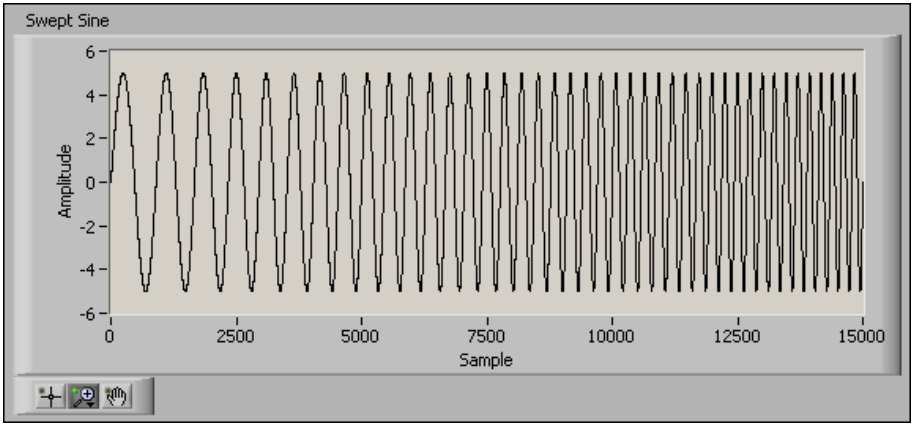

Figure 2-4. Chirp, or Swept Sine Wave

Selecting a Sampling Rate

The time constants of a system influence the selection of a sampling rate. Sampling at rates substantially greater than the system bandwidth leads to data redundancy, numerical issues, and modeling of high frequency artifacts likely due to noise. Sampling at rates slower than system dynamics leads to difficulties determining an accurate system model and problems introduced by aliasing. You can use an anti-aliasing filter to counter the effects of aliasing. Refer to the *Applying an Anti-Aliasing Filter* section for information about anti-aliasing filters.

A common rule of thumb is to sample signals at 10 times the bandwidth of the system or the bandwidth of interest for the model. If uncertainty exists in the system bandwidth and a fast data acquisition environment is available, you can sample as fast as possible, then use a digital filter and decimation to reduce the sampling rate to the desired value. Decimation is a form of downsampling the data set. Refer to the *Filtering and Downsampling* section for information about filtering and downsampling a data set.

Applying an Anti-Aliasing Filter

According to the Nyquist sampling theorem, the sampling rate must be greater than twice the maximum frequency component of the signal of interest. In other words, the maximum frequency of the input signal must be greater than half the sampling rate.

This criterion, in practice, is often difficult to ensure. Even if you are sure that the measured signal has an upper limit on its frequency, external factors such as signals from the powerline interference or radio stations, can contain frequencies higher than the Nyquist frequency. These frequencies might then alias into the frequency range of interest and give you inaccurate results.

To ensure that you limit the frequency content of the input signal, add a low-pass filter, which is a filter that passes low frequencies but attenuates the high frequencies, before the sampler and the analog to discrete converter. This filter is an anti-aliasing filter because by attenuating the frequencies greater than the Nyquist frequency, the filter prevents the sampling of aliased components. When you use a filter before the sampler and analog to discrete converter, the anti-aliasing filter is an analog filter. Using an analog filter satisfies the Nyquist sampling theorem.

Similarly, you can use a digital filter to remove frequency content above the system bandwidth and then decimate or downsample the data to the desired sampling rate.

Preprocessing Data from a System

A number of preprocessing techniques ensure that the incoming data samples are free from external noise, scaling problems, outliers, and other corruptions. These preprocessing techniques include the following methods:

- Visually inspecting data
- Removing offsets and trends
- Removing outliers
- Filtering and downsampling

Validating the quality of the data at each step in the preprocessing procedure is important in ensuring that you accurately identify a model in later steps of the system identification process.

The following sections describe these preprocessing techniques and how you can use the System Identification Toolkit to apply these techniques.

Visually Inspecting the Data

Visual inspection of the data is the best way to detect major signal corruptions or errors—such as outliers, clipped saturation, or quantization effects—that occur during acquisition or preprocessing. You also can plot the data waveform and the spectral density function of the data to discover periodic disturbances.

Traditionally, you examine data samples either in the time domain or the frequency domain. An effective approach is to display the data in the joint time-frequency domain, which provides a better understanding about the measured signals. Refer to the *Signal Processing Toolset User Manual*, available at `ni.com/manuals`, for more information about joint time-frequency domain techniques for data processing.

Removing Offsets and Trends

The SI Remove Trend VI enables you to remove offsets and trends from the raw data set. You can specify which you want to remove using the input **trend type**. The following sections describe the difference between removing offsets and removing trends.

Removing Offsets

The estimated system model is a linearized version of the true system around the operating point. You must subtract the operating points from the raw data samples because linearization is done with respect to the signal values relative to the operating point, which is the offset level of the signal.

Figure 2-5 shows an example of removing the offset level of a signal. The goal of the water tank is to keep the water level at six meters. The **Water level record** graph shows that the water level changes in the vicinity of the operating point of six meters. If you use the water level record for system identification, you must remove the six meter operating point value.

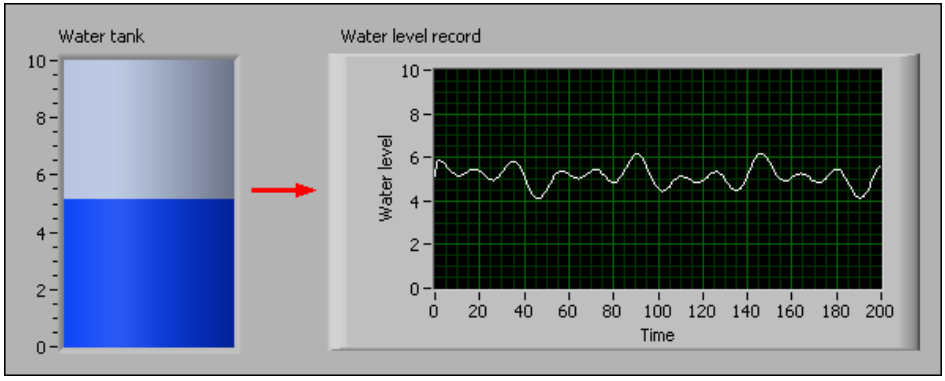

Figure 2-5. Operating Point of Water Tank

The SI Remove Trend VI enables you to remove the offset from the raw data set. You must set the **trend type** to **mean** to use this preprocessing technique.

Removing Trends

External influences might add some low frequency or periodic components, which are not relevant to the specific modeling problem, to the data. Examples of external influences include variations due to the 24-hour day cycle in power plants, seasonal influences in biological and economical systems, thermal expansion in rolling mills, 50 Hz and 60 Hz powerline interferences, and so on. The amplitude of these trends can be large and can corrupt the results of signal analysis and parametric identification algorithms.

The SI Remove Trend VI provides a way for you to remove trends from the raw data set. You must set the **trend type** to **linear** to use this preprocessing technique.

Outliers

Various unexpected events, such as an abnormal pulse, a temporary sensor failure, or transmitter failure, can corrupt the raw data samples. These disturbances, or outliers, can severely distort the resulting model estimation. However, you often can recognize outliers by visually inspecting the data, as shown in Figure 2-6.

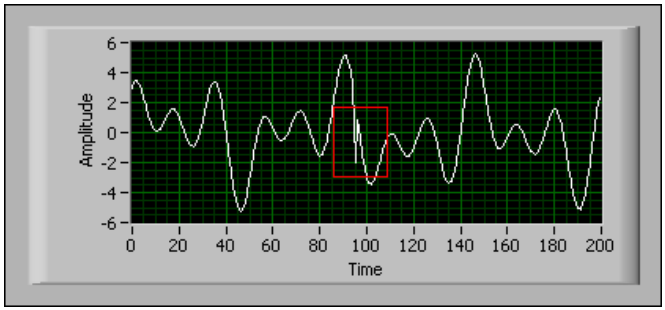

Figure 2-6. Data with Outliers

Visually inspecting the graph tells you that the data acquired between 85–100 seconds is abnormal. When preprocessing data, you want to remove all outliers in the data set. You must remove the outliers manually.

Filtering and Downsampling

You might be interested in only a specific frequency range of the frequency response for a model. You can filter and enhance the data in the frequency range to improve the fit in the regions of interest. If the sampling frequency is much higher than the bandwidth of the system, the sampling frequency might substantially increase the computation burden for complicated identification algorithms. You can decrease the sampling frequency by taking every n^{th} sample to construct a new downsampled data set. Applying an anti-alias filter on the data before downsampling prevents corruption of the downsampled data set.

You can use the SI Lowpass Filter VI or the SI Bandpass Filter VI to apply a lowpass or bandpass filter, respectively, to the data from the system. You then can use the SI Downsampling VI to reduce the number of samples in the data set.

After preprocessing the data you acquired from a dynamic system, the result is a data set that you can use to estimate a model that reflects the system dynamics. Refer to Chapter 3, *Nonparametric Model Estimation Methods*, for information about the nonparametric model estimation

methods that use the impulse response and frequency response. Refer to Chapter 4, *Parametric Model Estimation Methods*, for information about the parametric model structures and the parametric model estimation methods. Refer to Chapter 5, *Recursive Model Estimation Methods*, for information about recursive model estimation methods.

Data Scaling

In multiple-input multiple-output (MIMO) systems, it is common to have inputs and outputs of different amplitude ranges. Such a diversity in amplitudes can make the model estimation calculation ill-conditioned, which deteriorates the precision of the dynamic response. For example, consider the values A and B in Figure 2-7. Valves A and B operate between 0–100% and 50–60% opening range, respectively.

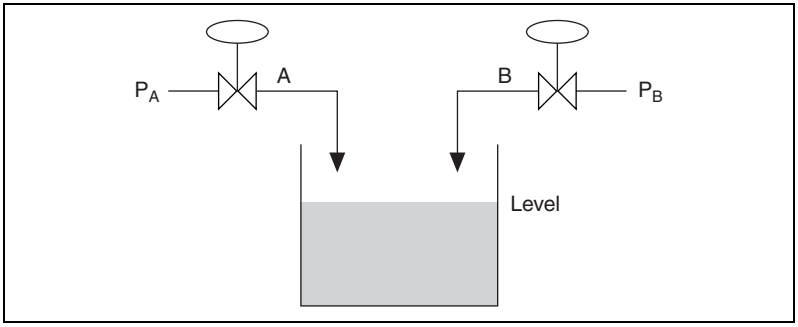

Figure 2-7. Tanks

The pressure in their respective stream lines are P_A and P_B. Assume that P_B can be much larger than P_A, you might need to normalize the range of operation of valve B for numerical robustness. You can use the following relationship to normalize the range of operation.

$$\frac{\Delta Level}{\Delta \%A} \approx \frac{\Delta Level}{(\Delta \%B - 50)10}$$

The SI Normalize VI ensures that all stimulus and response signals have a zero mean and unit variance over the sample data range used for model estimation. This process standardizes the range of the equation for all signals considered for model estimation. This data preprocessing step considers all inputs and outputs equally important from the numerical calculation viewpoint.

3

Nonparametric Model Estimation Methods

After acquiring and preprocessing the data from a linear time-invariant system, the next step in the system identification process is to estimate the model. The two most common techniques to estimate models that represent linear time-invariant systems are nonparametric estimation and parametric estimation. This chapter describes the nonparametric estimation methods.

You can describe linear time-invariant models with transfer functions or by using the impulse responses or frequency response of the system. The impulse response and frequency response are two ways of estimating a nonparametric model. The impulse response reveals the time-domain properties of the system, such as time delay and damping, whereas the frequency response reveals the frequency-domain properties, such as the natural frequency of a dynamic system.

Nonparametric model estimation is simple and more efficient, but often less accurate, than parametric estimation. However, you can use a nonparametric model estimation method to obtain useful information about a system before applying parametric model estimation. For example, you can use nonparametric model estimation to determine whether the system requires preconditioning, what the time delay of the system is, what model order to select, and so on. You also can use nonparametric model estimation to verify parametric models. For example, you can compare the Bode plot of a parametric model with the frequency response of the nonparametric model. Refer to Chapter 4, *Parametric Model Estimation Methods*, for information about parametric model estimation methods.

The LabVIEW System Identification Toolkit uses correlation analysis method to estimate the impulse response and spectral analysis method to estimate the frequency response. The following sections describe the impulse response and frequency response methods.

Impulse Response

An impulse input, as shown in Figure 3-1, to a dynamic system is defined differently depending on whether the system is discrete or continuous. For a continuous dynamic system, an impulse input is a unit-area signal with an infinite amplitude and infinitely small duration occurring at a specified time. At all other times, the input signal value is zero. For a discrete system, an impulse is a physical pulse that has unit amplitude at the first sample period and zero amplitude for all other times.

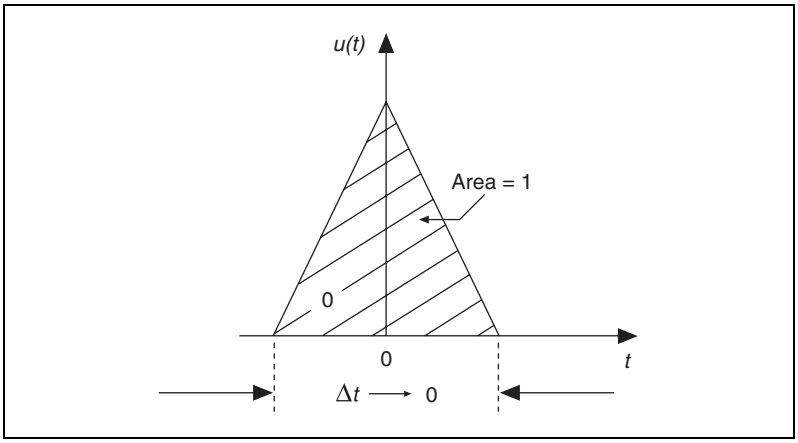

Figure 3-1. Impulse Response

Because the impulse signal excites all frequencies and the duration of this signal is infinitely small, you can see the natural response of the system.

Figure 3-2 shows that the impulse response of a linear time-invariant system is equal to the output $y(n)$ of the system when you apply an impulse signal to the input $u(n)$ of the system. The impulse response provides the complete characteristic information of a system.

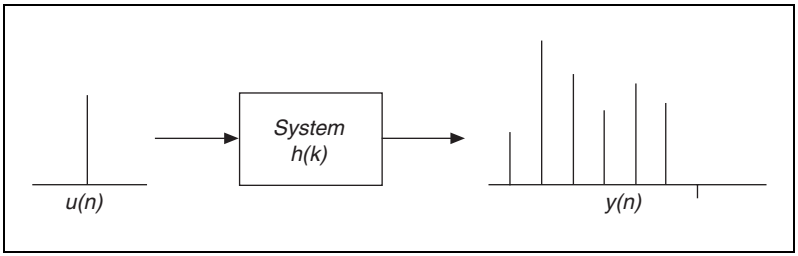

Figure 3-2. Impulse Response Definition

If you know the impulse response $h(k)$ and the input signal $u(n)$ of a system, then you can compute the output $y(n)$ of the system by using the following equation.

$$y(n) = \sum_{k=0}^{\infty} u(n-k)h(k) + e(n)$$

$e(n)$ is the disturbance of the system.

According to impulse response theory, when you apply a Dirac delta function to the input of a system, the output of the system is the impulse response. However, generating an ideal Dirac delta function is unrealistic. You can think of the Dirac delta function $\delta(x)$ as a function that has the value of infinity for $x = 0$, the value zero elsewhere, and a total integral of one. You can think of the graph of the delta function as the following whole x-axis and positive y-axis.

If you apply an approximate impulse with a small duration to the input of a system, the output of the system is the approximation of the impulse response of the system. The smaller the duration of the impulse, the closer the output of the system is to the true impulse response. However, an impulse carries little energy and might not excite the system, and noise might corrupt the output of the system. An impulse with a large amplitude and duration can improve the signal-to-noise ratio of the output signal. However, the large amplitude impulse can damage the hardware of the system, and a long-duration impulse leads to inaccuracy. For these reasons, the System Identification Toolkit uses the correlation analysis method to estimate the impulse response.

Correlation Analysis

The correlation analysis method uses the cross correlation between the input and output signals as an estimation of the impulse response. The input signal must be zero-mean white noise with a spectral density that is equally distributed across the whole frequency range. The SI Estimate Impulse Response VI can prewhiten signals that are not white noise.

