LabVIEW™

Analysis Concepts

March 2004 Edition
Part Number 370192C-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) 514 685 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, Greece 30 2 10 42 96 427, 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 001 800 010 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 the documentation, send email to techpubs@ni.com.

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

CVI™, LabVIEW™, National Instruments™, NI™, and ni.com™ are trademarks of National Instruments Corporation.

MATLAB® is a registered trademark of The MathWorks, Inc. Other product and company names mentioned herein are trademarks or trade names of their respective companies.

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

Chapter 4
Frequency Analysis

Chapter 5
Smoothing Windows

Chapter 8
Limit Testing

Part II
Mathematics

Chapter 9
Curve Fitting

Chapter 10
Probability and Statistics

Chapter 13
Polynomials

Part III
Point-By-Point Analysis

Chapter 14
Point-By-Point Analysis

About This Manual

This manual describes analysis and mathematical concepts in LabVIEW. The information in this manual directly relates to how you can use LabVIEW to perform analysis and measurement operations.

Conventions

This manual uses the following conventions:

»	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.

Related Documentation

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

- *LabVIEW Measurements Manual*
- *The Fundamentals of FFT-Based Signal Analysis and Measurement in LabVIEW and LabWindows™/CVI™* Application Note, available on the National Instruments Web site at `ni.com/info`, where you enter the info code `rdlv04`

- *LabVIEW Help*, available by selecting **Help»VI, Function, & How-To Help**
- *LabVIEW User Manual*
- *Getting Started with LabVIEW*
- *On the Use of Windows for Harmonic Analysis with the Discrete Fourier Transform* (Proceedings of the IEEE, Volume 66, No. 1, January 1978)

Part I

Signal Processing and Signal Analysis

This part describes signal processing and signal analysis concepts.

- Chapter 1, *Introduction to Digital Signal Processing and Analysis in LabVIEW*, provides a background in basic digital signal processing and an introduction to signal processing and measurement VIs in LabVIEW.
- Chapter 2, *Signal Generation*, describes the fundamentals of signal generation.
- Chapter 3, *Digital Filtering*, introduces the concept of filtering, compares analog and digital filters, describes finite infinite response (FIR) and infinite impulse response (IIR) filters, and describes how to choose the appropriate digital filter for a particular application.
- Chapter 4, *Frequency Analysis*, describes the fundamentals of the discrete Fourier transform (DFT), the fast Fourier transform (FFT), basic signal analysis computations, computations performed on the power spectrum, and how to use FFT-based functions for network measurement.
- Chapter 5, *Smoothing Windows*, describes spectral leakage, how to use smoothing windows to decrease spectral leakage, the different types of smoothing windows, how to choose the correct type of smoothing window, the differences between smoothing windows used for spectral analysis and smoothing windows used for filter coefficient design, and the importance of scaling smoothing windows.

- Chapter 6, *Distortion Measurements*, describes harmonic distortion, total harmonic distortion (THD), signal noise and distortion (SINAD), and when to use distortion measurements.
- Chapter 7, *DC/RMS Measurements*, introduces measurement analysis techniques for making DC and RMS measurements of a signal.
- Chapter 8, *Limit Testing*, provides information about setting up an automated system for performing limit testing, specifying limits, and applications for limit testing.

1

Introduction to Digital Signal Processing and Analysis in LabVIEW

Digital signals are everywhere in the world around us. Telephone companies use digital signals to represent the human voice. Radio, television, and hi-fi sound systems are all gradually converting to the digital domain because of its superior fidelity, noise reduction, and signal processing flexibility. Data is transmitted from satellites to earth ground stations in digital form. NASA pictures of distant planets and outer space are often processed digitally to remove noise and extract useful information. Economic data, census results, and stock market prices are all available in digital form. Because of the many advantages of digital signal processing, analog signals also are converted to digital form before they are processed with a computer.

This chapter provides a background in basic digital signal processing and an introduction to signal processing and measurement VIs in LabVIEW.

The Importance of Data Analysis

The importance of integrating analysis libraries into engineering stations is that the raw data, as shown in Figure 1-1, does not always immediately convey useful information. Often, you must transform the signal, remove noise disturbances, correct for data corrupted by faulty equipment, or compensate for environmental effects, such as temperature and humidity.

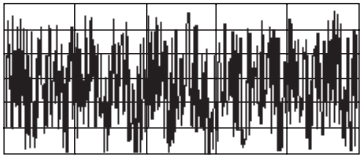


Figure 1-1. Raw Data

By analyzing and processing the digital data, you can extract the useful information from the noise and present it in a form more comprehensible than the raw data, as shown in Figure 1-2.

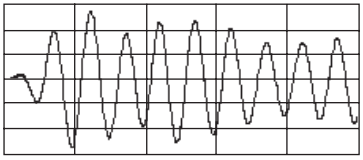


Figure 1-2. Processed Data

The LabVIEW block diagram programming approach and the extensive set of LabVIEW signal processing and measurement VIs simplify the development of analysis applications.

Sampling Signals

Measuring the frequency content of a signal requires digitalization of a continuous signal. To use digital signal processing techniques, you must first convert an analog signal into its digital representation. In practice, the conversion is implemented by using an analog-to-digital (A/D) converter. Consider an analog signal $x(t)$ that is sampled every Δt seconds. The time interval Δt is the sampling interval or sampling period. Its reciprocal, $1/\Delta t$, is the sampling frequency, with units of samples/second. Each of the discrete values of $x(t)$ at $t = 0$, Δt, $2\Delta t$, $3\Delta t$, and so on, is a sample. Thus, $x(0)$, $x(\Delta t)$, $x(2\Delta t)$, …, are all samples. The signal $x(t)$ thus can be represented by the following discrete set of samples.

$$\{x(0), x(\Delta t), x(2\Delta t), x(3\Delta t), \ldots, x(k\Delta t), \ldots\}$$

Figure 1-3 shows an analog signal and its corresponding sampled version. The sampling interval is Δt. The samples are defined at discrete points in time.

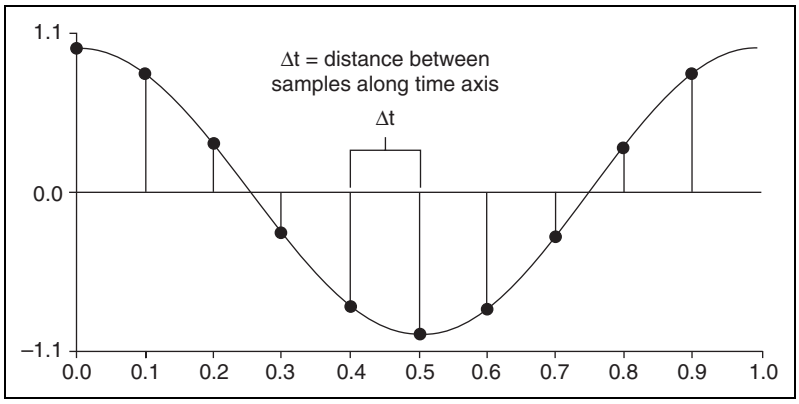


Figure 1-3. Analog Signal and Corresponding Sampled Version

The following notation represents the individual samples.

$$x[i] = x(i\Delta t)$$

for

$$i = 0, 1, 2, \ldots$$

If N samples are obtained from the signal $x(t)$, then you can represent $x(t)$ by the following sequence.

$$X = \{x[0], x[1], x[2], x[3], \ldots, x[N-1]\}$$

The preceding sequence representing $x(t)$ is the digital representation, or the sampled version, of $x(t)$. The sequence $X = \{x[i]\}$ is indexed on the integer variable i and does not contain any information about the sampling rate. So knowing only the values of the samples contained in X gives you no information about the sampling frequency.

One of the most important parameters of an analog input system is the frequency at which the DAQ device samples an incoming signal. The sampling frequency determines how often an A/D conversion takes place. Sampling a signal too slowly can result in an aliased signal.

Aliasing

An aliased signal provides a poor representation of the analog signal. Aliasing causes a false lower frequency component to appear in the sampled data of a signal. Figure 1-4 shows an adequately sampled signal and an undersampled signal.

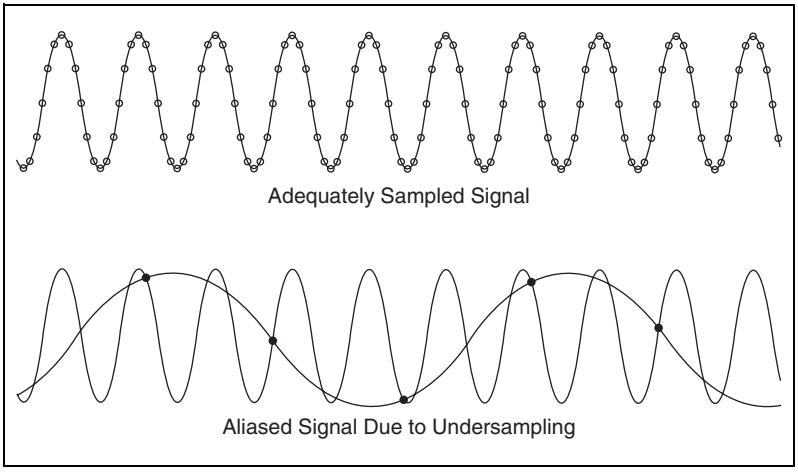


Figure 1-4. Aliasing Effects of an Improper Sampling Rate

In Figure 1-4, the undersampled signal appears to have a lower frequency than the actual signal—three cycles instead of ten cycles.

Increasing the sampling frequency increases the number of data points acquired in a given time period. Often, a fast sampling frequency provides a better representation of the original signal than a slower sampling rate.

For a given sampling frequency, the maximum frequency you can accurately represent without aliasing is the Nyquist frequency. The Nyquist frequency equals one-half the sampling frequency, as shown by the following equation.

$$f_N = \frac{f_s}{2},$$

where f_N is the Nyquist frequency and f_s is the sampling frequency.

Signals with frequency components above the Nyquist frequency appear aliased between DC and the Nyquist frequency. In an aliased signal, frequency components actually above the Nyquist frequency appear as

frequency components below the Nyquist frequency. For example, a component at frequency $f_N < f_0 < f_s$ appears as the frequency $f_s - f_0$.

Figures 1-5 and 1-6 illustrate the aliasing phenomenon. Figure 1-5 shows the frequencies contained in an input signal acquired at a sampling frequency, f_s, of 100 Hz.

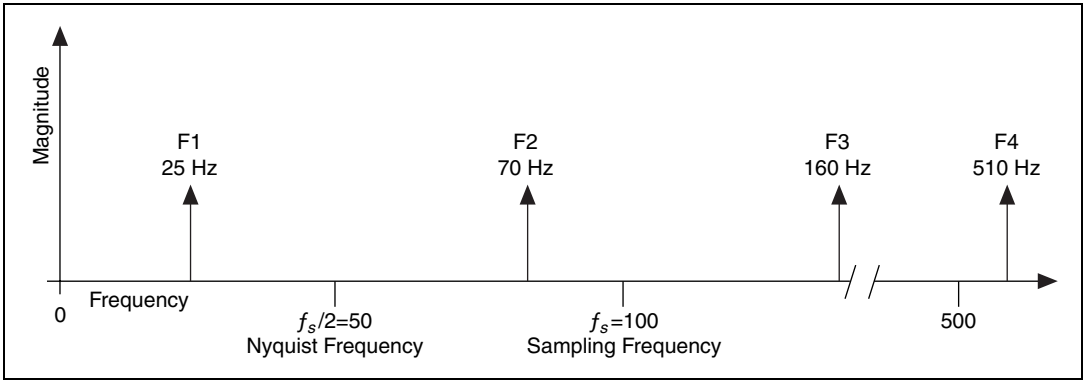


Figure 1-5. Actual Signal Frequency Components

Figure 1-6 shows the frequency components and the aliases for the input signal from Figure 1-5.

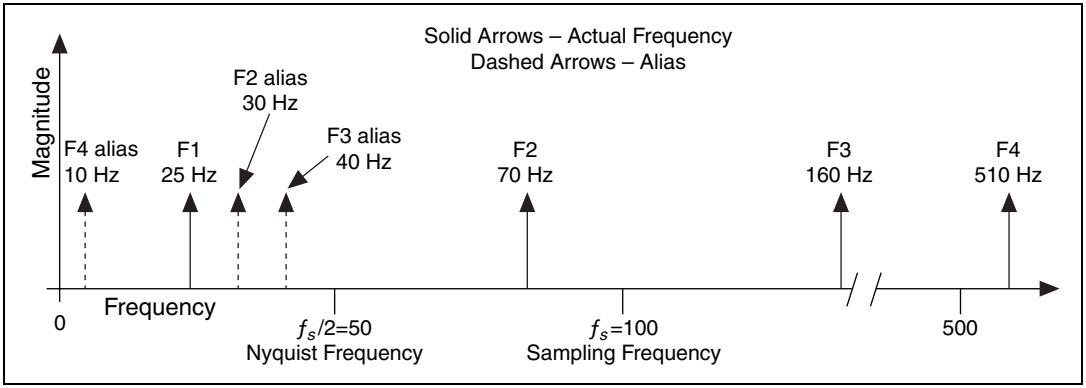


Figure 1-6. Signal Frequency Components and Aliases

In Figure 1-6, frequencies below the Nyquist frequency of $f_s/2$ = 50 Hz are sampled correctly. For example, F1 appears at the correct frequency. Frequencies above the Nyquist frequency appear as aliases. For example, aliases for F2, F3, and F4 appear at 30 Hz, 40 Hz, and 10 Hz, respectively.