Assuming the input $u(n)$ of the system is a stationary, stochastic process and statistically independent of the disturbance $e(n)$, the following equation is true.

$$R_{uy}(\tau) = \sum_{k=0}^{\infty} R_{uu}(k-\tau)h(k)$$

R_{uy} represents the cross correlation function between the stimulus signal $u(n)$ and the response signal $y(n)$, as defined by the following equation.

$$R_{uy}(\tau) = \frac{1}{N} \sum_{n=min(\tau,\ 0)}^{N-max(\tau,\ 0)-1} y(n+\tau)u(n)$$

R_{uu} represents the autocorrelation of the stimulus signal $u(n)$, as defined by the following equation.

$$R_{uu}(\tau) = \frac{1}{N} \sum_{n=0}^{N-\tau-1} u(n+\tau)u(n)$$

N is the number of data points. If the stimulus signal is a zero-mean white noise signal, the autocorrelation function reduces to the following equation.

$$R_{uu}(\tau) = \sigma_u^2 \delta(\tau)$$

where σ_u is the standard deviation of the stimulus white noise and $\delta(\tau)$ is the Dirac function. Substituting $R_{uu}(\tau)$ into the cross correlation function between the stimulus signal $u(n)$ and the response signal $y(n)$ yields the following equation.

$$R_{uy}(\tau) = \sigma_u^2 \sum_{k=0}^{\infty} \delta(k-\tau)h(k) = \sigma_u^2 h(\tau)$$

You can rearrange the terms of this equation to obtain the following equation defining the impulse response $h(k)$.

$$h(k) = \frac{R_{uy}(k)}{\sigma_u^2}$$

Prewhitening

The correlation analysis method that estimates the impulse response is useful only when the input signal $u(n)$ is a zero-mean white noise signal. However, the input signal is not white noise in most real-world applications. The input $u(n)$ and output $y(n)$ signals therefore must be preconditioned before you apply the correlation analysis method.

Prewhitening is a preconditioning technique for the correlation analysis method. Prewhitening involves applying a filter to the input signal $u(n)$ and the output signal $y(n)$ to obtain a prewhitened input signal $u'(n)$ and a prewhitened output signal $y'(n)$. If the filter is well designed such that $u'(n)$ is white noise, you can perform a correlation analysis on $u'(n)$ and $y'(n)$ to estimate the impulse response. The impulse response that you estimate with $u'(n)$ and $y'(n)$ is equivalent to the impulse response that you estimate with $u(n)$ and $y(n)$ because the following equation remains true.

$$y'(n) = \sum_{k=0}^{\infty} u'(n-k)h(k) + e(n)$$

You now must design the prewhitening filter so that $u'(n)$ is white noise. The SI Estimate Impulse Response VI uses an AR model for this purpose. Refer to Chapter 4, *Parametric Model Estimation Methods*, for more information about AR model estimation.

Accuracy of the Impulse Response

The accuracy of the impulse response estimation using the correlation analysis method depends on the performance of the prewhitening filter, specifically whether the filter produces a white noise result $u'(n)$ for $u(n)$. The performance of the filter depends on the signal and the AR order of the filter. The rule of thumb for selecting the AR order is trial-and-error. If $u'(n)$ is not white enough, the result from the correlation method is not reliable. You can increase the AR order to improve the accuracy of the impulse response.

The SI Estimate Impulse Response VI provides the outputs **whiteness test** and **rejected?** to indicate whether you have properly set the AR order and consequently whether the impulse response estimation is reliable. The following example shows how the whiteness property of the input signal affects the correlation analysis method and how to use the outputs **whiteness test** and **rejected?** to justify the impulse response estimation.

Figure 3-3 shows the front panel of a VI that simulates a system defined by the following equation.

$$y(n) = 0.2u(n) + 0.8u(n-1) + 0.3u(n-2)$$

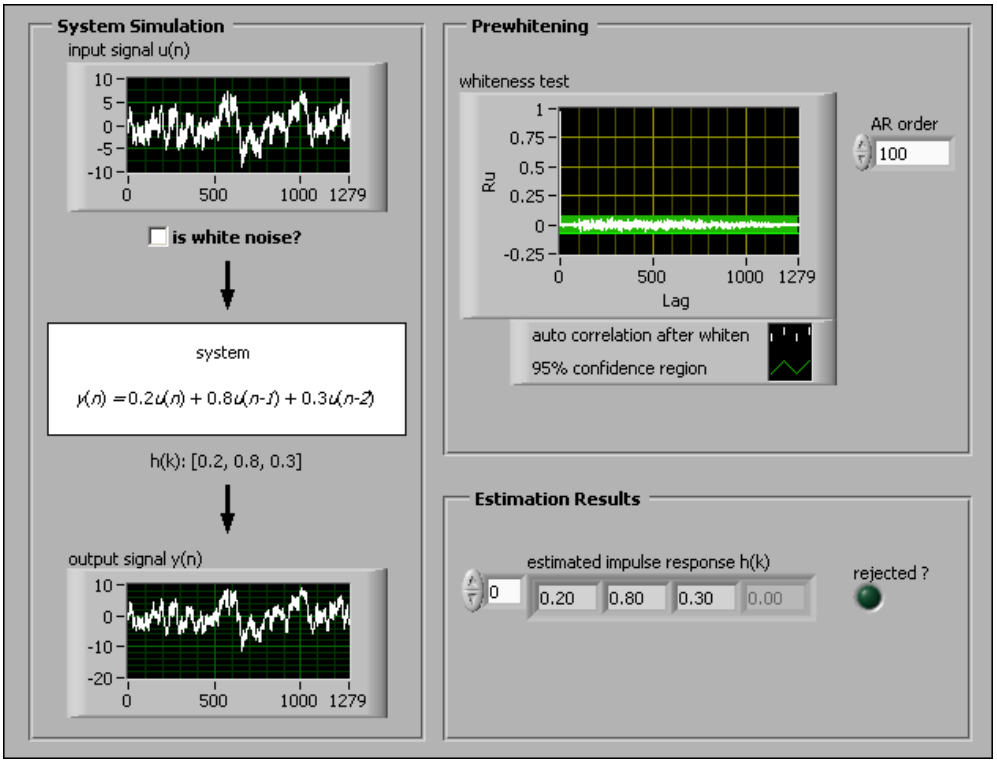

Figure 3-3. Front Panel of Prewhitening Example VI

Figure 3-4 shows the block diagram of this VI. This example VI demonstrates the accuracy of the impulse response estimation in the following circumstances:

- Zero-mean, pseudo-white noise input signal without prewhitening
- Zero-mean, pseudo-white noise input signal with prewhitening
- Non-zero-mean, white noise input signal without prewhitening
- Non-zero-mean white noise input signal with prewhitening

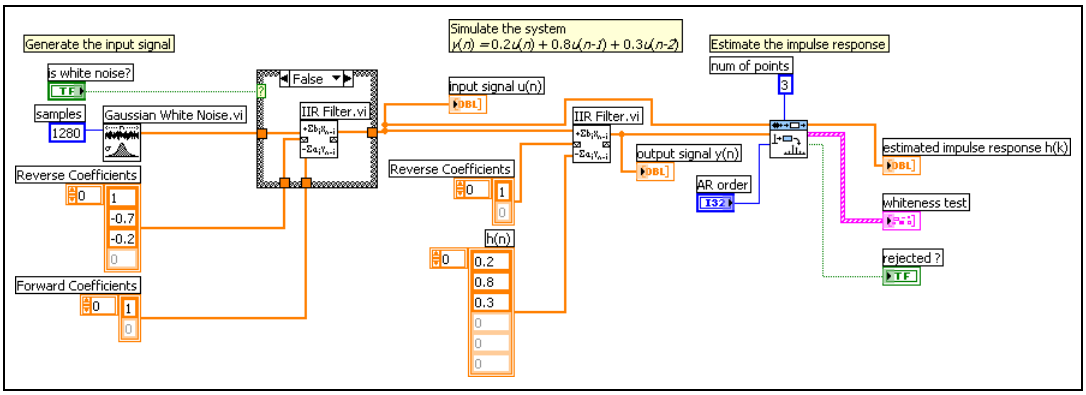

Figure 3-4. Block Diagram of Prewhitening Example VI

In this example VI, the **is white noise?** checkbox determines whether the SI Estimate Impulse Response VI generates zero-mean white noise as an input to the system. When you place a checkmark in the **is white noise?** checkbox and run the VI, the generated input signal is zero-mean white noise, and the estimated impulse response closely approximates the true impulse response. When you do not place a checkmark in the **is white noise?** checkbox, the generated input signal is not zero-mean white noise. As a result, the estimated impulse response is different from the true impulse response. These results indicate that the correlation analysis method is accurate and reliable when the input signal is zero-mean white noise.

The **AR order** box determines the level of prewhitening. When **AR order** equals 0, the SI Estimate Impulse Response VI does not apply prewhitening to the system. When **AR order** is small and you do not place a checkmark in the **is white noise?** checkbox, the variance of the impulse response is large because the input signal is not always white noise. The greater the value of **AR order**, the better the VI whitens the signal, but the more computation time and memory the VI requires.

The **whiteness test** indicator of this VI shows whether the input is zero-mean white noise. This indicator displays the autocorrelation of the stimulus signal after whitening. If most of the autocorrelation is within the confidence region, the input signal is well prewhitened, and the estimation of the impulse response is reliable. If the autocorrelation is outside of the confidence region, the estimation is unreliable. When the estimation is unreliable, **rejected?** is TRUE and indicates a 5% risk of rejecting an impulse response estimation that might be reliable.

If you apply proper prewhitening, the correlation analysis method is accurate and reliable for any input signal. To obtain the best prewhitening settings, start with a small **AR order** value like 2 and observe the **whiteness test** and **rejected?** outputs of the SI Estimate Impulse Response VI. If necessary, increase the value of **AR order**. Generally, the smaller the bandwidth of the input signal, the larger the **AR order** you need. However, avoid setting the value of **AR order** greater than 500.

Selecting Impulse Response Length

Theoretically, the length of the impulse response might be infinite. For some systems, the impulse response quickly reaches zero, and the number of non-zero points is finite. For other systems, the impulse response never reaches zero. Realistically, you only can obtain the first *N* points of the impulse response due to limited signal length and limited memory size. Therefore, the SI Estimate Impulse Response VI has an input **num of points** to specify how many points of the impulse response to observe. You can set **num of points** to be as large as the signal length.

Applications of the Impulse Response

The impulse response not only indicates the stability and causality of the system, but it also provides information on properties such as the damping, dominating time constant, and time delay. Some of this information, such as the time delay, is useful for parametric model estimation. Therefore, you can use nonparametric impulse response estimation before parametric model estimation to help estimate the parameters. The following example demonstrates how to use the SI Estimate Impulse Response VI to estimate the impulse response and determine the time delay of a system.

Figure 3-5 shows the front panel of a VI that simulates a system defined by the following equation.

$$y(n) = 0.2u(n-2) + 0.8u(n-3) + 0.3u(n-4)$$

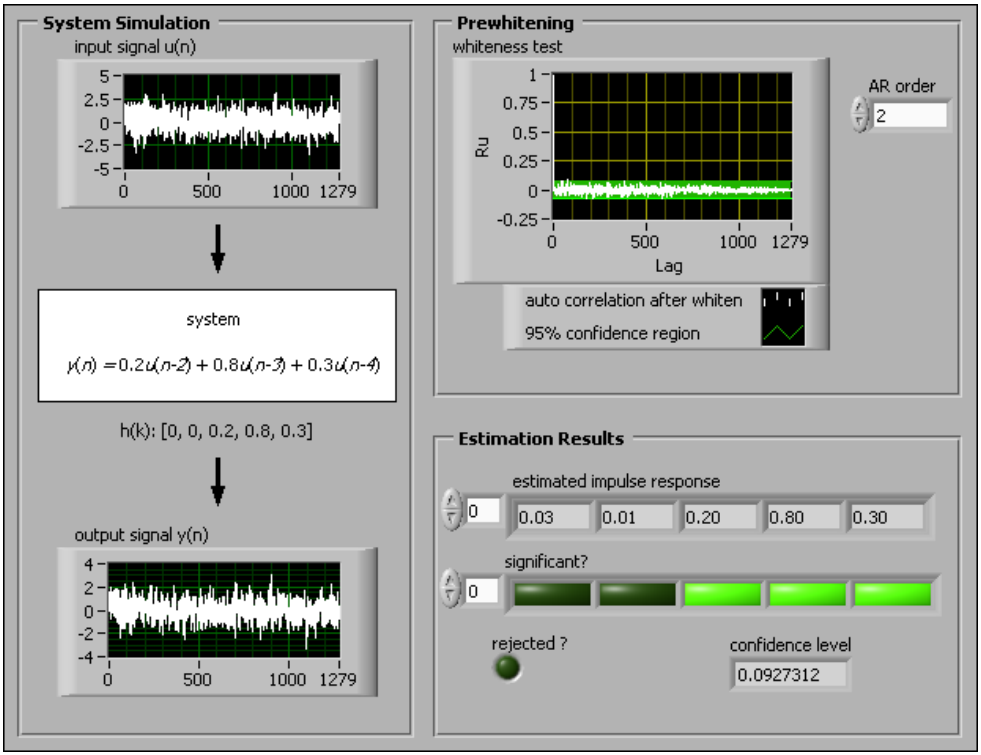

Figure 3-5. Front Panel of Time Delay Example VI

Figure 3-6 shows the block diagram of this VI.

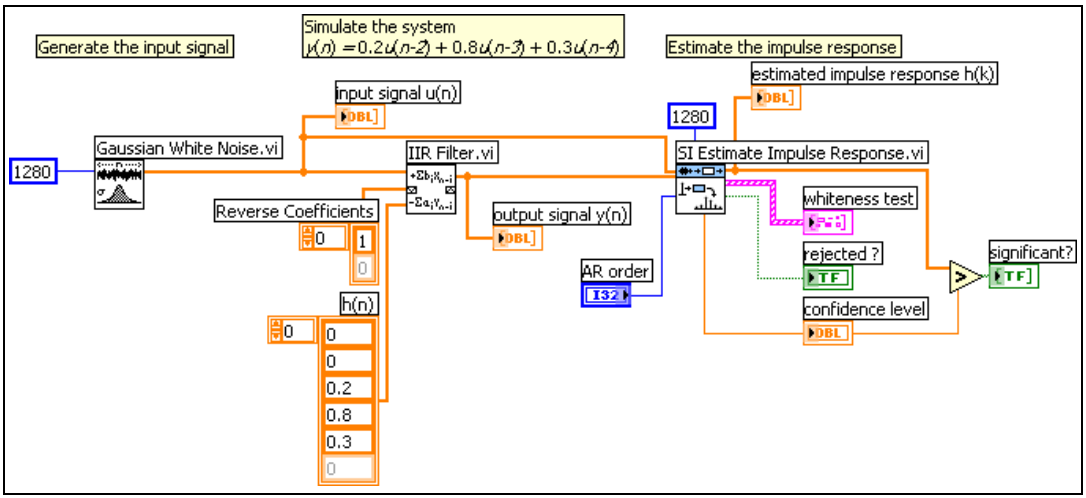

Figure 3-6. Block Diagram of Time Delay Example VI

In Figure 3-5, the two initial values of the **estimated impulse response** are smaller than the **confidence level**. You can have 99.0% confidence that values less than the **confidence level** are insignificant, and you can consider those values to be equal to 0. Therefore, you can conclude that the time delay of the system is 2 because the beginning of the first two values of the impulse response are zero.

Frequency Response

In theory, the results from impulse response estimation and the results from frequency response estimation are equivalent. For example, the Fourier transform of the impulse response $h(k)$, which you can compute using impulse response estimation, equals the frequency response $G(e^{j\omega})$. However, this equivalence does not hold in most real-world applications because of different preprocessing schemes in impulse response estimation and frequency response estimation.

The frequency response provides the complete frequency-domain characteristics of the system, including the passband and the natural frequency of the system. A sinusoidal input signal has the following general form:

$$u(t) = \sin(\omega_0 t)$$

For a linear time-invariant system, the response of a linear time-invariant system to a sinusoidal input also is a sinusoidal signal but potentially with a different magnitude and phase, as shown in the following equation.

$$y(t) = b\sin(\omega_0 t + \theta)$$

where b and θ are the magnitude and phase, respectively, of the frequency response of the system to an input sinusoidal frequency ω_0. If you apply input signals with a number of sinusoids at different frequencies, then you can obtain an estimate of the frequency response $G(\omega)$ of the system at those frequencies. The frequency response is a complex-valued sequence. The magnitude of $G(\omega)$ is the magnitude response of the system and the phase of $G(\omega)$ is the phase response of the system. This method of obtaining the frequency response is straightforward but takes a long time to complete and is sensitive to noise. For these reasons, the System Identification Toolkit uses the spectral analysis method to estimate the frequency response function.