The alias frequency equals the absolute value of the difference between the closest integer multiple of the sampling frequency and the input frequency, as shown in the following equation.

$$AF = |CIMSF - IF|$$

where *AF* is the alias frequency, *CIMSF* is the closest integer multiple of the sampling frequency, and *IF* is the input frequency. For example, you can compute the alias frequencies for F2, F3, and F4 from Figure 1-6 with the following equations.

$$\text{Alias F2} = |100 - 70| = 30 \text{ Hz}$$
$$\text{Alias F3} = |(2)100 - 160| = 40 \text{ Hz}$$
$$\text{Alias F4} = |(5)100 - 510| = 10 \text{ Hz}$$

Increasing Sampling Frequency to Avoid Aliasing

According to the Shannon Sampling Theorem, use a sampling frequency at least twice the maximum frequency component in the sampled signal to avoid aliasing. Figure 1-7 shows the effects of various sampling frequencies.

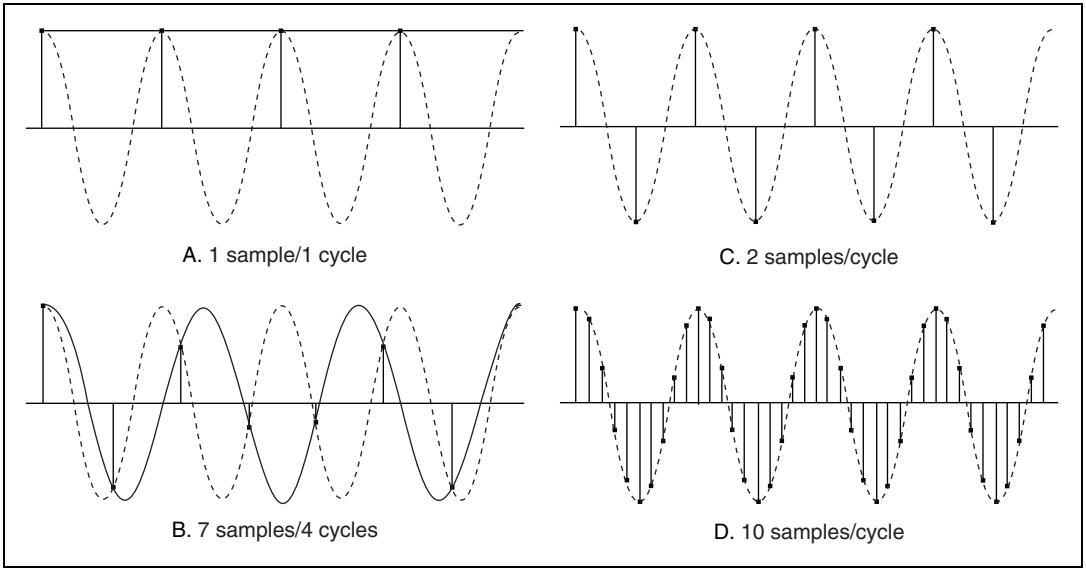


Figure 1-7. Effects of Sampling at Different Rates

In case A of Figure 1-7, the sampling frequency f_s equals the frequency f of the sine wave. f_s is measured in samples/second. f is measured in cycles/second. Therefore, in case A, one sample per cycle is acquired. The reconstructed waveform appears as an alias at DC.

In case B of Figure 1-7, $f_s = 7/4f$, or 7 samples/4 cycles. In case B, increasing the sampling rate increases the frequency of the waveform. However, the signal aliases to a frequency less than the original signal—three cycles instead of four.

In case C of Figure 1-7, increasing the sampling rate to $f_s = 2f$ results in the digitized waveform having the correct frequency or the same number of cycles as the original signal. In case C, the reconstructed waveform more accurately represents the original sinusoidal wave than case A or case B. By increasing the sampling rate to well above f, for example, $f_s = 10f = 10$ samples/cycle, you can accurately reproduce the waveform. Case D of Figure 1-7 shows the result of increasing the sampling rate to $f_s = 10f$.

Anti-Aliasing Filters

In the digital domain, you cannot distinguish alias frequencies from the frequencies that actually lie between 0 and the Nyquist frequency. Even with a sampling frequency equal to twice the Nyquist frequency, pickup from stray signals, such as signals from power lines or local radio stations, can contain frequencies higher than the Nyquist frequency. Frequency components of stray signals above the Nyquist frequency might alias into the desired frequency range of a test signal and cause erroneous results. Therefore, you need to remove alias frequencies from an analog signal before the signal reaches the A/D converter.

Use an anti-aliasing analog lowpass filter before the A/D converter to remove alias frequencies higher than the Nyquist frequency. A lowpass filter allows low frequencies to pass but attenuates high frequencies. By attenuating the frequencies higher than the Nyquist frequency, the anti-aliasing analog lowpass filter prevents the sampling of aliasing components. An anti-aliasing analog lowpass filter should exhibit a flat passband frequency response with a good high-frequency alias rejection and a fast roll-off in the transition band. Because you apply the anti-aliasing filter to the analog signal before it is converted to a digital signal, the anti-aliasing filter is an analog filter.

Figure 1-8 shows both an ideal anti-alias filter and a practical anti-alias filter. The following information applies to Figure 1-8:

- f_1 is the maximum input frequency.
- Frequencies less than f_1 are desired frequencies.
- Frequencies greater than f_1 are undesired frequencies.

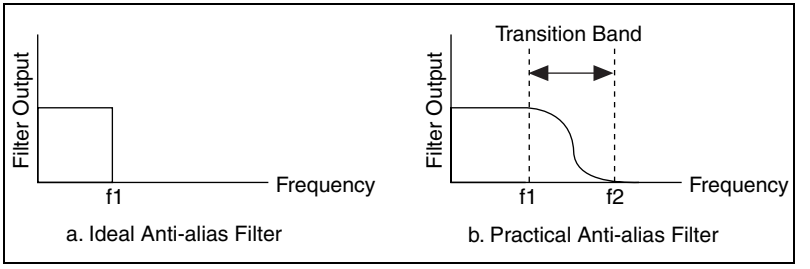


Figure 1-8. Ideal versus Practical Anti-Alias Filter

An ideal anti-alias filter, shown in Figure 1-8a, passes all the desired input frequencies and cuts off all the undesired frequencies. However, an ideal anti-alias filter is not physically realizable.

Figure 1-8b illustrates actual anti-alias filter behavior. Practical anti-alias filters pass all frequencies $<f_1$ and cut off all frequencies $>f_2$. The region between f_1 and f_2 is the transition band, which contains a gradual attenuation of the input frequencies. Although you want to pass only signals with frequencies $<f_1$, the signals in the transition band might cause aliasing. Therefore, in practice, use a sampling frequency greater than two times the highest frequency in the transition band. Using a sampling frequency greater than two times the highest frequency in the transition band means f_s might be more than $2f_1$.

Converting to Logarithmic Units

On some instruments, you can display amplitude on either a linear scale or a decibel (dB) scale. The linear scale shows the amplitudes as they are. The decibel is a unit of ratio. The decibel scale is a transformation of the linear scale into a logarithmic scale.

The following equations define the decibel. Equation 1-1 defines the decibel in terms of power. Equation 1-2 defines the decibel in terms of amplitude.

$$\text{dB} = 10\log_{10}\frac{P}{P_r}, \quad (1\text{-}1)$$

where P is the measured power, P_r is the reference power, and $\frac{P}{P_r}$ is the power ratio.

$$\text{dB} = 20\log_{10}\frac{A}{A_r}, \quad (1\text{-}2)$$

where A is the measured amplitude, A_r is the reference amplitude, and $\frac{A}{A_r}$ is the voltage ratio.

Equations 1-1 and 1-2 require a reference value to measure power and amplitude in decibels. The reference value serves as the 0 dB level. Several conventions exist for specifying a reference value. You can use the following common conventions to specify a reference value for calculating decibels:

- Use the reference one volt-rms squared (1V_{rms}^2) for power, which yields the unit of measure dBV_{rms}.
- Use the reference one volt-rms ($1\ \text{V}_{\text{rms}}$) for amplitude, which yields the unit of measure dBV.
- Use the reference 1 mW into a load of 50 Ω for radio frequencies where 0 dB is $0.22\ \text{V}_{\text{rms}}$, which yields the unit of measure dBm.
- Use the reference 1 mW into a load of 600 Ω for audio frequencies where 0 dB is $0.78\ \text{V}_{\text{rms}}$, which yields the unit of measure dBm.

When using amplitude or power as the amplitude-squared of the same signal, the resulting decibel level is exactly the same. Multiplying the decibel ratio by two is equivalent to having a squared ratio. Therefore, you obtain the same decibel level and display regardless of whether you use the amplitude or power spectrum.

Displaying Results on a Decibel Scale

Amplitude or power spectra usually are displayed on a decibel scale. Displaying amplitude or power spectra on a decibel scale allows you to view wide dynamic ranges and to see small signal components in the presence of large ones. For example, suppose you want to display a signal containing amplitudes from a minimum of 0.1 V to a maximum of 100 V

on a device with a display height of 10 cm. Using a linear scale, if the device requires the entire display height to display the 100 V amplitude, the device displays 10 V of amplitude per centimeter of height. If the device displays 10 V/cm, displaying the 0.1 V amplitude of the signal requires a height of only 0.1 mm. Because a height of 0.1 mm is barely visible on the display screen, you might overlook the 0.1 V amplitude component of the signal. Using a logarithmic scale in decibels allows you to see the 0.1 V amplitude component of the signal.

Table 1-1 shows the relationship between the decibel and the power and voltage ratios.

Table 1-1. Decibels and Power and Voltage Ratio Relationship

dB	Power Ratio	Amplitude Ratio
+40	10,000	100
+20	100	10
+6	4	2
+3	2	1.4
0	1	1
–3	1/2	1/1.4
–6	1/4	1/2
–20	1/100	1/10
–40	1/10,000	1/100

Table 1-1 shows how you can compress a wide range of amplitudes into a small set of numbers by using the logarithmic decibel scale.

2

Signal Generation

The generation of signals is an important part of any test or measurement system. The following applications are examples of uses for signal generation:

- Simulate signals to test your algorithm when real-world signals are not available, for example, when you do not have a DAQ device for obtaining real-world signals or when access to real-world signals is not possible.
- Generate signals to apply to a digital-to-analog (D/A) converter.

This chapter describes the fundamentals of signal generation.

Common Test Signals

Common test signals include the sine wave, the square wave, the triangle wave, the sawtooth wave, several types of noise waveforms, and multitone signals consisting of a superposition of sine waves.

The most common signal for audio testing is the sine wave. A single sine wave is often used to determine the amount of harmonic distortion introduced by a system. Multiple sine waves are widely used to measure the intermodulation distortion or to determine the frequency response. Table 2-1 lists the signals used for some typical measurements.

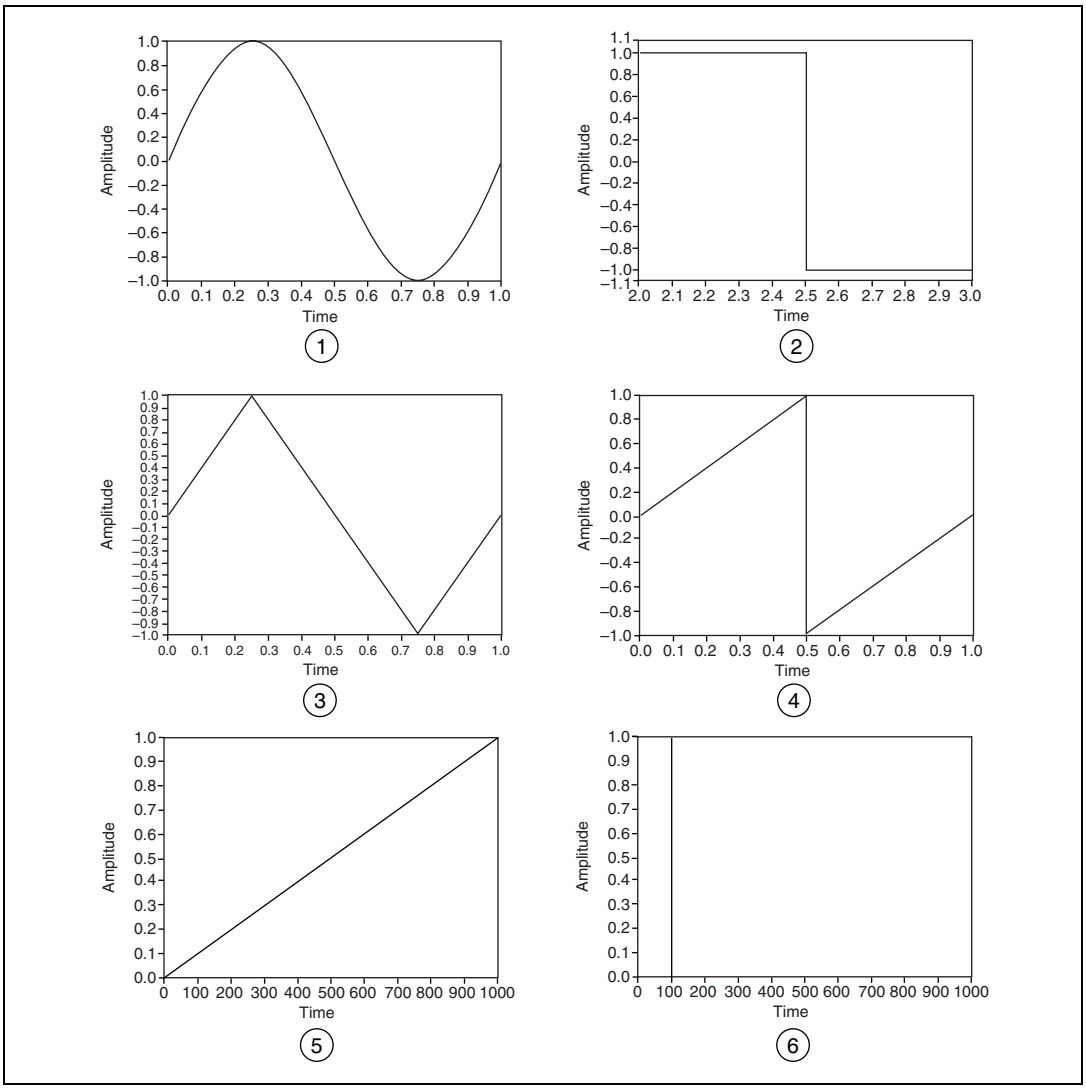
Table 2-1. Typical Measurements and Signals