Spectral Analysis Method

You can use the spectral analysis method with any input signal. However, the frequency bandwidth of the input signal must cover the range of interest.

Because the frequency response is the Fourier transform of the impulse response, applying the Fourier transform to both sides of the cross correlation function yields the following equation.

$$\Phi_{uy}(e^{j\omega}) = \Phi_{uu}(e^{j\omega})G(e^{j\omega})$$

$G(e^{j\omega})$ is the frequency response of the system. $\Phi_{uu}(e^{j\omega})$ is the auto-spectral density of the stimulus signal $u(n)$. $\Phi_{uy}(e^{j\omega})$ is the cross-spectral density between the stimulus signal $u(n)$ and the response signal $y(n)$.

You then can use the following equation to compute the frequency response $G(e^{j\omega})$.

$$G(e^{j\omega}) = \frac{\Phi_{uy}(e^{j\omega})}{\Phi_{uu}(e^{j\omega})}$$

You can compute $\Phi_{uu}(e^{j\omega})$ and $\Phi_{uy}(e^{j\omega})$ by applying a fast Fourier transform (FFT) to the autocorrelation R_{uu} and the cross correlation R_{uy}, respectively. As shown in the autocorrelation function R_{uu} and the cross

correlation function R_{uy}, described in the *Correlation Analysis* section, the number of data points you need to compute R_{uu} and R_{uy} decreases as the lag τ increases. Therefore, R_{uu} and R_{uy} become inaccurate for a large lag τ.

When computing $\Phi_{uu}(e^{j\omega})$ and $\Phi_{uy}(e^{j\omega})$, you can apply a lag window $w(\tau)$ to R_{uu} and R_{uy} before conducting the FFT operation to improve the accuracy of the frequency response estimation, as shown in the following equations.

$$\Phi_{uu}(e^{j\omega}) = \sum_{\tau=-N}^{N} R_{uu}(\tau) w_m(\tau) e^{-j\omega\tau}$$

$$\Phi_{uy}(e^{j\omega}) = \sum_{\tau=-N}^{N} R_{uy}(\tau) w_m(\tau) e^{-j\omega\tau}$$

The lag window approaches zero when the lag τ is large. The window weighs out the points of R_{uu} and R_{uy} with large lag τ, thereby improving the accuracy of the frequency response estimation. The SI Estimate Frequency Response VI uses a Hanning window as the lag window.

Refer to the book, *System Identification Theory for the User*[1], for information about using a Hanning window.

Accuracy of the Lag Window

The frequency response with the lag window, $G'(e^{j\omega})$, is equivalent to the moving average version of the frequency response without the lag window, $G(e^{j\omega})$. The average smooths the frequency response, but the smooth frequency response also can deviate more from the true frequency response. Adjusting the length of the lag window can balance the trade-off between variance and bias of the frequency response estimation. The larger the length of the lag window, the fewer points of $G(e^{j\omega})$ the SI Estimate Frequency Response VI averages to compute $G'(e^{j\omega})$, and hence the larger the variance and the smaller the bias of the frequency estimation.

[1] Ljung, L. 1999. *System Identification Theory for the User.* 2d ed. Prentice Hall.

The following example demonstrates how the length of the lag window affects the frequency response estimation. Figure 3-7 shows the front panel of a VI that simulates a system defined by the following equation.

$$y(n) - 1.46y(n-1) + 2.5y(n-2) - 1.46y(n-3) + y(n-4) = u(n) + 0.45u(n-1) + u(n-2)$$

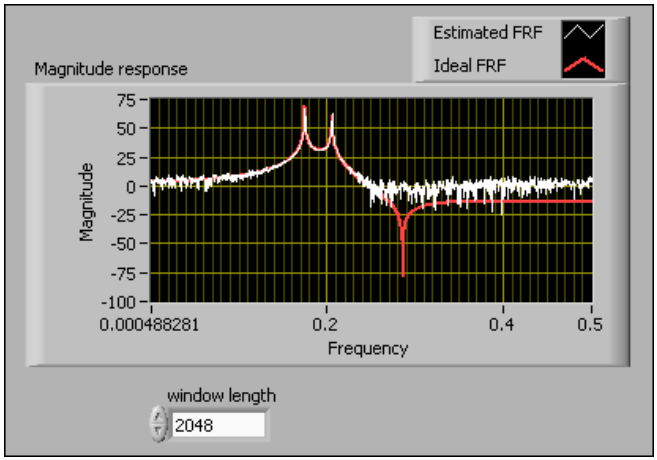

Figure 3-7. Front Panel of Lag Window Example VI

Figure 3-8 shows the block diagram of this VI.

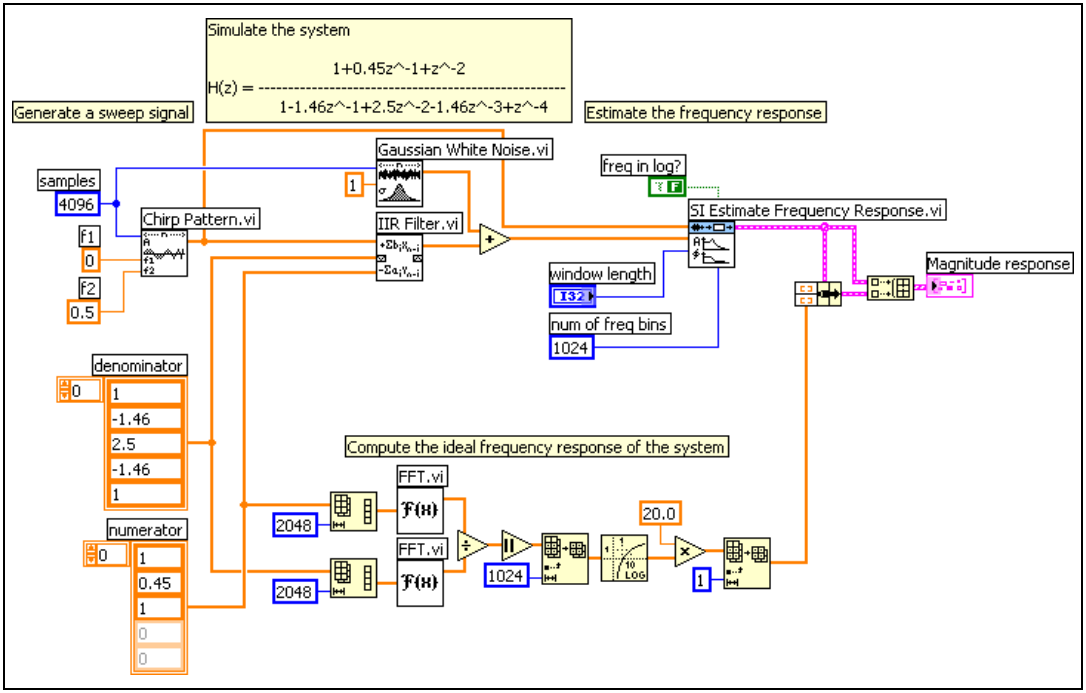

Figure 3-8. Block Diagram of Lag Window Example VI

In this example VI, the input signal $u(n)$ is a swept sine wave whose normalized frequency is from 0 to 0.5. The number of data points in the input signal is 4096. The length of the lag window therefore must be less than or equal to 4096. Figure 3-9 and Figure 3-10 show the resulting frequency responses when the window length is 4096 and 64 respectively.

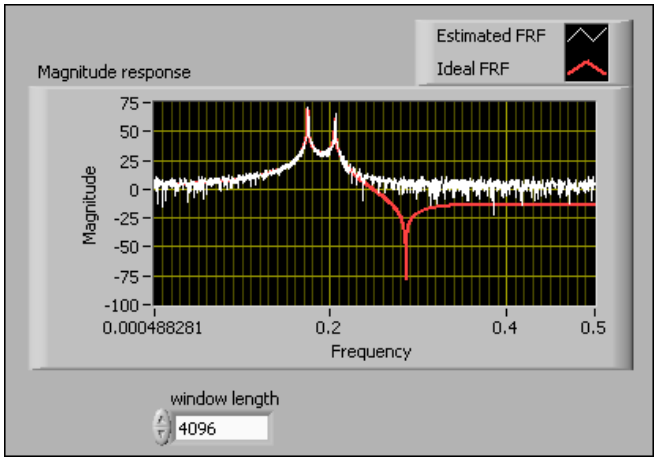

Figure 3-9. Frequency Response with Large Window Length

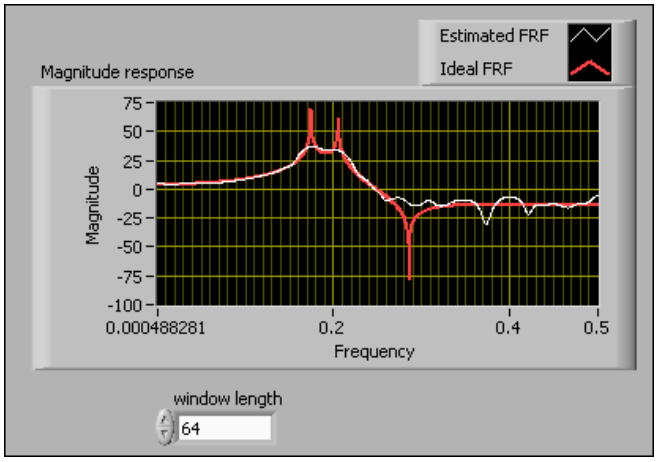

Figure 3-10. Frequency Response with Small Window Length

The frequency response curve is smoother and the variance is smaller when the length of the lag window is small. However, when the length of the lag window is too small, you cannot distinguish between the two close peaks in the frequency response, as shown in Figure 3-10. When the length of the lag window is large, the SI Estimate Frequency Response VI accurately estimates the peaks, as shown in Figure 3-9. The bias is small with a large lag window, but the variance of the estimated frequency response is large with a large lag window.

Setting the length of the lag window to 5–10% of the number of data points when estimating the frequency response often results in a good trade-off between the bias and variance. However, the selection of the length of the lag window is not trivial. The length also depends on the signals, the properties of the system, and the purpose of application. For example, if you want to know the passband of a system, use a smaller lag window. If you want to identify the dynamic properties of a system, such as its natural frequency, use a larger lag window.

Applications of the Frequency Response

The frequency response gives the characteristics of the system in the frequency domain. You can use the frequency response to obtain useful information before applying parametric estimation. For example, you can use the frequency response to determine whether you need to pre-filter the signals or what the model order of the system is. You also can use nonparametric frequency response to verify parametric model estimation results by comparing the frequency response of the parametric model with the nonparametric frequency response.

One example of a real-world application of the frequency response is with the flexible arm, as shown in Figure 3-11. The input of this system is the reaction torque of the structure on the ground. This input is a multi-sine wave with 200 frequency points equally spaced over the frequency band from 0.122 Hz to 24.4 Hz. The output of this system is the acceleration of the flexible arm. The frequency response of this system is not significant outside of the range of interest, which is the frequency band of the input signal, or 0.122 Hz to 24.4 Hz. However, notice that the **magnitude response** has a peak around 42 Hz. The peak around 42 Hz may be the result of noise, or nonlinearity, or another input source. You can use lowpass filtering to remove the 42 Hz peak before applying parametric estimation.

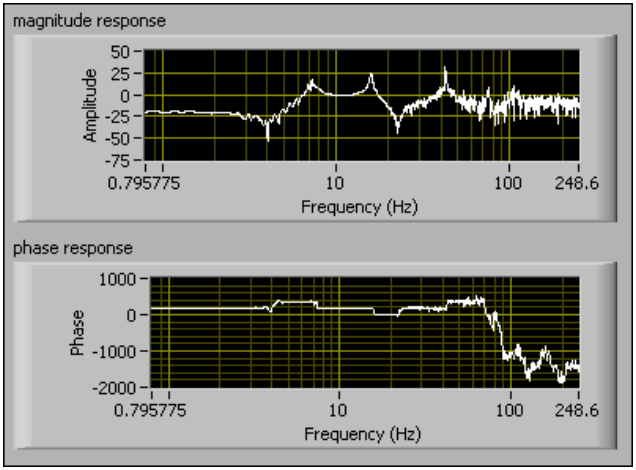

Figure 3-11. Frequency Response of a Flexible Arm

4

Parametric Model Estimation Methods

Parametric models describe systems in terms of differential equations and transfer functions. Compared to nonparametric models, parametric models provide more insight into the physics of the system and are a more compact model structure. However, parametric modeling requires prior knowledge about the system dynamics to determine the model orders, time delays, and so on.

This chapter describes the parametric model estimation methods and the underlying assumptions about each method. This chapter also describes different parametric model representations, reasons for choosing one representation over another, and how to validate the estimated models.

Parametric Model Estimation

The LabVIEW System Identification Toolkit provides two categories of parametric models—polynomial and state-space. The polynomial model includes AR, ARX, ARMAX, output-error, Box-Jenkins, and general-linear models.

General-Linear Polynomial Model

Generally, you can describe a system using the following equation, which is known as the general-linear polynomial model or the general-linear model.

$$y(n) = q^{-k}G(q^{-1}, \theta)u(n) + H(q^{-1}, \theta)e(n)$$

$u(n)$ and $y(n)$ are the input and output of the system respectively.

$e(n)$ is zero-mean white noise, or the disturbance of the system.

$G(q^{-1}, \theta)$ is the transfer function of the deterministic part of the system.

$H(q^{-1}, \theta)$ is the transfer function of the stochastic part of the system.

The deterministic transfer function specifies the relationship between the output and the input signal, while the stochastic transfer function specifies how the output is affected by the disturbance. Some literature refer to the deterministic and stochastic parts as system dynamics and stochastic dynamics, respectively.

The term q^{-1} is the backward shift operator, which is defined by the following equation.

$$q^{-1}x(n) = x(n-1)$$

q^{-k} defines the number of delay samples d between the input and the output.

$G(q^{-1}, \theta)$ and $H(q^{-1}, \theta)$ are rational polynomials as defined by the following equations.

$$G(q^{-1}, \theta) = \frac{B(q, \theta)}{A(q, \theta)F(q, \theta)}$$

$$H(q^{-1}, \theta) = \frac{C(q, \theta)}{A(q, \theta)D(q, \theta)}$$

The vector θ is the set of model parameters. Equations in the following sections of this manual will not display θ to make the equations simpler and easier to read.

The following equations define $A(q)$, $B(q)$, $C(q)$, $D(q)$ and $F(q)$:

$$A(q) = 1 + a_1 q^{-1} + a_2 q^{-2} + \ldots + a_{n_a} q^{-n_a}$$

$$B(q) = b_0 + b_1 q^{-1} + \ldots + b_{n_b - 1} q^{-(n_b - 1)}$$

$$C(q) = 1 + c_1 q^{-1} + c_2 q^{-2} + \ldots + c_{n_c} q^{-n_c}$$

$$D(q) = 1 + d_1 q^{-1} + d_2 q^{-2} + \ldots + d_{n_d} q^{-n_d}$$

$$F(q) = 1 + f_1 q^{-1} + f_2 q^{-2} + \ldots + f_{n_f} q^{-n_f}$$

where n_a, n_b, n_c, n_d, and n_f are the model orders.

Figure 4-1 depicts the signal flow of a general-linear model.

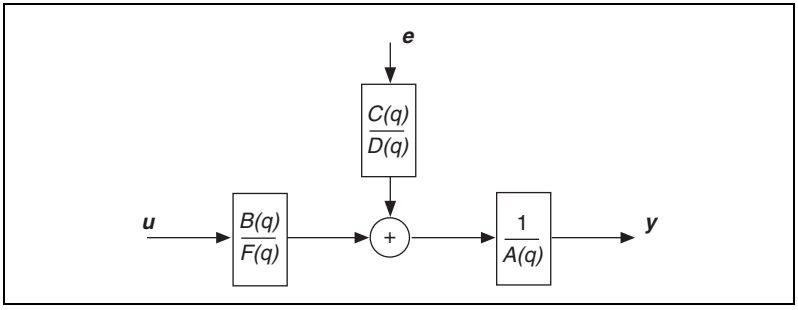

Figure 4-1. Signal Flow of General-Linear Model

A general-linear model provides flexibility for both the system dynamics and stochastic dynamics. However, a nonlinear optimization method computes the estimation of the general-linear model. This method requires intensive computation with no guarantee of global convergence.

Setting one or more of $A(q)$, $C(q)$, $D(q)$, and $F(q)$ equal to 1 can create simpler models such as ARX, ARMAX, output-error, and Box-Jenkins models, which you commonly find in real applications.