Measurement	Signal
Total harmonic distortion	Sine wave
Intermodulation distortion	Multitone (two sine waves)
Frequency response	Multitone (many sine waves, impulse, chirp), broadband noise
Interpolation	Sinc

Table 2-1. Typical Measurements and Signals (Continued)

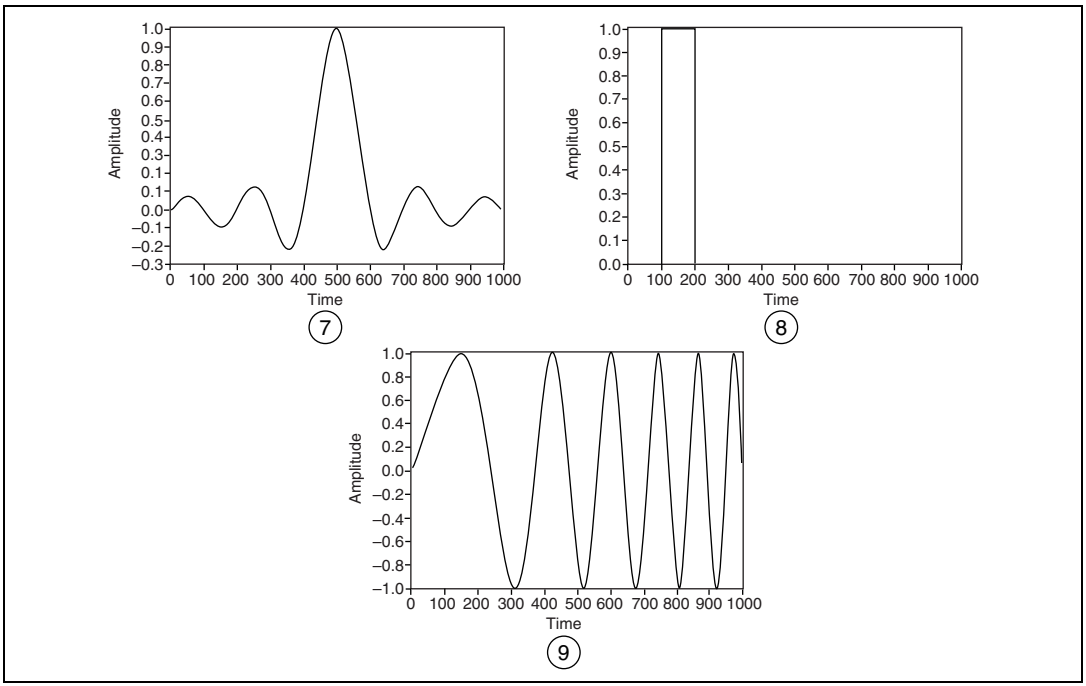
Measurement	Signal
Rise time, fall time, overshoot, undershoot	Pulse
Jitter	Square wave

These signals form the basis for many tests and are used to measure the response of a system to a particular stimulus. Some of the common test signals available in most signal generators are shown in Figure 2-1 and Figure 2-2.



1 Sine Wave	3 Triangle Wave	5 Ramp
2 Square Wave	4 Sawtooth Wave	6 Impulse

Figure 2-1. Common Test Signals



7 Sinc 8 Pulse 9 Chirp

Figure 2-2. More Common Test Signals

The most useful way to view the common test signals is in terms of their frequency content. The common test signals have the following frequency content characteristics:

- Sine waves have a single frequency component.
- Square waves consist of the superposition of many sine waves at odd harmonics of the fundamental frequency. The amplitude of each harmonic is inversely proportional to its frequency.
- Triangle and sawtooth waves have harmonic components that are multiples of the fundamental frequency.
- An impulse contains all frequencies that can be represented for a given sampling rate and number of samples.
- Chirp signals are sinusoids swept from a start frequency to a stop frequency, thus generating energy across a given frequency range. Chirp patterns have discrete frequencies that lie within a certain range. The discrete frequencies of chirp patterns depend on the sampling rate, the start and end frequencies, and the number of samples.

Frequency Response Measurements

To achieve a good frequency response measurement, the frequency range of interest must contain a significant amount of stimulus energy. Two common signals used for frequency response measurements are the chirp signal and a broadband noise signal, such as white noise. Refer to the *Common Test Signals* section of this chapter for information about chirp signals. Refer to the *Noise Generation* section of this chapter for information about white noise.

It is best not to use windows when analyzing frequency response signals. If you generate a chirp stimulus signal at the same rate you acquire the response, you can match the acquisition frame size to the length of the chirp. No window is generally the best choice for a broadband signal source. Because some stimulus signals are not constant in frequency across the time record, applying a window might obscure important portions of the transient response.

Multitone Generation

Except for the sine wave, the common test signals do not allow full control over their spectral content. For example, the harmonic components of a square wave are fixed in frequency, phase, and amplitude relative to the fundamental. However, you can generate multitone signals with a specific amplitude and phase for each individual frequency component.

A multitone signal is the superposition of several sine waves or tones, each with a distinct amplitude, phase, and frequency. A multitone signal is typically created so that an integer number of cycles of each individual tone are contained in the signal. If an FFT of the entire multitone signal is computed, each of the tones falls exactly onto a single frequency bin, which means no spectral spread or leakage occurs.

Multitone signals are a part of many test specifications and allow the fast and efficient stimulus of a system across an arbitrary band of frequencies. Multitone test signals are used to determine the frequency response of a device and with appropriate selection of frequencies, also can be used to measure such quantities as intermodulation distortion.

Crest Factor

The relative phases of the constituent tones with respect to each other determine the crest factor of a multitone signal with specified amplitude. The crest factor is defined as the ratio of the peak magnitude to the RMS value of the signal. For example, a sine wave has a crest factor of 1.414:1.

For the same maximum amplitude, a multitone signal with a large crest factor contains less energy than one with a smaller crest factor. In other words, a large crest factor means that the amplitude of a given component sine tone is lower than the same sine tone in a multitone signal with a smaller crest factor. A higher crest factor results in individual sine tones with lower signal-to-noise ratios. Therefore, proper selection of phases is critical to generating a useful multitone signal.

To avoid clipping, the maximum value of the multitone signal should not exceed the maximum capability of the hardware that generates the signal, which means a limit is placed on the maximum amplitude of the signal. You can generate a multitone signal with a specific amplitude by using different combinations of the phase relationships and amplitudes of the constituent sine tones. A good approach to generating a signal is to choose amplitudes and phases that result in a lower crest factor.

Phase Generation

The following schemes are used to generate tone phases of multitone signals:

- Varying the phase difference between adjacent frequency tones linearly from 0 to 360 degrees
- Varying the tone phases randomly

Varying the phase difference between adjacent frequency tones linearly from 0 to 360 degrees allows the creation of multitone signals with very low crest factors. However, the resulting multitone signals have the following potentially undesirable characteristics:

- The multitone signal is very sensitive to phase distortion. If in the course of generating the multitone signal the hardware or signal path induces non-linear phase distortion, the crest factor might vary considerably.
- The multitone signal might display some repetitive time-domain characteristics, as shown in the multitone signal in Figure 2-3.

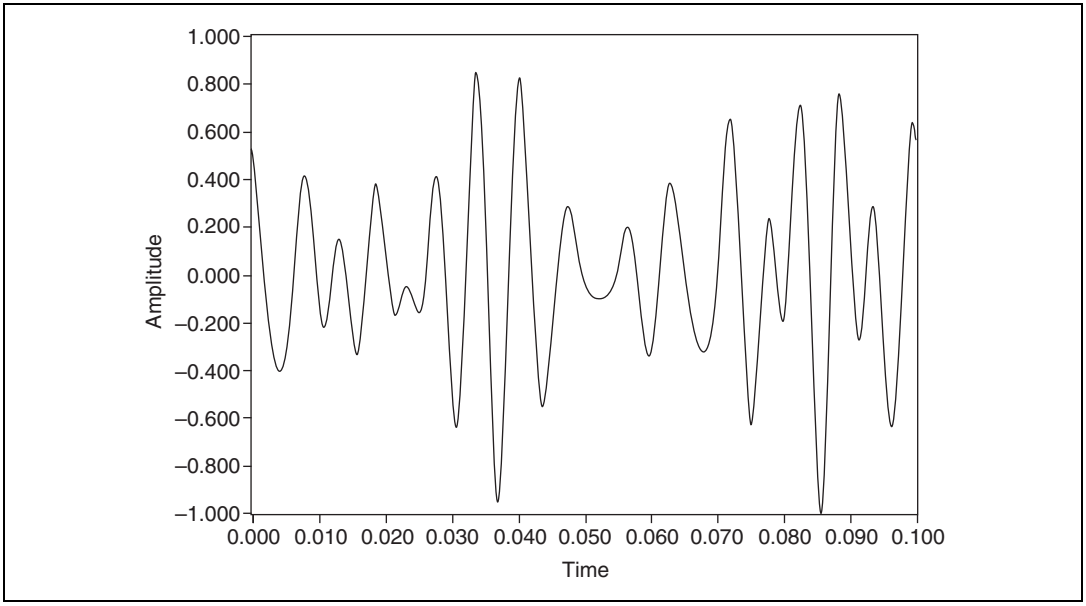


Figure 2-3. Multitone Signal with Linearly Varying Phase Difference between Adjacent Tones

The signal in Figure 2-3 resembles a chirp signal in that its frequency appears to decrease from left to right. The apparent decrease in frequency from left to right is characteristic of multitone signals generated by linearly varying the phase difference between adjacent frequency tones. Having a signal that is more noise-like than the signal in Figure 2-3 often is more desirable.

Varying the tone phases randomly results in a multitone signal whose amplitudes are nearly Gaussian in distribution as the number of tones increases. Figure 2-4 illustrates a signal created by varying the tone phases randomly.

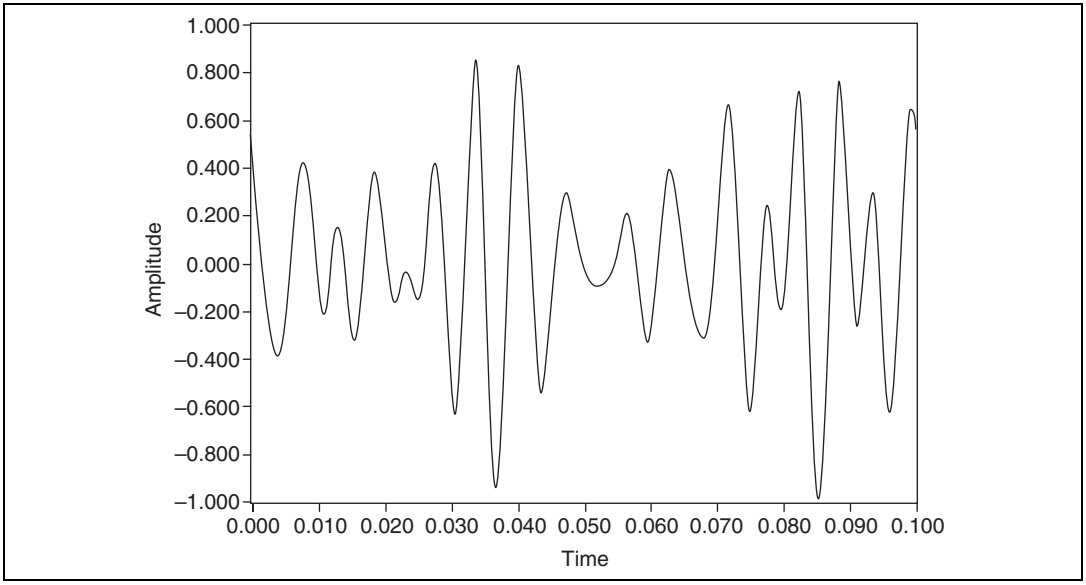


Figure 2-4. Multitone Signal with Random Phase Difference between Adjacent Tones

In addition to being more noise-like, the signal in Figure 2-4 also is much less sensitive to phase distortion. Multitone signals with the sort of phase relationship shown in Figure 2-4 generally achieve a crest factor between 10 dB and 11 dB.

Swept Sine versus Multitone

To characterize a system, you often must measure the response of the system at many different frequencies. You can use the following methods to measure the response of a system at many different frequencies:

- Swept sine continuously and smoothly changes the frequency of a sine wave across a range of frequencies.
- Stepped sine provides a single sine tone of fixed frequency as the stimulus for a certain time and then increments the frequency by a discrete amount. The process continues until all the frequencies of interest have been reached.
- Multitone provides a signal composed of multiple sine tones.