ARX Model

When $C(q)$, $D(q)$, and $F(q)$ equal 1, the general-linear polynomial model reduces to an ARX model. The following equation describes an ARX model.

$$A(q)y(n) = q^{-k}B(q)u(n) + e(n) = B(q)u(n-k) + e(n)$$

Note Remember that the backward shift operator makes $q^{-k}u(n) = u(n-k)$.

Figure 4-2 depicts the signal flow of an ARX model.

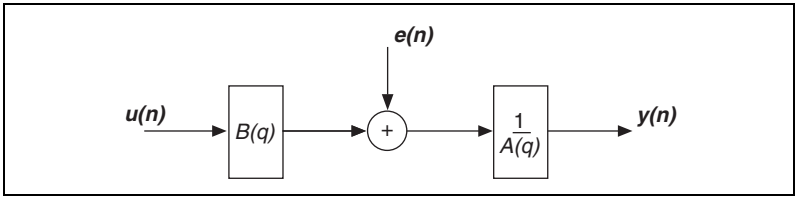

Figure 4-2. Signal Flow of ARX Model

The ARX model is the simplest model incorporating the stimulus signal. The estimation of the ARX model is the most efficient of the polynomial estimation methods because it is the result of solving linear regression

equations in analytic form. Moreover, the solution is unique. In other words, the solution always satisfies the global minimum of the loss function. The ARX model therefore is preferable, especially when the model order is high.

The disadvantage of the ARX model is that disturbances are part of the system dynamics. The transfer function of the deterministic part $G(q^{-1}, \theta)$ of the system and the transfer function of the stochastic part $H(q^{-1}, \theta)$ of the system have the same set of poles. This coupling can be unrealistic. The system dynamics and stochastic dynamics of the system do not share the same set of poles all the time. However, you can reduce this disadvantage if you have a good signal-to-noise ratio.

When the disturbance $e(n)$ of the system is not white noise, the coupling between the deterministic and stochastic dynamics can bias the estimation of the ARX model. Set the model order higher than the actual model order to minimize the equation error, especially when the signal-to-noise ratio is low. However, increasing the model order can change some dynamic characteristics of the model, such as the stability of the model.

The identification method for the ARX model is the least squares (LS) method. The least squares method is a special case of the prediction error method. Refer to the *LabVIEW System Identification Toolkit Algorithm References* (`SIreference.pdf`), available in the `labview\manuals` directory, for more information about the least squares and prediction error methods.

ARMAX Model

When $D(q)$ and $F(q)$ equal 1, the general-linear polynomial model reduces to the ARMAX model. The following equation describes an ARMAX model.

$$A(q)y(n) = q^{-k}B(q)u(n) + C(q)e(n) = B(q)u(n-k) + C(q)e(n)$$

Figure 4-3 depicts the signal flow of the ARMAX model.

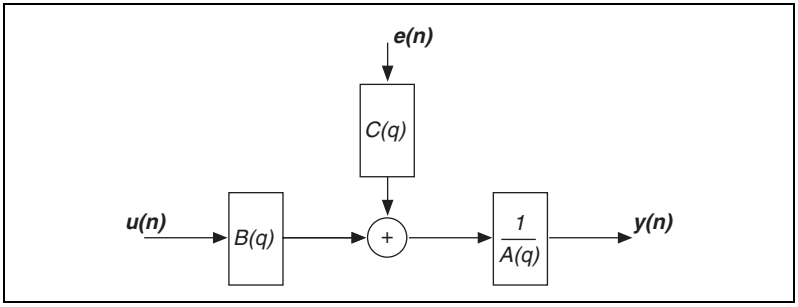

Figure 4-3. Signal Flow of ARMAX Model

Unlike the ARX model, the system structure of an ARMAX model include disturbances dynamics. ARMAX models are useful when you have dominating disturbances that enter early in the process, such as at the input. For example, a wind gust affecting an aircraft is a dominating disturbance early in the process. The ARMAX model has more flexibility in the handling of disturbance modeling than the ARX model.

The identification method of the ARMAX model is the prediction error method. However, you cannot use an analytic form to solve the problem. The SI Estimate ARMAX Model VI uses the Newton-Gauss method to search for the optimal ARMAX model. This searching algorithm is an iterative procedure so it is inefficient and can get stuck at a local minimum, especially when the signal-to-noise ratio is low. Use model validation methods to verify whether the model achieves the required quality and to verify whether the estimation is stuck at a local minimum. If the estimation is stuck at a local minimum, select a new model structure or change the model order.

Output-Error Model

When $A(q)$, $C(q)$, and $D(q)$ equal 1, the general-linear polynomial model reduces to the output-error model. The following equation describes an output-error model.

$$y(n) = \frac{q^{-k}B(q)}{F(q)}u(n) + e(n) = \frac{B(q)}{F(q)}u(n-k) + e(n)$$

Figure 4-4 depicts the signal flow of the output-error model.

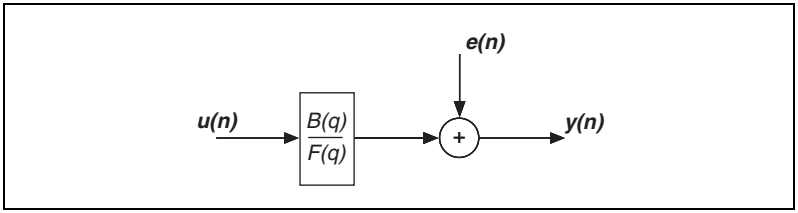

Figure 4-4. Signal Flow of Output-Error Model

The output-error model describes the system dynamics separately. The output-error model does not use any parameters for modeling the disturbance characteristics.

The identification method of the output-error model is the prediction error method, which is the same as that of the ARMAX model. If the input signal $u(n)$ is white noise, all minima are global. There is no local minimum. However, a local minimum can exist if the input signal is not white.

Box-Jenkins Model

When $A(q)$ equals 1, the general-linear polynomial model reduces to the Box-Jenkins model. The following equation describes a Box-Jenkins model.

$$y(n) = \frac{q^{-k}B(q)}{F(q)}u(n) + \frac{C(q)}{D(q)}e(n) = \frac{B(q)}{F(q)}u(n-k) + \frac{C(q)}{D(q)}e(n)$$

Figure 4-5 depicts the signal flow of the Box-Jenkins model.

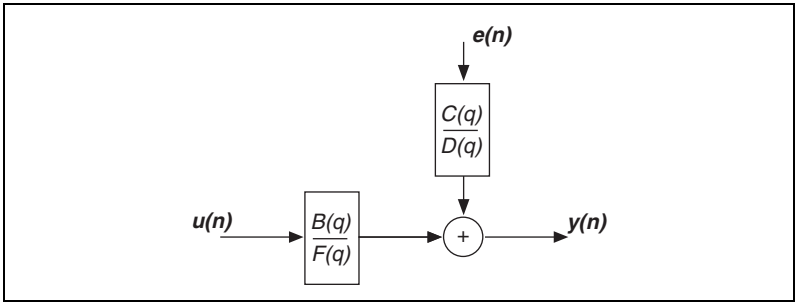

Figure 4-5. Signal Flow of Box-Jenkins Model

The Box-Jenkins model provides a complete model of a system. It models disturbance properties separately from system dynamics. The Box-Jenkins model is useful when you have disturbances that enter late in the process.

For example, measurement noise on the output is a disturbance late in the process.

The identification method of the Box-Jenkins model is the prediction error method, which is the same as that of the ARMAX model.

AR Model

When $C(q)$, $D(q)$, and $F(q)$ equal 1 and $B(q)$ equals 0, the general-linear polynomial model reduces to the AR model. The following equation describes an AR model.

$$A(q)y(n) = e(n)$$

Figure 4-6 depicts the signal flow of the AR model.

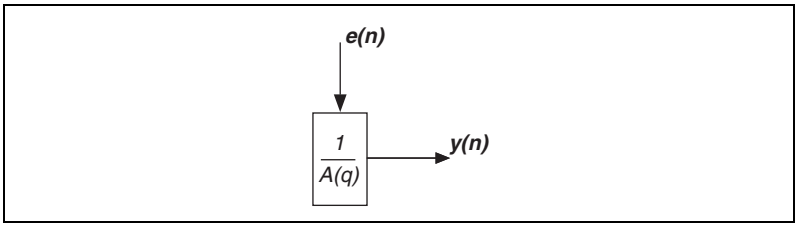

Figure 4-6. Signal Flow of AR Model

Strictly speaking, the AR model is not a model for a system but a model for a signal. A system is a device with an input and an output, but an AR model does not include the dynamics between the input and output.

Time series analyses, such as power spectrum envelope estimation, prewhitening, and linear prediction coding, commonly use the AR model.

If you consider $A(q)$ to be a filter, $A(q)y(n)$ is the filtering of $A(q)$ on the signal $y(n)$. The result of the filtering is white noise $e(n)$, as shown in AR model equation. Hence, the filter $A(q)$ is also known as the whitening or prewhitening filter. From the frequency-domain standpoint, the whitening filter $A(q)$ suppresses the spectrum at frequencies where the magnitude of the spectrum is large. Suppressing the high magnitude frequencies results in a flat spectrum. Refer to Chapter 3, *Nonparametric Model Estimation Methods*, for examples on how to apply the AR model for prewhitening.

As shown in AR model equation, if you know the AR coefficients $A(q)$ and the noise $e(n)$, you can reconstruct the signal $y(n)$. $A(q)$ and $e(n)$ completely characterize a signal. $A(q)$ normally has a small number of coefficients. $e(n)$ has a small dynamic range and requires a smaller number of bits for

encoding. Therefore, you can use the AR model for compression purposes in a process known as linear prediction coding (LPC). Speech and vibration signal processing, such as compression, pattern recognition, and so on, commonly use LPC. $A(q)$ and $e(n)$ also can be used for estimating the power spectrum of the signal $y(n)$.

State-Space Model

In addition to polynomial models, the System Identification Toolkit also can estimate state-space models. The following equations describe a state-space model.

$$x(n+1) = \boldsymbol{A}x(n) + \boldsymbol{B}u(n) + \boldsymbol{K}e(n)$$

$$y(n) = \boldsymbol{C}x(n) + \boldsymbol{D}u(n) + e(n)$$

$x(n)$ is the state vector. $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C}$, $\boldsymbol{D}$, and $\boldsymbol{K}$ are the system matrices. The dimension of the state vector $x(n)$ is the only setting you need to provide for the state-space model. The state-space model describes a system using difference equations with an auxiliary state vector $x(n)$. The state-space transfer matrices often reflect physical characteristics of a system. Hence, state-space models often are preferable to polynomial models, especially in modern control applications.

The SI Estimate State-Space Model VI uses the N4SID algorithm to estimate the state-space model. Refer to the *System Identification Toolkit Algorithm References* (`SIreference.pdf`), available in the `labview\manuals` directory, for more information about this algorithm.

Polynomial Models versus State-Space Models

Selecting the correct model type and model order is crucial for successful parametric model estimation. In general, the state-space model provides a more complete representation of the system, especially for MIMO systems, than polynomial models because state-space models are similar to first principle models. The identification procedure does not involve nonlinear optimization so the estimation reaches a solution regardless of the initial guess. Moreover, the parameter settings for the state-space model are simpler. You need to select only the order, or the number of states, of the model. The order can come from prior knowledge of the system. You also can determine the order by analyzing the singular values of the information matrix. Refer to Chapter 6, *System Identification Case Study*, for an example of how to estimate the order of the state-space model of a system.

When the model order is high, use an ARX model because the algorithm involved in ARX model estimation is fast and efficient when the number of data points is very large. The state-space model estimation with a large number of data points is slow and requires a large amount of memory. If you must use a state-space model, for example in modern control methods, reduce the sampling rate of the signal in case the sampling rate is unnecessarily high.

The other polynomial models, including the ARMAX, output-error, Box-Jenkins, and general-linear models, involve iterative, nonlinear optimization in the identification procedure. They require excessive computation time, and the minimization can get stuck at a false local minimum, especially when the order is high and the signal-to-noise ratio is low. However, you can use these models when the stochastic dynamic is important because they provide more flexibility for the stochastic dynamics.

Determining Parameters for the Prediction Error Method

The identification method for the ARMAX, output-error, and Box-Jenkins models is the prediction error method. Determining the delay and model order for the prediction error method is typically a trial-and-error process. The following is a useful set of steps that can lead to a suitable model. This is not the only methodology you can use, nor is this a comprehensive procedure.

1. Obtain useful information about the model order by observing the number of resonance peaks in the nonparametric frequency response function. Normally, the number of peaks in the magnitude response equals half the order of $A(q)F(q)$.
2. Obtain a reasonable estimate of the delay by observing the impulse response or by testing reasonable values in a medium-sized ARX model. Choose the delay that provides the best model fit based on prediction errors or another criterion.
3. Test various ARX model orders with this delay choosing those orders that provide the best fit.
4. Because the ARX model describes both the system dynamics and noise properties using the same set of poles, the resulting model can be unnecessarily high in order. By plotting the poles and zeros with the uncertainty intervals and looking for potential cancellations of pole-zero pairs, you can reduce the model order. The resulting orders of the poles and zeros are a good starting point for ARMAX,

output-error, and Box-Jenkins models. Use these orders as the *B* and *F* model parameters and the first- or second-order models for the noise characteristics.

5. If you cannot obtain a suitable model at this point, determine if additional signals influence the output. You can incorporate measurements of these signals as extra input signals.

If you still cannot obtain a suitable model, additional physical insight into the problem might be necessary. Compensating for nonlinear sensors or actuators and handling of important physical nonlinearities are often necessary in addition to using a ready-made model.

From the prediction error standpoint, the higher the order of the model is, the better the model fits the data because the model has more degrees of freedom. However, you need more computation time and memory for higher orders. The parsimony principle advocates choosing the model with the smallest degree of freedom, or number of parameters, if all the models fit the data well and pass the verification test. The criteria to assess the model order therefore not only must rely on the prediction error but also must incorporate a penalty when the order increases. Akaike's Information Criterion (AIC), Akaike's Final Prediction Error Criterion (FPE), and the Minimum Data Length Criterion (MDL) are criteria you can use to estimate the model order. The SI Estimate Orders of System Model VI implements the AIC, FPE, and MDL methods to search for the optimal model order in the range of interest. You also can plot the prediction error as a function of the model dimension and then visually find the knee in the curve or apply an F-test to obtain an appropriate estimation of the model order.

Akaike's Information Criterion

The Akaike's Information Criterion (AIC) is a weighted estimation error based on the unexplained variation of the actual data with a penalty term when exceeding the optimal number of parameters to represent the system.

An optimal model is the one that minimizes the following equation.

$$AIC = N\log(V_n(\hat{\theta})) + 2p$$

N is the number of data points, V_n is an index related to the prediction error, or the residual sum of squares, θ is the vector of parameters that the model uses, and p defines the number of parameters in the model.

Akaike's Final Prediction Error Criterion

Akaike's Final Prediction Error Criterion (FPE) estimates the prediction error when you use the model to predict new outputs. The following equation defines the FPE criterion.

$$FPE = \left(\frac{1 + \frac{p}{N}}{1 - \frac{p}{N}} \right) MSE$$

MSE is the mean square error.

You want to choose model that minimizes the FPE, which represents a balance between number of parameters and explained variation.

Minimum Data Length Criterion

The Minimum Data Length Criterion (MDL) is based on V_n plus a penalty for the number of terms used. The following equation defines the MDL criterion.

$$MDL = V_n + \frac{p \log(N)}{N}$$

Converting Models

According to linear system theory, you can represent a linear, discrete-time system with different models. You can convert one model representation to another using the System Identification Toolkit VIs. Use the Model Conversion VIs to convert an AR, ARX, ARMAX, output-error, Box-Jenkins or general-linear model into a state-space, transfer function, or zero-pole-gain model and to convert a state-space model into a transfer function or zero-pole-gain model.

Each model representation has benefits and drawbacks for characterizing a dynamic system. Refer to the *Polynomial Models versus State-Space Models* section for information about choosing an appropriate model representation to use.

You also can use the Model Conversion VIs to convert models created in the System Identification Toolkit into transfer function, zero-pole-gain, or state-space models that you can use with the LabVIEW Control Design Toolkit. This model conversion process enables you to identify a model for

an unknown system with the System Identification Toolkit and then design a controller for that same system using the Control Design Toolkit.

Refer to the *Model-Based Control Design Process* section of Chapter 1, *Introduction to System Identification*, for more information about the integration of the System Identification Toolkit and the Control Design Toolkit in the model-based control design process. Refer to `ni.com` for more information about the Control Design Toolkit.