A multitone signal has significant advantages over the swept sine and stepped sine approaches. For a given range of frequencies, the multitone approach can be much faster than the equivalent swept sine measurement, due mainly to settling time issues. For each sine tone in a stepped sine

measurement, you must wait for the settling time of the system to end before starting the measurement.

The settling time issue for a swept sine can be even more complex. If the system has low-frequency poles and/or zeroes or high Q-resonances, the system might take a relatively long time to settle. For a multitone signal, you must wait only once for the settling time. A multitone signal containing one period of the lowest frequency—actually one period of the highest frequency resolution—is enough for the settling time. After the response to the multitone signal is acquired, the processing can be very fast. You can use a single fast Fourier transform (FFT) to measure many frequency points, amplitude and phase, simultaneously.

The swept sine approach is more appropriate than the multitone approach in certain situations. Each measured tone within a multitone signal is more sensitive to noise because the energy of each tone is lower than that in a single pure tone. For example, consider a single sine tone of amplitude 10 V peak and frequency 100 Hz. A multitone signal containing 10 tones, including the 100 Hz tone, might have a maximum amplitude of 10 V. However, the 100 Hz tone component has an amplitude somewhat less than 10 V. The lower amplitude of the 100 Hz tone component is due to the way that all the sine tones sum. Assuming the same level of noise, the signal-to-noise ratio (SNR) of the 100 Hz component is better for the case of the swept sine approach. In the multitone approach, you can mitigate the reduced SNR by adjusting the amplitudes and phases of the tones, applying higher energy where needed, and applying lower energy at less critical frequencies.

When viewing the response of a system to a multitone stimulus, any energy between FFT bins is due to noise or unit-under-test (UUT) induced distortion. The frequency resolution of the FFT is limited by your measurement time. If you want to measure your system at 1.000 kHz and 1.001 kHz, using two independent sine tones is the best approach. Using two independent sine tones, you can perform the measurement in a few milliseconds, while a multitone measurement requires at least one second. The difference in measurement speed is because you must wait long enough to obtain the required number of samples to achieve a frequency resolution of 1 Hz. Some applications, such as finding the resonant frequency of a crystal, combine a multitone measurement for coarse measurement and a narrow-range sweep for fine measurement.

Noise Generation

You can use noise signals to perform frequency response measurements or to simulate certain processes. Several types of noise are typically used, namely uniform white noise, Gaussian white noise, and periodic random noise.

The term white in the definition of noise refers to the frequency domain characteristic of noise. Ideal white noise has equal power per unit bandwidth, resulting in a flat power spectral density across the frequency range of interest. Thus, the power in the frequency range from 100 Hz to 110 Hz is the same as the power in the frequency range from 1,000 Hz to 1,010 Hz. In practical measurements, achieving the flat power spectral density requires an infinite number of samples. Thus, when making measurements of white noise, the power spectra are usually averaged, with more number of averages resulting in a flatter power spectrum.

The terms uniform and Gaussian refer to the probability density function (PDF) of the amplitudes of the time-domain samples of the noise. For uniform white noise, the PDF of the amplitudes of the time domain samples is uniform within the specified maximum and minimum levels. In other words, all amplitude values between some limits are equally likely or probable. Thermal noise produced in active components tends to be uniform white in distribution. Figure 2-5 shows the distribution of the samples of uniform white noise.

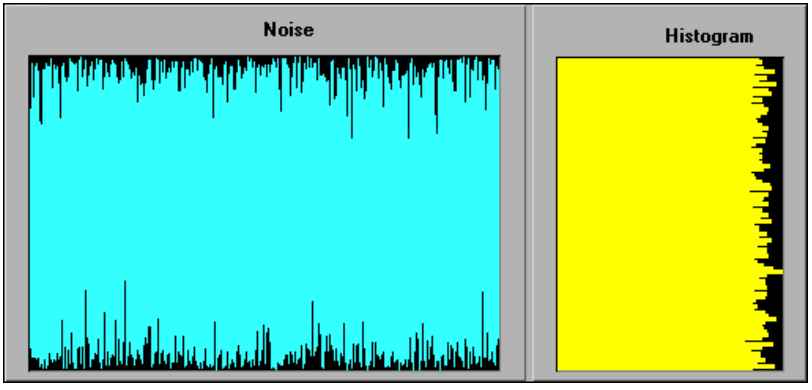


Figure 2-5. Uniform White Noise

For Gaussian white noise, the PDF of the amplitudes of the time domain samples is Gaussian. If uniform white noise is passed through a linear system, the resulting output is Gaussian white noise. Figure 2-6 shows the distribution of the samples of Gaussian white noise.

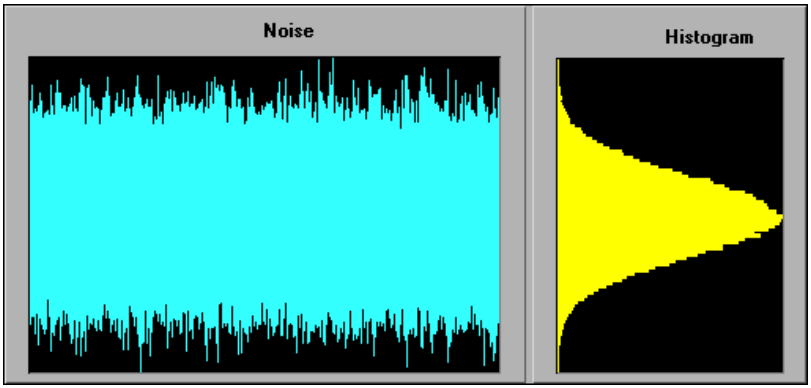


Figure 2-6. Gaussian White Noise

Periodic random noise (PRN) is a summation of sinusoidal signals with the same amplitudes but with random phases. PRN consists of all sine waves with frequencies that can be represented with an integral number of cycles in the requested number of samples. Because PRN contains only integral-cycle sinusoids, you do not need to window PRN before performing spectral analysis. PRN is self-windowing and therefore has no spectral leakage.

PRN does not have energy at all frequencies as white noise does but has energy only at discrete frequencies that correspond to harmonics of a fundamental frequency. The fundamental frequency is equal to the sampling frequency divided by the number of samples. However, the level of noise at each of the discrete frequencies is the same.

You can use PRN to compute the frequency response of a linear system with one time record instead of averaging the frequency response over several time records, as you must for nonperiodic random noise sources. Figure 2-7 shows the spectrum of PRN and the averaged spectra of white noise.