Validating Models

Model estimation only determines the best model of the system within the chosen model structure. Model estimation does not determine whether the model provides the most accurate description of the system. Once you obtain a model, you then must validate the model to determine how well the behavior of the model corresponds to both the observed data, any prior knowledge of the system, and the purpose for which you will use the model. If the model is inadequate, you must revise the system identification process or consider using another approach.

Validating the model provides a measure of the quality of the model obtained through the system identification process. Model validation determines whether the model is flexible enough to describe the system and whether the model is too complex.

The best way to validate a model is to experiment with the model under real-world conditions. If the model works as well as expected, then the model estimation is successful. However, experimenting with the model under real-world conditions can be dangerous. For example, introducing arbitrary perturbations to the input of a chemical plant can lead to a harmful explosion.

Before you incorporate the model into real applications you can validate the model by using plots and common sense or by using statistical tests on the prediction error.

The System Identification Toolkit provides three of the most common validation methods: model simulation, model prediction, and model residual analysis. In addition to these methods, you also can perform Bode, Nyquist, or pole-zero analyses to investigate the results of the model estimation.

Validation Methods

Once you build a model, you can use at least three different methods to validate the model and evaluate its flexibility. You can use model simulation to understand the underlying dynamic relationship between the model inputs and outputs. You can use model prediction to test the ability of the model to predict the response of the system using past input and output data. You also can use model residual analysis to test the whiteness of the prediction error and the independency between the prediction error and the input signal using statistical techniques. The methods you select to validate the model depend on the purpose for which you created the model. You can use one or all of these methods to validate the model.

Simulation

The CD Model Simulation VI determines the outputs of a system for given inputs. The best way to determine the outputs is to verify the behavior of a system in a real-world environment by intentionally injecting perturbations into the input data. For example, you can study the characteristics of an electric motor by varying the input voltage and observing how the motor responds. However, in many cases you cannot experiment with a system because of safety concerns.

Once you build a model for the system based on the input and output data, you do not have to evaluate the behavior of the system in a real-world environment. You can use the model to simulate the response of the system by using the model equations and then evaluate the behavior of the system. You also can use a simulation to validate the model by comparing the simulated response with the measured response.

Some systems that you want to identify may be unstable. For an unstable system, simulation is not suitable for validating the system. Use model prediction to validate the system instead.

Prediction

The SI Model Prediction VI determines the response of a system at time t based on the output information available at time $t - k$ and all the inputs applied from time $t - k$ to time t. k represents the size of prediction window. Therefore, model prediction is helpful in determining how useful a model is in estimating future responses of the system, given all information at time t and an expected input profile in the future. There are some control techniques that take advantage of model prediction to improve control

performance. For example, model predictive control uses of the prediction properties of a model to determine if a particular limitation or constraint is active in the future. This method allows the controller to take preventive action before such constraints become active.

If you have the measured input and output of a system, you also can validate the model of the system by comparing the predicted output and the measured output. If the prediction error is small, the model is acceptable.

When a system is unstable and you cannot use simulation to validate the system, use model prediction to validate the model.

Residual Analysis

Residual analysis is the third validation method that the System Identification Toolkit provides. The response that an estimated model predicts and the actual response from the system are different. This difference is called the prediction error or residual. The following equation defines the residual $e(n)$.

$$e(n) = y(n) - y'(n)$$

$y(n)$ is measured output data and $y'(n)$ is output data from the one-step-ahead prediction. If the model is capable of describing the true system, the residual is zero-mean white noise and independent of the input signal. You can use autocorrelation analysis to test whether the residual is zero-mean white noise. You can use cross correlation analysis to test whether the residual is independent of the input signal. The SI Model Residual Analysis VI calculates both the autocorrelation and the cross correlation values.

Autocorrelation

The following equation defines the autocorrelation of the residuals.

$$R_e{}^N(\tau) = \frac{1}{N}\sum_{n=1}^{N} e(n)e(n+\tau)$$

Ideally, the residual is white noise and therefore the autocorrelation $R_e{}^N(\tau)$ is zero when τ is non-zero. A large autocorrelation when τ is non-zero indicates that the residual is not zero-mean white noise and normally implies that the model structure is not relevant to the system or that you need to increase the model order.

In real applications, the autocorrelation $R_e^N(\tau)$ cannot be zero when τ is non-zero because of the limited length of data points. However, the **confidence level of autocorrelation** output of the SI Model Residual Analysis VI assesses whether the autocorrelation value is sufficiently small to be ignored. If the value of the autocorrelation is smaller than the value of **confidence level of autocorrelation**, the value is insignificant and you can consider it to be equal to zero.

Cross Correlation

The following equation defines the cross correlation between residuals and past inputs.

$$R_{eu}^{N}(\tau) = \frac{1}{N}\sum_{n=1}^{N} e(n)u(n+\tau)$$

If the residual is independent of the input, the cross correlation is zero for all τ. If the residual correlates with the input, the model has not captured all deterministic variations from the data. Therefore you need to revise the model variation.

The **confidence level of cross correlation** output of the SI Model Residual Analysis VI assesses whether the value of the cross correlation is sufficiently small. If the value of the cross correlation is smaller than the value of **confidence level of cross correlation**, the value is insignificant and you can consider it to be equal to zero.

Model Order Reduction

You can plot the pole-zero diagram of the system dynamic of the model by using the SI Pole-Zero Plot VI and then observing whether any pole-zero pairs have overlapping confidence intervals. If overlapping confidence intervals exist, then pole-zero cancellations exist, and the model possibly is too complex.

You also can reduce the model order and then use the F-test criterion to assess whether the reduction in model order leads to a significant increase in the prediction error. If the reduction in model order does lead to a significant increase in the prediction error, then you can reduce the model order.

5

Recursive Model Estimation Methods

The model estimation methods in Chapter 3, *Nonparametric Model Estimation Methods*, and Chapter 4, *Parametric Model Estimation Methods*, use nonrecursive methods to estimate a model of the system. Nonrecursive model estimation identifies a model for a system based on input-output data gathered at a time prior to the current time. In many real-world applications, such as adaptive control and adaptive prediction, having a model of the system update while the system is running is necessary or helpful. In this type of application, you obtain the mathematical model of the system in real time.

Recursive model estimation is a common system identification technique that enables you to develop a model that adjusts based on real-time data coming from the system. Recursive model estimation processes the measured input-output data recursively as the data becomes available. This chapter discusses recursive model estimation techniques and various adaptive algorithms associated with each method.

Defining Recursive Model Estimation

Figure 5-1 represents a general recursive system identification application. A system identification application consists of an unknown system that has an input signal, or stimulus signal $u(n)$ and an output signal, or response signal $y(n)$.

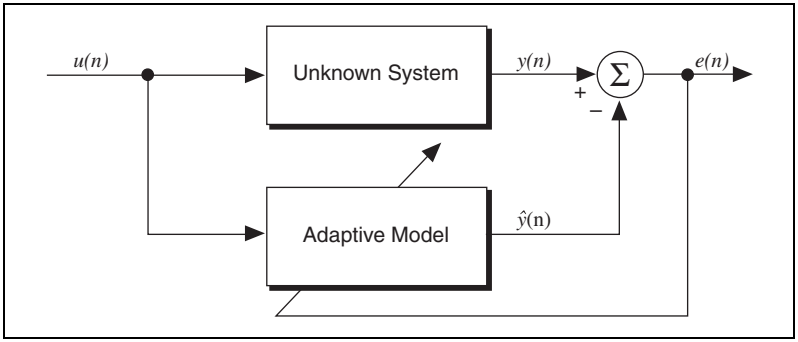

Figure 5-1. Recursive System Identification Diagram

The stimulus signal $u(n)$ is the input to both the unknown system and the adaptive model. The response of the system $y(n)$ and the predicted response of the adaptive model $\hat{y}(n)$ are combined to determine the error of the system. The error of the system is defined by the following equation.

$$e(n) = y(n) - \hat{y}(n)$$

The adaptive model generates the predicted response $\hat{y}(n+1)$ based on $u(n+1)$ after adjusting the parametric vector $\vec{w}(n)$ based on the error $e(n)$.

Figure 5-1 shows how the error information $e(n)$ is sent back to the adaptive model, which adjusts the parametric vector $\vec{w}(n)$ to account for the error. You iterate on this process until you minimize the magnitude of the least mean square error $e(n)$.

Before you apply the recursive model estimation, you must first select the parametric model structure that determines the parametric vector $\vec{w}(n)$. Then, you must select the method that automatically adjusts the parametric vector such that the error $e(n)$ goes to the minimum.

The System Identification Toolkit provides Recursive Model Estimation VIs that support the following model structures:

- ARX
- ARMAX
- Output Error
- Box-Jenkins
- General Linear

Refer to the *Parametric Model Estimation* section of Chapter 4, *Parametric Model Estimation Methods*, for information about each of these

models. Refer to the *LabVIEW Help*, available by selecting **Help»VI, Function, & How-To Help**, for information about the Recursive Model Estimation VIs and mathematical definitions of each of these models.

You can compute each model recursively using the following four types of adaptive algorithms:

- Least mean square (LMS)
- Normalized least mean square (NLMS)
- Recursive least squares (RLS)
- Kalman filter (KF)

The following sections provide more information about each of these adaptive algorithms.

Adaptive Algorithms

Adaptive algorithms are fundamental in recursive system identification. The adaptive method you use affects the performance of recursive system identification application.

The goal of an adaptive algorithm is to minimize the cost function. The following equation defines the cost function $J(n)$.

$$J(n) = E[e^2(n)]$$

Again, $e(n)$ represents the difference between the predicted response $\hat{y}(n)$ and the response $y(n)$ of the unknown system, as shown in Figure 5-1.

The goal of all recursive algorithms is to adjust the parametric vector $\vec{w}(n)$ until you minimize the cost function $J(n)$. When the cost function $J(n)$ is sufficiently small, the parametric vector $\vec{w}(n)$ is considered optimal for the estimation of the actual system.

Least Mean Square

The following equation uses the least mean squares (LMS) method to define the cost function.

$$J(n) = E[e^2(n)]$$

The parametric vector $\vec{w}(n)$ updates according to the following equation.

$$\vec{w}(n+1) = \vec{w}(n) + \mu e(n)\vec{\varphi}(n)$$

n is the number of iterations, μ is step-size, which is a positive constant, and $\vec{\varphi}(n)$ is the data vector from the past input data $u(n)$ and output data $y(n)$. $\vec{\varphi}(n)$ is defined by the following equation.

$$\varphi(n) = \left[-y(t-1)\ldots-y(t-n)\ u(t-1)\ldots u(t-m)\right]^T$$

The following procedure describes how to implement the LMS algorithm.

1. Initialize the step-size μ.
2. Initialize the parameter vector $\vec{w}(n)$ using a small positive number ε.

$$\vec{w}(0) = [\varepsilon,\ \varepsilon,\ \ldots,\ \varepsilon]^T$$

3. Initialize the data vector $\vec{\varphi}(n) = \left[u(n)\ y(n)\right]$.

$$\vec{\varphi}(0) = [0,\ 0,\ \ldots,\ 0]^T$$

4. For $n = 1$, update the data vector $\vec{\varphi}(n)$ based on $\vec{\varphi}(n-1)$ and the current input data $u(n)$ and output data $y(n)$.
5. Compute the predicted response $\hat{y}(n)$ by using the following equation.

$$\hat{y}(n) = \vec{\varphi}^T(n)\vec{w}(n)$$

6. Compute the error $e(n)$ by solving the following equation.

$$e(n) = y(n) - \hat{y}(n)$$

7. Update the parameter vector $\vec{w}(n)$.

$$\vec{w}(n+1) = \vec{w}(n) + \mu e(n)\vec{\varphi}(n)$$

8. Stop if the error is small enough, else set $n = n + 1$ and repeat steps 4–8.

The LMS algorithm is one of the most widely used and understood adaptive algorithms. Selecting the step-size μ is important with the LMS algorithm.

The selection of the step-size μ directly affects the rate of convergence and the stability of the algorithm. The convergence rate of the LMS algorithm

is usually proportional to the step-size μ. The larger the step-size μ, the faster the convergence rate. However, a large step-size μ can cause the LMS algorithm to become unstable. The following equation describes the range of the step-size μ.

$$0 < \mu < \mu_{max}$$

μ_{max} is the maximum step-size that maintains stability in the LMS algorithm. μ_{max} is related to the statistical property of the stimulus signal. There is not a uniformly optimized step-size μ that achieves a fast convergence speed while maintaining the stability in the system regardless of the statistical property of the stimulus signal. For better performance, use a self-adjusted step-size μ and the normalized least mean squares (NLMS) algorithm.

Normalized Least Mean Square

The following equation defines a popular self-adjustable step-size $\mu(n)$ that you use in the normalized least mean squares algorithm.

$$\mu(n) = \frac{\mu}{\varepsilon + \left\|\vec{\varphi}(n)\right\|^2}$$

Again, $\vec{\varphi}(n)$ represents the data vector. ε is a very small positive number that prevents the denominator from equaling zero when $\left\|\vec{\varphi}(n)\right\|^2$ is approaches to zero.

The step-size $\mu(n)$ is time-varying because the step-size changes with the time index n.

Substituting $\mu(n)$ into the parametric vector $\vec{w}(n)$ equation yields the following equation.

$$\vec{w}(n+1) = \vec{w}(n) + \mu(n)e(n)\vec{\varphi}(n)$$

Compared to the LMS algorithm, the NLMS algorithm is always stable if the step-size $\mu(n)$ is between zero and two, regardless of the statistical property of the stimulus signal $u(n)$.

The procedure of the NLMS algorithm is the same as the LMS algorithm except for the estimation of the time varying step-size $\mu(n)$.

Recursive Least Squares

The following equation uses the recursive least squares (RLS) algorithm and Kalman filter algorithm to modify the cost function $J(n)$ defined in the *Adaptive Algorithms* section. Refer to the *Kalman Filter* section for information about the Kalman filter algorithm.

$$J(n) = E[e^2(n)] \cong \frac{1}{N}\sum_{i=0}^{N-1} e^2(n-i)$$

Compare this modified cost function, which uses the previous N error terms, to the cost function, $J(n) = E[e^2(n)]$, which uses only the current error information $e(n)$. The modified cost function $J(n)$ is more robust. The corresponding convergence rate in the RLS algorithm is faster, but the implementation is more complex than LMS-based algorithms.

The following procedure describes how to implement the RLS algorithm.

1. Initialize the parameter vector $\vec{w}(n)$ using a small positive number ε.

$$\vec{w}(0) = [\varepsilon,\ \varepsilon,\ \ldots,\ \varepsilon]^T$$

2. Initialize the data vector $\vec{\varphi}(n)$.

$$\vec{\varphi}(0) = [0,\ 0,\ \ldots,\ 0]^T$$

3. Initialize the $n \times n$ matrix $\boldsymbol{P}(0)$.

$$\boldsymbol{P}(0) = \begin{bmatrix} \varepsilon & 0 & 0 & 0 \\ 0 & \varepsilon & 0 & 0 \\ 0 & 0 & \ldots & 0 \\ 0 & 0 & 0 & \varepsilon \end{bmatrix}$$

4. For $n = 1$, update the data vector $\vec{\varphi}(n)$ based on $\vec{\varphi}(n-1)$ and the current input data $u(n)$ and output data $y(n)$.
5. Compute the predicted response $\hat{y}(n)$ by using the following equation.

$$\hat{y}(n) = \vec{\varphi}^T(n) \cdot \vec{w}(n)$$

6. Compute the error $e(n)$ by solving the following equation.

$$e(n) = y(n) - \hat{y}(n)$$

7. Update the gain vector $\vec{K}(n)$ defined by the following equation.

$$\vec{K}(n) = \frac{\boldsymbol{P}(n) \cdot \vec{\varphi}(n)}{\lambda + \vec{\varphi}^T(n) \cdot \boldsymbol{P}(n) \cdot \vec{\varphi}(n)}$$

The properties of a system might vary with time, so you need to ensure that the algorithm tracks the variations. You can use the forgetting factor λ, which is an adjustable parameter, to track these variations. The smaller the forgetting factor λ, the less previous information this algorithm uses. When you use small forgetting factors, the adaptive filter is able to track time-varying systems that vary rapidly. The range of the forgetting factor λ is between zero and one, typically $0.98 < \lambda < 1$.

$\boldsymbol{P}(n)$ is a $n \times n$ matrix whose initial value is defined by $\boldsymbol{P}(0)$ in step 3.

8. Update the parameter vector $\vec{w}(n+1)$.

$$\vec{w}(n+1) = \vec{w}(n) + e(n) \cdot \vec{K}(n)$$

9. Update the $\boldsymbol{P}(n)$ matrix.

$$\boldsymbol{P}(n+1) = \boldsymbol{P}(n) - \vec{K}(n) \cdot \vec{\varphi}^T(n) \cdot \boldsymbol{P}(n)$$

10. Stop if the error is small enough, else set $n = n + 1$ and repeat steps 4–10.

Kalman Filter

The Kalman filter is a linear optimum filter that minimizes the mean of the squared error recursively. The convergence rate of the Kalman filter is usually faster, but the implementation is more complex than LMS-based algorithms.

Recall that the equation $J(n) = E[e^2(n)]$ defines the cost function. The following procedure lists the steps of the Kalman filter algorithm.

1. Initialize the parameter vector $\vec{w}(n)$ using a small positive number ε.

$$\vec{w}(0) = [\varepsilon, \varepsilon, \ldots, \varepsilon]^T$$

2. Initialize the data vector $\vec{\varphi}(n)$.

$$\vec{\varphi}(0) = [0, 0, \ldots, 0]^T$$

3. Initialize the $n \times n$ matrix $\boldsymbol{P}(0)$.

$$\boldsymbol{P}(0) = \begin{bmatrix} \varepsilon & 0 & 0 & 0 \\ 0 & \varepsilon & 0 & 0 \\ 0 & 0 & \ldots & 0 \\ 0 & 0 & 0 & \varepsilon \end{bmatrix}$$

4. For $n = 1$, update the data vector $\vec{\varphi}(n)$ based on $\vec{\varphi}(n-1)$ and the current input data $u(n)$ and output data $y(n)$.
5. Compute the predicted response $\hat{y}(n)$ by solving the following equation.

$$\hat{y}(n) = \vec{\varphi}^T(n) \cdot \vec{w}(n)$$

6. Compute the error $e(n)$ by solving the following equation.

$$e(n) = y(n) - \hat{y}(n)$$

7. Update the Kalman gain vector $\vec{K}(n)$ defined by the following equation.

$$\vec{K}(n) = \frac{\boldsymbol{P}(n) \cdot \vec{\varphi}(n)}{\boldsymbol{Q}_M + \vec{\varphi}^T(n) \cdot \boldsymbol{P}(n) \cdot \vec{\varphi}(n)}$$

   $\boldsymbol{Q}_M$ is the measurement noise and $\boldsymbol{P}(n)$ is a $n \times n$ matrix whose initial value is defined by $\boldsymbol{P}(0)$ in step 3.
8. Update the parameter vector $\vec{w}(n)$.

$$\vec{w}(n+1) = \vec{w}(n) + e(n) \cdot \vec{K}(n)$$

9. Update the $\boldsymbol{P}(n)$ matrix.

$$\boldsymbol{P}(n+1) = \boldsymbol{P}(n) - \vec{K}(n) \cdot \vec{\varphi}^T(n) \cdot \boldsymbol{P}(n) + \boldsymbol{Q}_P$$

   $\boldsymbol{Q}_P$ is the correlation matrix of the process noise.
10. Stop if the error is small enough, else set $n = n + 1$ and repeat steps 4–10.

The Recursive Model Estimation VIs have a **recursive method** parameter that enables you to specify which recursive estimation method to use.

6

System Identification Case Study

This chapter contains a case study that guides you through the system identification process. The case study uses sample data that the LabVIEW System Identification Toolkit provides in the SI Data Samples VI. The SI Data Samples VI includes data sets for a DC motor, a flexible robot[1] arm, and a ball and beam apparatus. The case study in this chapter uses the flexible arm data to demonstrate the system identification process and compare different estimation methods.

The flexible arm is a nonlinear dynamic system.The System Identification Toolkit enables you to model systems linearly. So this case study demonstrates how you obtain a linear representation of a nonlinear system.

The VIs for this case study are located in `labview\examples\system identification\SICaseStudy1.llb`.

Refer to the *LabVIEW Help*, available by selecting **Help»VI, Function, & How-To Help**, for information about specific System Identification VIs in this case study.

Data Preprocessing

The next step in the system identification process after gathering data is to preprocess the data. The input to this system is the reaction torque of the structure on the ground. This input is a swept sine wave with 200 frequency points equally spaced over the frequency band from 0.122 Hz to 24.4 Hz.

The output of this system is the acceleration of the flexible arm. The acceleration contains information about the flexible resonances and anti-resonances.

[1] The flexible robotic arm data was adopted from a case study in the *MATRIXx Interactive System Identification Module, Part 2* manual. Hendrik Van Brussel and Jan Swevers of the laboratory of Production Manufacturing and Automation of the Katholieke Universiteit Leuven provided this data, which they obtained in the framework of the Belgian Programme on Interuniversity Attracion Poles.

The data set contains 4096 samples at a sampling rate of 500 Hz or sampling time of 0.002 seconds. Thus the total time of the response is 8.192 seconds.

The following sections show you how to preprocess the raw data by examining the time and frequency responses of the system. Based on those analyses, you can filter and downsample the data set to reduce the amount of data in the raw data set for simpler identification.

Examining the Time Response Data

Using the data in the SI Data Sample VI for the flexible robotic arm, you can view the input and output data, as shown in Figure 6-1.

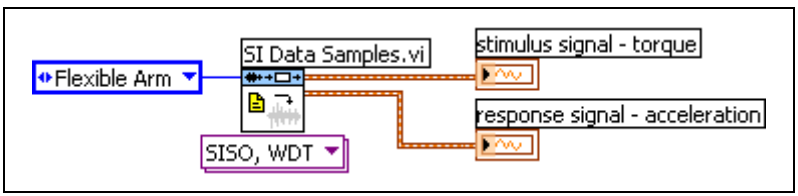

Figure 6-1. Flexible Arm Data Set VI

Note The names of the figures in this chapter reflect the names of the example VIs located in the `labview\examples\system identification\SICaseStudy1.llb`.

The **stimulus signal – torque** output corresponds to the input data, or the torque, and the **response signal – acceleration** output corresponds to the output data, or the acceleration.

Figure 6-2 shows the input and output data on graphs during the length of the response. By looking at the graphs, you can inspect the data for outliers, clipped saturation, or quantization effects that you can remove because they are not representative of the system behavior.

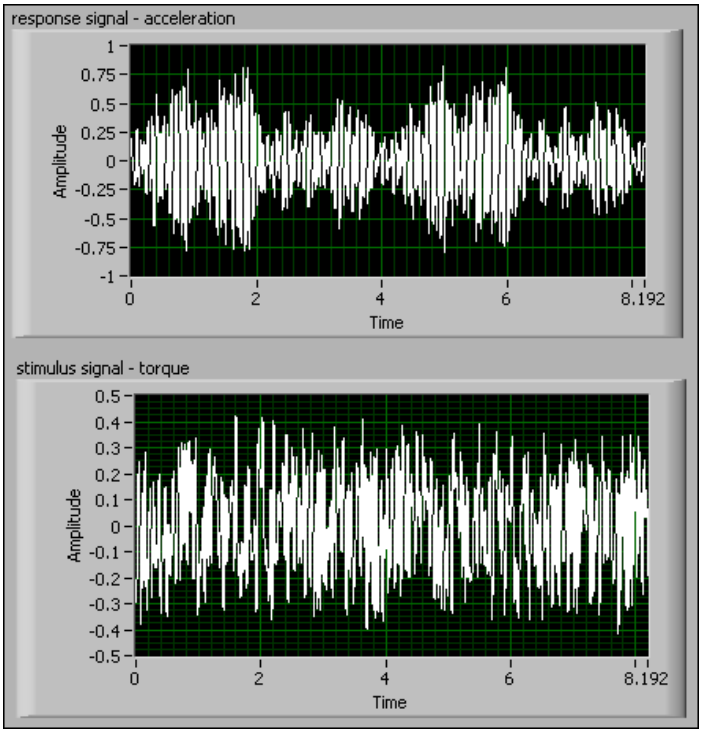

Figure 6-2. Flexible Arm Data Set Plotted in the Time Domain

Figure 6-2 shows no obvious nominal, trend, or outlier values in the input or output time waveforms.

Examining the Frequency Response Data

In addition to examining the time response data, you also want to examine the frequency response data. You can use the SI Estimate Frequency Response VI to view the frequency response of the measured output signal, as shown in Figure 6-3.

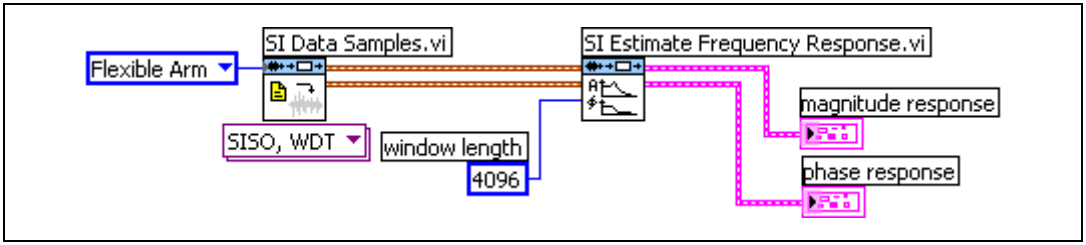

Figure 6-3. Non-Parametric FRF VI

The input data is periodic over 4096 samples, which is the signal length. Notice that in Figure 6-3 the **window length**, 4096, is the same as the signal length so as to obtain a smaller bias in the frequency response estimation.

Figure 6-4 shows the magnitude and phase responses of the measured output signal. The **magnitude response** graph shows three resonances and two anti-resonances in the frequency domain. Resonances are vibrations of large amplitude in a system caused by exciting the system at its natural frequency.

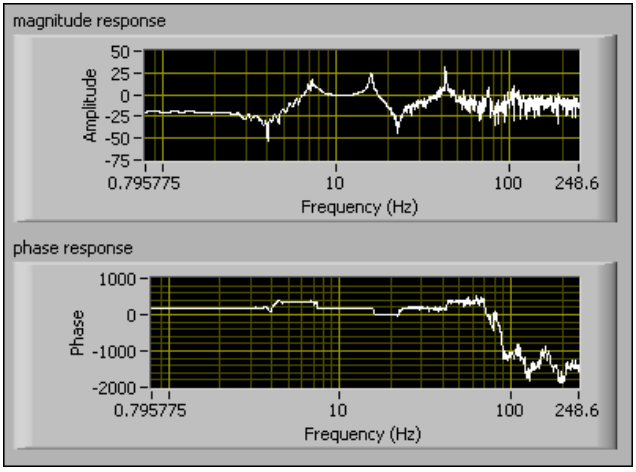

Figure 6-4. Frequency Response of the Flexible Arm Data Set

Notice the resonance at approximately 42 Hz. You can deduce that this resonance is caused by noise or nonlinear system behavior because the 42 Hz falls outside the frequency range of the input data, 0.122–24.4 Hz. At 42 Hz, there is no input energy, thus implying that the response at 42 Hz is not a result of the input.

By examining the frequency response data, you see that filtering is necessary to remove this resonance peak at 42 Hz. The following section describes how to use the System Identification Toolkit to apply a filter to the flexible arm data.

Applying a Filter to the Raw Data

To eliminate the resonance peak at 42 Hz, you can apply a filter to the raw data. By first applying a lowpass filter with a cutoff frequency of 25 Hz, you eliminate the high-frequency noise from the raw data set. Figure 6-5 shows how to use SI Lowpass Filter to apply a lowpass filter to the raw data set.

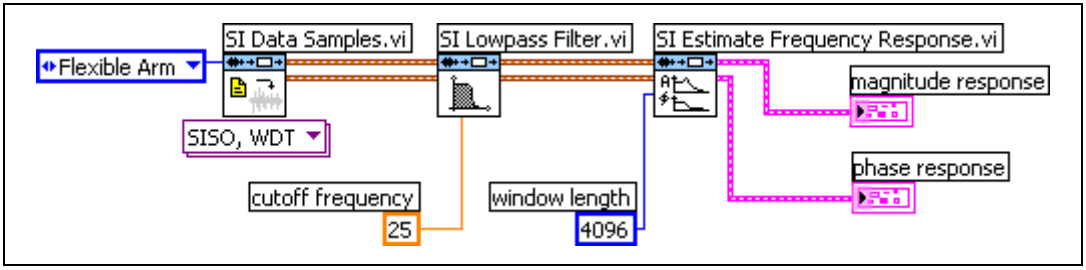

Figure 6-5. Non-Parametric FRF with Prefiltering VI

You can see the effects of the lowpass filter by comparing the frequency response of the filtered data set in Figure 6-6 to the frequency response of the non-filtered data set in Figure 6-4. By using a lowpass filter, you can see that the resonance at approximately 42 Hz is no longer part of the data set you will use to estimate the model.

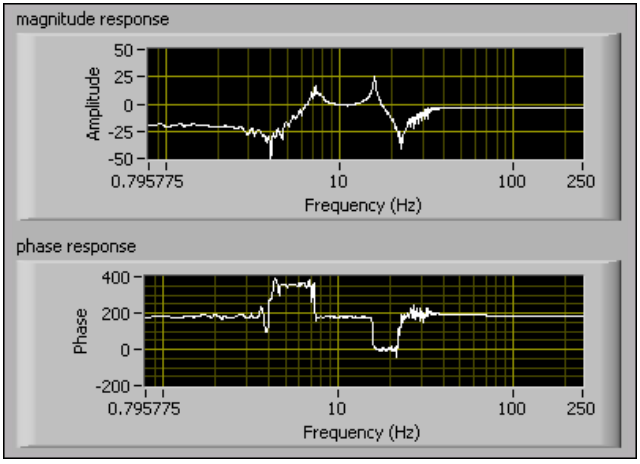

Figure 6-6. Frequency Response of the Filtered Data Set

Downsampling the Raw Data

Sampling theory, in conjunction with the Nyquist criterion, enables you to reduce the sampling rate from 500 Hz to 50 Hz. Applying a filter and downsampling the data set reduces the number of samples in and the computational complexity of the data set. The goal is to use as few samples as possible to evaluate the behavior of the system.

Sampling theory enables you to downsample, or decimate, the data set. Downsampling reduces the sampling rate, 500 Hz, by a factor of 10. Thus downsampling enables you to acquire the data at a sampling rate of 50 Hz. The Nyquist criterion states that you need to sample the signal at a minimum of twice the highest frequency in the system.

Recall that the input data is equally spaced over the frequency band 0.122 – 24.4 Hz. Therefore, according to the Nyquist criterion, you need to sample at a minimum of 50 Hz to avoid any antialiasing. The benefit of sampling at 50 Hz is that you still acquire all the data in the frequency band, yet you eliminate the resonance peak at 42 Hz.

Therefore, in Figure 6-7, the SI Lowpass Filter VI sets the **cutoff frequency** to 25. In addition to applying a lowpass filter to the data, you must downsample the reduced data set. The SI Down Sample VI in Figure 6-7 uses a **decimation factor** of 10.

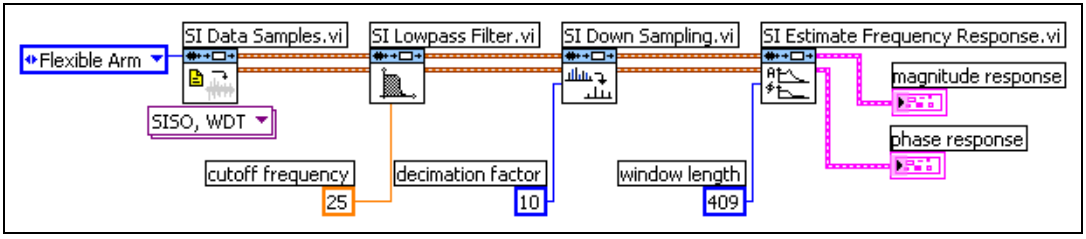

Figure 6-7. Non-Parametric FRF with Prefiltering and Down Sampling VI

The SI Lowpass Filter VI applies a lowpass filter before downsampling the data set to avoid aliasing at the 42 Hz resonance. Together, the lowpass filter and downsampling removes the high frequency disturbance and makes the process faster and more efficient.

Notice that the **window length** parameter of the SI Estimate Frequency Response VI in Figure 6-7 is around 400 instead of 4096, as shown in Figure 6-3. You can reduce the window length by a factor of 10 because the number of samples in the reduced data set is one tenth of the number of samples in the raw data set.

Figure 6-8 shows the frequency response after applying a filter to and downsampling the raw data set.

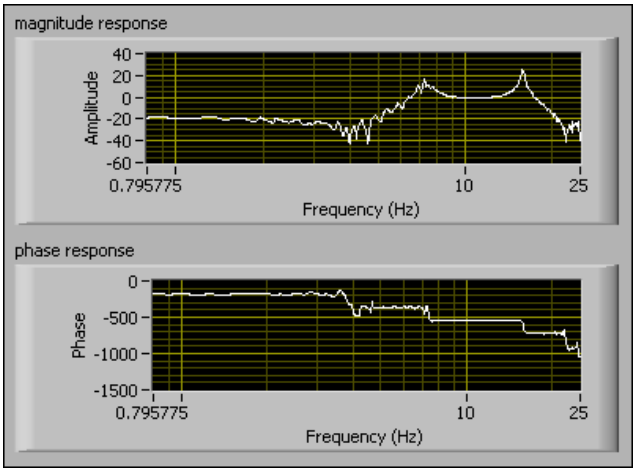

Figure 6-8. Frequency Response of the Filtered Data Set after Downsampling

Filtering and downsampling is beneficial because they eliminated the nonrealistic parts of the frequency response and reduced the amount of work required in the model estimation process.

Refer to Chapter 2, *Acquiring and Preprocessing Data*, for more information about filtering and downsampling data.

Estimating the Model

One of the biggest challenges in model estimation is selecting the correct model and the order of the model. The System Identification Toolkit supports three different criteria to aid in the estimation of the order of a model.

- FPE—Akaike's Final Prediction Error Criterion
- AIC—Akaike's Information Criterion
- MDL—Minimum Data Length criterion

Sometimes the results you obtain with these three criteria might be inconsistent. You can use a pole-zero plot for further investigation and to verify the results of the order estimation. Refer to the *Akaike's Information Criterion*, *Akaike's Final Prediction Error Criterion*, and *Minimum Data*

Length Criterion sections of Chapter 4, *Parametric Model Estimation Methods*, for more information about these criteria.

Figure 6-9 shows a prediction error plot generated by the SI Estimate Orders of System Model VI. The y-axis is the prediction error and the x-axis is the model dimension. For an ARX model, the model dimension is equal to the sum of the A order, B order, and delay values. The three different color bars on the chart represent the FPE, AIC, and MDL criteria. The lowest prediction error usually corresponds to the optimal order.

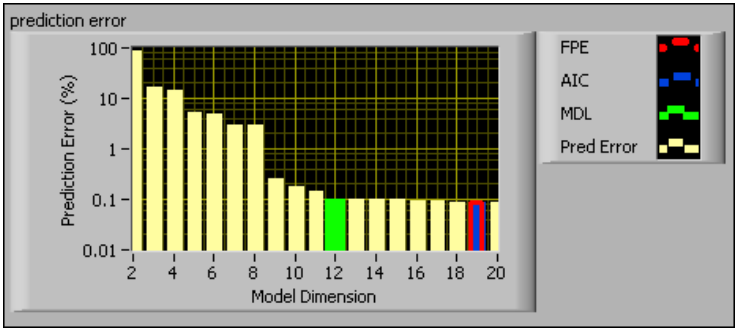

Figure 6-9. Prediction Error Plot for an ARX Model

The following sections show you how to use the AIC, MDL, and a user-defined criterion to determine the A and B orders of an ARX model.

Akaike's Information Criterion

The block diagram in Figure 6-10 uses the SI Estimate Orders of System Model VI for order estimation. To estimate the orders of a model, the SI Estimated Orders of System Model VI requires two data sets—one for estimation and one for validation. You do not need to acquire two data sets from a system, rather, you can partition one data set into two using the SI Split Signals VI. The SI Split Signals VI divides the preprocessing data samples into a portion for model estimation and a portion for model validation.

In Figure 6-10, the **1st portion** is 66, which means the SI Estimate Orders of System Model VI will use 66% of the data samples for estimation and the remainder of the data samples for validation.

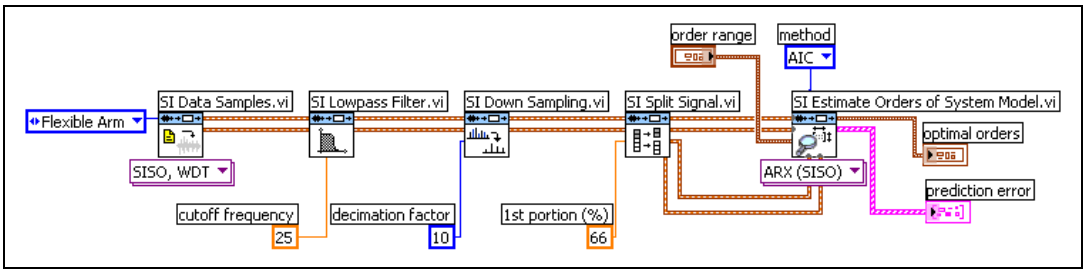

Figure 6-10. Estimate Orders of ARX Model VI

The SI Estimate Orders of System Model VI generates the **prediction error** plot for the ARX model and the optimal A order and B order based on the AIC criterion. By using the AIC criterion, the lowest prediction error corresponds a model dimension of 19, as shown in Figure 6-9, which corresponds to the following optimal orders:

- A order = 9
- B order = 10
- delay = 0

Verifying the Results

After determining the orders of the model, you want to verify the results to ensure the model accurately describes the system. One method is to plot a pole-zero map and visually inspect the plot to determine whether there is any redundancy in the data. If a pole and a zero overlap, the pole and zero cancel out each other, which indicates the estimated optimal order is too high.

The **Pole-Zero Plot** graph in Figure 6-11 shows a pole-zero plot with three overlapping pole-zero pairs. Due to numerical error, it is unlikely that a zero and a pole perfectly overlap. You can use the confidence region to justify whether the pole and the zero cancel out each other.

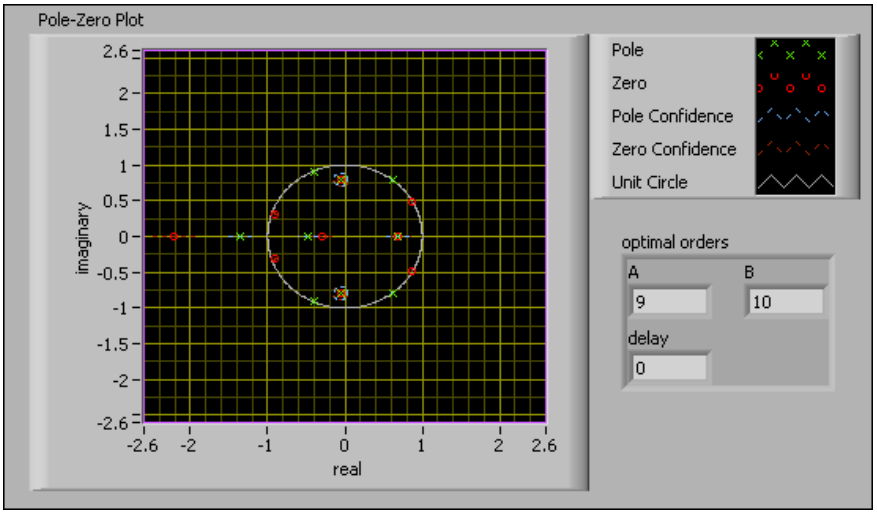

Figure 6-11. Pole-Zero Plot for an ARX Model

Because there are three pole-zero pairs, you can conclude that the AIC criterion does not produce the most optimal orders.

Minimum Data Length Criterion

Because the AIC criterion produced a model with non-optimal orders, you can try estimating the model orders with the MDL criterion. By using the MDL criterion, the lowest prediction error corresponds a model dimension of 12, as shown in Figure 6-9, which corresponds to the following optimal orders:

- A order = 6
- B order = 6
- delay = 0

Figure 6-12 shows a pole-zero plot of a model with a model dimension of 12.

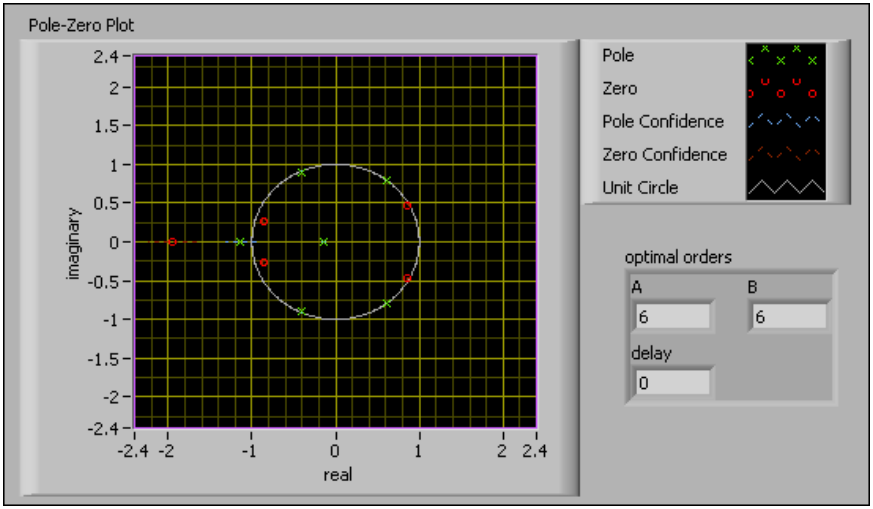

Figure 6-12. Pole-Zero Plot for a MDL Model

Compare Figure 6-12, which uses the MDL criterion and Figure 6-11, which uses the AIC criterion. Because there are no overlapping pole-zero pairs in Figure 6-11, you can conclude that the MDL criterion fits better than AIC criterion in this particular example.

In addition to examining redundancy, you also can use the pole-zero plot for other purposes. For example, both Figure 6-11 and Figure 6-12 show poles outside the unit circle. Having poles outside the unit circle implies that this model is not optimal because the ARX system based on the AIC or MDL criteria is unstable. One way to stabilize the system is to change the order.

In addition to the FPE, AIC, and MDL criteria, you can set user-defined orders in the SI Estimate Orders of System Model VI.

User-Defined Criterion

If you know nothing about the system, you might have to rely on trial and error to determine the optimal orders of the model. However, if you have some knowledge about a system, you can customize the estimation to find a model that fits a certain model dimension. For this model, assume you know that the system is stable, therefore, there should not be any poles outside the unit circle. Therefore, because both the AIC and MDL criterion did not produce stable models, the model orders do not describe the system accurately.

On the block diagram, as shown in Figure 6-13, you can customize the **method** parameter. Instead of **AIC** or **MDL**, you can select **<Other>** and enter the desired model dimension in the textbox. Assume you know that the model dimension is nine.

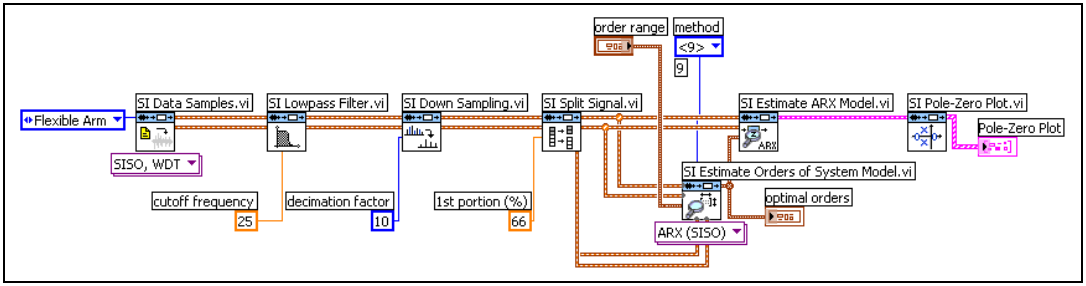

Figure 6-13. Pole-Zero Cancellation VI

Figure 6-14 shows the corresponding **Pole-Zero Plot** graph with a model dimension of nine, which corresponds to the following optimal orders:

- A order = 4
- B order = 5
- delay = 0

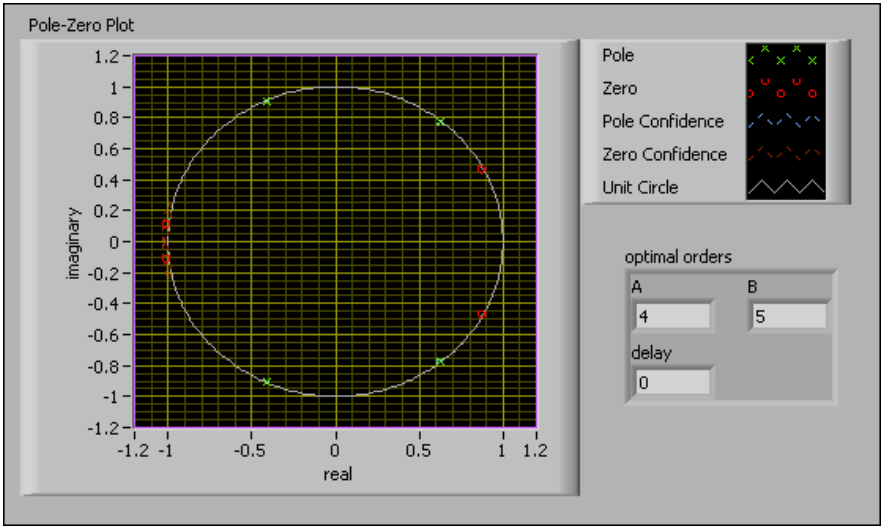

Figure 6-14. Pole-Zero Plot with User-Defined Orders

Compare the pole-zero plot in Figure 6-14 with Figure 6-11 and Figure 6-12. Figure 6-14, which has no overlapping pole-zero pairs and all

the poles are within the unit circle. By visually inspecting the pole-zero plot, you can see that this model is stable and not redundant. Using these model orders, you now can estimate and verify the system model.

ARX Model Validation

The goal of model validation is to determine whether or not the estimated model accurately reflects the actual system. Using the model orders found in the *User-Defined Criterion* section, you can simulate and predict the response of the system. You can compare these responses to the actual response and determine the accuracy of the estimated model. You also can analyze the residuals to determine the accuracy of the estimated model. Refer to the *Validating Models* section of Chapter 4, *Parametric Model Estimation Methods*, for more information about validating a model.

The following sections describe how to apply these techniques to model validation.

Simulation and Prediction

You can use the SI Model Simulation VI and SI Model Prediction VI to help you determine the accuracy of the estimated model. The SI Model Simulation VI simulates the system model and the SI Model Prediction VI performs a prediction of the system model. The results of the SI Model Prediction VI might differ from the SI Model Simulation VI because the SI Model Prediction VI periodically makes corrections to the estimated response based on the actual response of the system.

Figure 6-15 shows how you use these VIs to verify the ARX model created in the *User-Defined Criterion* section.

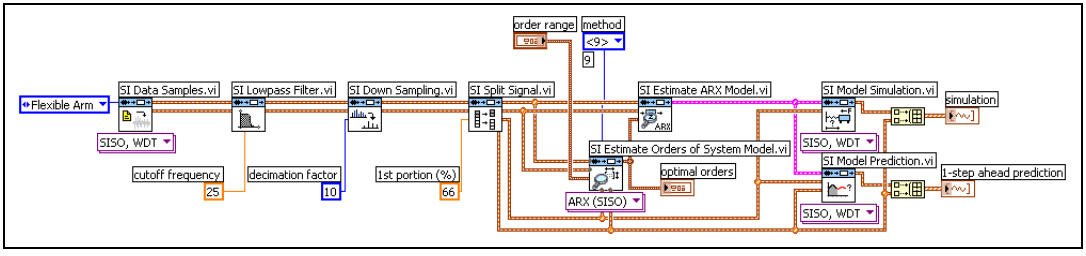

Figure 6-15. Simulation & Prediction with ARX Model VI

The outputs, **simulation** and **1-step ahead prediction**, enable you to visually determine how accurate the model is. Figure 6-16 shows the results of the simulation and prediction as well as the actual response of the system.

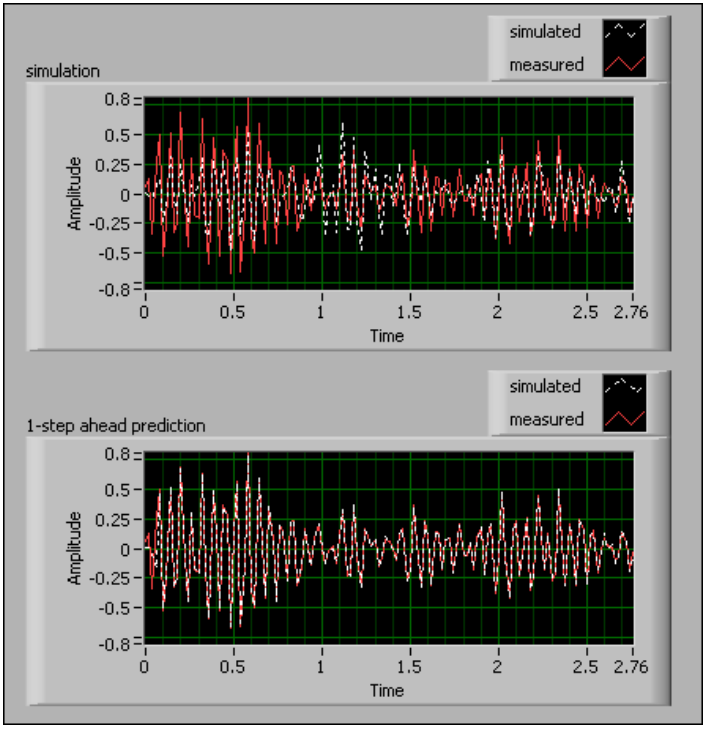

Figure 6-16. Simulation and Prediction Graphs for an ARX Model

Notice how the actual response, or the **measured response**, is different from the **simulated response** in the **simulation** graph. The SI Model Simulation VI simulates the response of the system without considering the actual response of and the noise dynamics in the system.

Residual Analysis

In addition to simulation and prediction, you can perform a residual analysis to validate the system model. Residual analysis tests whether the prediction error correlates to the stimulus signal. Prediction errors are usually uncorrelated with all stimulus signals in an open-loop system.

The block diagram in Figure 6-17 shows how you can use the SI Model Residual Analysis VI to with the ARX model identified in the *User-Defined Criterion* section to analyze the residuals.

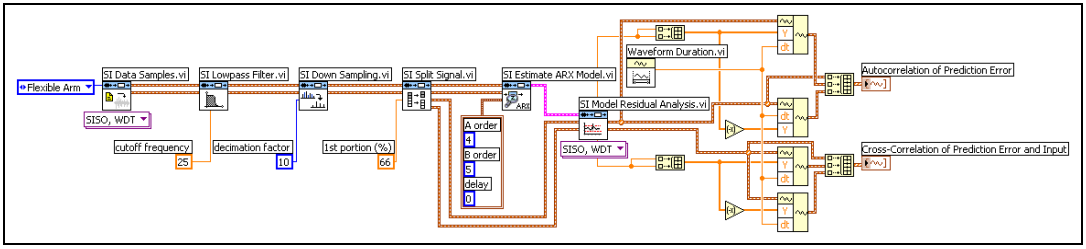

Figure 6-17. Residual Analysis VI

Figure 6-18 shows an example of ideal results where both autocorrelation and cross correlation are inside the confidence region except those in the vicinity of $\tau = 0$. This result indicates that the estimated model accurately describes the system.

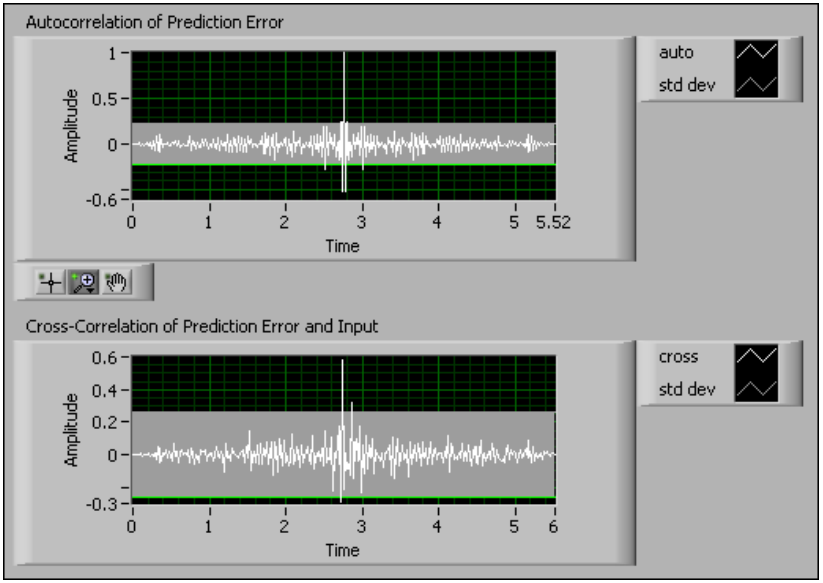

Figure 6-18. Residual Analysis for A order = 4, B order = 5, and delay = 0

When you verify and validate the identified model, you must use multiple analysis techniques to determine if the estimated model accurately represents the system. Some analysis techniques can be misleading. For example, if you performed a residual analysis on the model identified in the

Minimum Data Length Criterion section, you might conclude that this model is an accurate representation of the system. Figure 6-19 shows the autocorrelation and cross correlation residual analysis for the model in the *Minimum Data Length Criterion* section. Recall that this model has the following orders:

- A order = 6
- B order = 6
- delay = 0

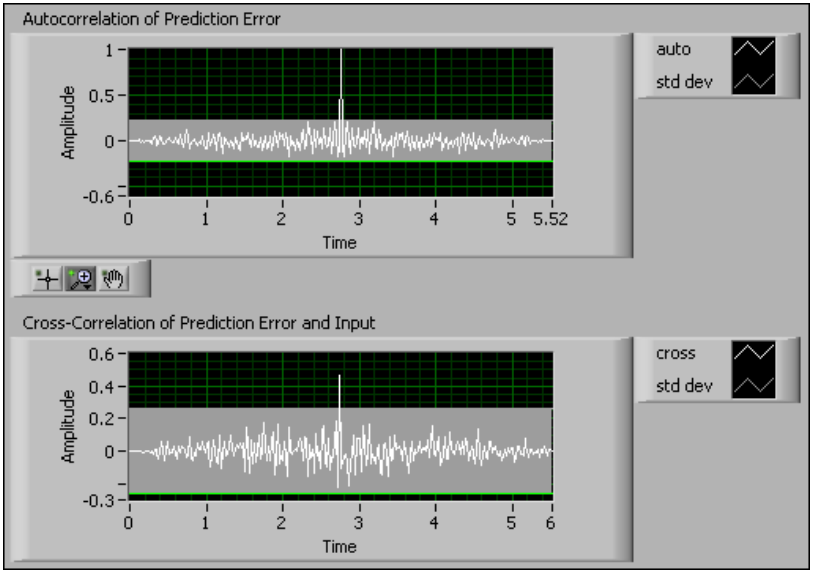

Figure 6-19. Residual Analysis of ARX Model with A Order= 6, B Order = 6, and Delay = 0

Figure 6-19 shows that both the autocorrelation and cross correlation are inside the confidence region. Therefore, without performing any other analyses, you might conclude that this model is an accurate representation of the system. However, the pole-zero analysis in the *Minimum Data Length Criterion* section showed poles outside of the unit circle. So you already determined that this model is unstable. Thus, despite acceptable autocorrelation and cross correlation values, concluding that this model is accurate is incorrect.

Thus, if you only performed a residual analysis, you might not discover that this model is actually unstable. When validating a model, perform multiple analyses to ensure the accuracy of the model.

Estimating a State-Space Model

For a state-space model, order estimation is equivalent to estimating the number of significant singular values, which correspond to the number of states in the model. After identifying a state-space model that represents the system, you can use the same validation and verification technique used in the *Simulation and Prediction* and *Residual Analysis* sections.

Refer to the *State-Space Model* section of Chapter 4, *Parametric Model Estimation Methods*, for more information about estimating state-space models.

The examples in this section use the same flexible robotic arm data and the same preprocessing techniques.

Finding the Singular Values

The block diagram in Figure 6-20 shows how to use the SI Estimate Orders of System Model VI to find the optimal order and the number of significant singular values.

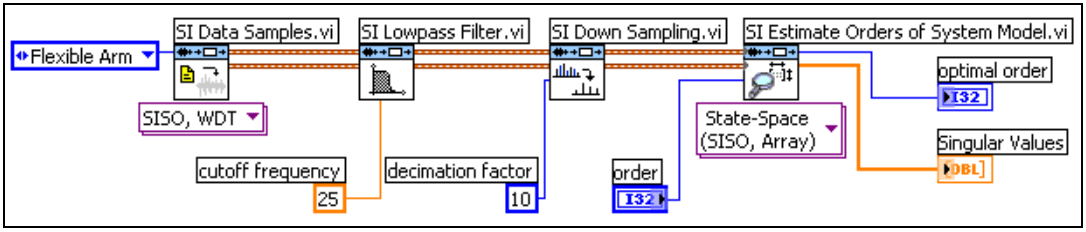

Figure 6-20. Estimate Orders of State-Space Model VI

The **Singular Values** graph in Figure 6-21 shows a singular value plot with four leading singular values.

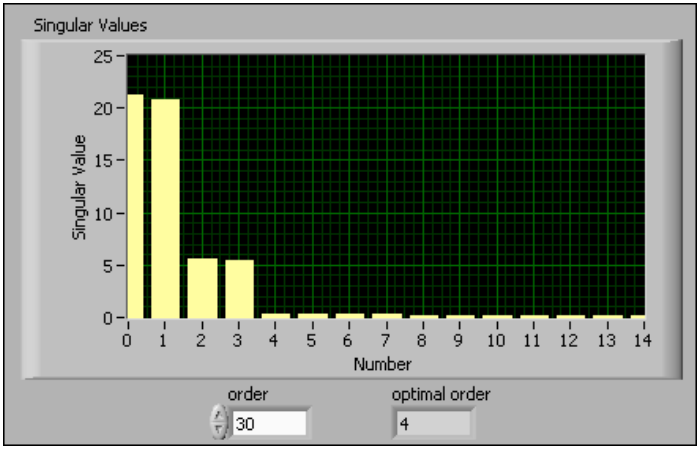

Figure 6-21. Singular Value Plot for State-Space Model

By looking both at the **Singular Values** graph and the **optimal order**, you can see that there are four states in this state-space model.

Validating the Estimated State-Space Model

You can validate the state-space model in the same way that you validated the ARX model. You use the SI Model Simulation VI and the SI Model Prediction VI to determine the accuracy of the state-space model.

Figure 6-22 shows the complete process, from estimating the state-space model to simulating and predicting the response of the model.

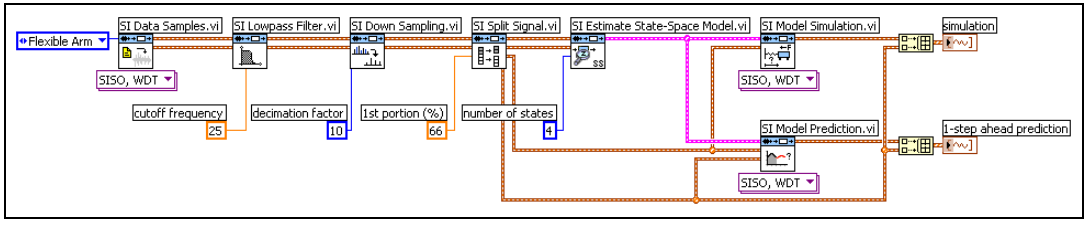

Figure 6-22. Simulation & Prediction with State-Space Model VI

The **simulation** and **1-step ahead prediction** graphs in Figure 6-22 show simulation and prediction plots for a state-space model.

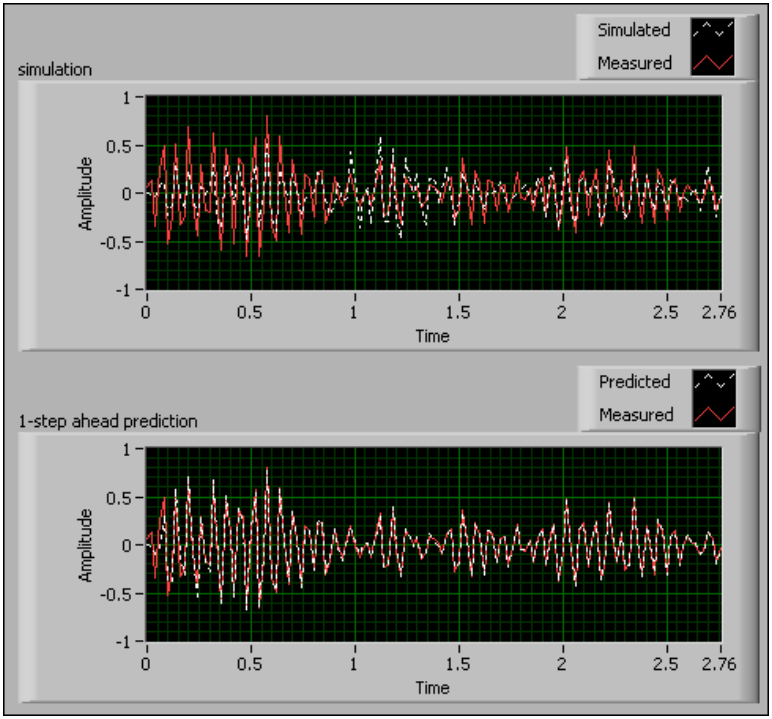

Figure 6-23. Simulation and Prediction Graphs for a State-Space Model

Refer to the *Validating Models* section of Chapter 4, *Parametric Model Estimation Methods*, for more information about validating a model.

Additional Examples

In this chapter, you learned how to start from raw data and find an accurate model to represent the system from which you acquired the data. The examples in this chapter are located in `labview\examples\system identification\SICaseStudy1.llb`.

The `labview\examples\system identification\` directory also contains other examples you can use to become familiar with the System Identification VIs.

Refer to the *LabVIEW Help*, available by selecting **Help»VI, Function, & How-To Help**, for information about specific System Identification VIs in this case study.

A

References

This manual contains information about using the LabVIEW System Identification Toolkit VIs in LabVIEW applications. In most cases, the underlying theory is omitted. If you are interested in the rigorous mathematical treatment of system identification techniques and algorithms, refer to the following textbooks and technical papers, which National Instruments used to develop this toolkit.

Astrom, K. J., and B. Wittenmark. 1974. *Adaptive control*. 2d ed. Addison-Wesley.

De Moor, B. 1988. *Mathematical concepts and techniques for modelling of static and dynamic systems*. Ph.D. thesis. Katholieke Universiteit Leuven, UDC 519.17.

Gill, P. E., W. Murray, and M. H. Wright. 1981. *Practical optimization*. Academic Press.

Goodwin, G. C., and R. L. Payne. 1977. *Dynamic system identification: Experiment design and data analysis*. Academic Press.

Kailath, T. 1980. *Linear systems*. Prentice Hall.

Ljung, L. 1999. *System Identification Theory for the User*. 2d ed. Prentice Hall.

Ljung, L., and T. Glad. 1994. *Modeling of dynamic systems*. Prentice Hall.

Ljung, L., and T. Söderström. 1983. *Theory and practice of recursive identification*. MIT Press.

Oppenheim, A. V., A. S. Willsky, and I. T. Young. 1997. *Signals and systems*. Prentice Hall.

Oppenheim, A. V., and R. W. Schafer. 1989. *Discrete-time signal processing*. 2d ed. Prentice Hall.

Overschee, P. V., and B. De Moor. 1993. *N4SID: Subspace algorithms for the stochastic identification problem*. Automatica 29, no. 3:649–660.

———. 1994. *Subspace algorithms for the identification of combined deterministic-stochastic systems*. Automatica 30, no. 1:75-93.

Powell, M. J. D. 1964. *An efficient method for finding the minimum of a function of several variables without calculating derivatives*. Computer Journal 7:155–162.

Priestley, M. 1981. *Spectral analysis and time series*. Academic Press.

B

Technical Support and Professional Services

Visit the following sections of the National Instruments Web site at ni.com for technical support and professional services:

- **Support**—Online technical support resources at ni.com/support include the following:
  - **Self-Help Resources**—For answers and solutions, visit the award-winning National Instruments Web site for software drivers and updates, a searchable KnowledgeBase, product manuals, step-by-step troubleshooting wizards, thousands of example programs, tutorials, application notes, instrument drivers, and so on.
  - **Free Technical Support**—All registered users receive free Basic Service, which includes access to hundreds of Application Engineers worldwide in the NI Developer Exchange at ni.com/exchange. National Instruments Application Engineers make sure every question receives an answer.

    For information about other technical support options in your area, go to ni.com/services or contact your local branch at ni.com/contact.
- **Training and Certification**—Visit ni.com/training for self-paced training, eLearning virtual classrooms, interactive CDs, and Certification program information. You also can register for instructor-led, hands-on courses at locations around the world.
- **System Integration**—If you have time constraints, limited in-house technical resources, or other project challenges, National Instruments Alliance Partner members can help. To learn more, call your local NI office or visit ni.com/alliance.

If you searched ni.com and could not find the answers you need, contact your local office or NI corporate headquarters. Phone numbers for our worldwide offices are listed at the front of this manual. You also can visit the Worldwide Offices section of ni.com/niglobal to access the branch office Web sites, which provide up-to-date contact information, support phone numbers, email addresses, and current events.