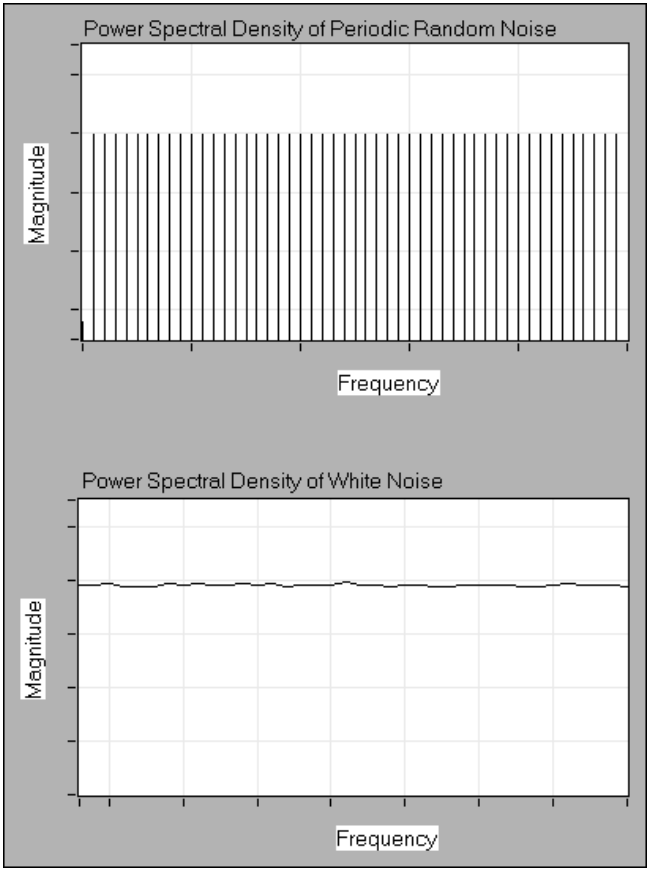


Figure 2-7. Spectral Representation of Periodic Random Noise and Averaged White Noise

Normalized Frequency

In the analog world, a signal frequency is measured in hertz (Hz), or cycles per second. But the digital system often uses a digital frequency, which is the ratio between the analog frequency and the sampling frequency, as shown by the following equation.

$$digital\ frequency = \frac{analog\ frequency}{sampling\ frequency}$$

The *digital frequency* is known as the normalized frequency and is measured in cycles per sample.

Some of the Signal Generation VIs use a frequency input **f** that is assumed to use normalized frequency units of cycles per sample. The normalized frequency ranges from 0.0 to 1.0, which corresponds to a real frequency range of 0 to the sampling frequency f_s. The normalized frequency also wraps around 1.0 so a normalized frequency of 1.1 is equivalent to 0.1. For example, a signal sampled at the Nyquist rate of $f_s/2$ means it is sampled twice per cycle, that is, two samples/cycle. This sampling rate corresponds to a normalized frequency of 1/2 cycles/sample = 0.5 cycles/sample. The reciprocal of the normalized frequency, $1/f$, gives you the number of times the signal is sampled in one cycle, that is, the number of samples per cycle.

When you use a VI that requires the normalized frequency as an input, you must convert your frequency units to the normalized units of cycles per sample. You must use normalized units of cycles per sample with the following Signal Generation VIs:

- Sine Wave
- Square Wave
- Sawtooth Wave
- Triangle Wave
- Arbitrary Wave
- Chirp Pattern

If you are used to working in frequency units of cycles, you can convert cycles to cycles per sample by dividing cycles by the number of samples generated.

You need only divide the frequency in cycles by the number of samples. For example, a frequency of two cycles is divided by 50 samples, resulting in a normalized frequency of $f = 1/25$ cycles/sample. This means that it takes 25, the reciprocal of f, samples to generate one cycle of the sine wave.

However, you might need to use frequency units of Hz, cycles per second. If you need to convert from Hz to cycles per sample, divide your frequency in Hz by the sampling rate given in samples per second, as shown in the following equation.

$$\frac{cycles\ per\ second}{samples\ per\ second} = \frac{cycles}{sample}$$

For example, you divide a frequency of 60 Hz by a sampling rate of 1,000 Hz to get the normalized frequency of $f = 0.06$ cycles/sample.

Therefore, it takes almost 17, or 1/0.06, samples to generate one cycle of the sine wave.

The Signal Generation VIs create many common signals required for network analysis and simulation. You also can use the Signal Generation VIs in conjunction with National Instruments hardware to generate analog output signals.

Wave and Pattern VIs

The names of most of the Signal Generation VIs contain the word *wave* or *pattern*. A basic difference exists between the operation of the two different types of VIs. The difference has to do with whether the VI can keep track of the phase of the signal it generates each time the VI is called.

Phase Control

The wave VIs have a **phase in** input that specifies the initial phase in degrees of the first sample of the generated waveform. The wave VIs also have a **phase out** output that indicates the phase of the next sample of the generated waveform. In addition, a **reset phase** input specifies whether the phase of the first sample generated when the wave VI is called is the phase specified in the **phase in** input or the phase available in the **phase out** output when the VI last executed. A TRUE value for **reset phase** sets the initial phase to **phase in**. A FALSE value for **reset phase** sets the initial phase to the value of **phase out** when the VI last executed.

All the wave VIs are reentrant, which means they can keep track of phase internally. The wave VIs accept frequency in normalized units of cycles per sample. The only pattern VI that uses normalized units is the Chirp Pattern VI. Wire FALSE to the **reset phase** input to allow for continuous sampling simulation.

3

Digital Filtering

This chapter introduces the concept of filtering, compares analog and digital filters, describes finite impulse response (FIR) and infinite impulse response (IIR) filters, and describes how to choose the appropriate digital filter for a particular application.

Introduction to Filtering

The filtering process alters the frequency content of a signal. For example, the bass control on a stereo system alters the low-frequency content of a signal, while the treble control alters the high-frequency content. Changing the bass and treble controls filters the audio signal. Two common filtering applications are removing noise and decimation. Decimation consists of lowpass filtering and reducing the sample rate.

The filtering process assumes that you can separate the signal content of interest from the raw signal. Classical linear filtering assumes that the signal content of interest is distinct from the remainder of the signal in the frequency domain.

Advantages of Digital Filtering Compared to Analog Filtering

An analog filter has an analog signal at both its input $x(t)$ and its output $y(t)$. Both $x(t)$ and $y(t)$ are functions of a continuous variable t and can have an infinite number of values. Analog filter design requires advanced mathematical knowledge and an understanding of the processes involved in the system affecting the filter.

Because of modern sampling and digital signal processing tools, you can replace analog filters with digital filters in applications that require flexibility and programmability, such as audio, telecommunications, geophysics, and medical monitoring applications.

Digital filters have the following advantages compared to analog filters:

- Digital filters are software programmable, which makes them easy to build and test.
- Digital filters require only the arithmetic operations of multiplication and addition/subtraction.
- Digital filters do not drift with temperature or humidity or require precision components.
- Digital filters have a superior performance-to-cost ratio.
- Digital filters do not suffer from manufacturing variations or aging.

Common Digital Filters

You can classify a digital filter as one of the following types:

- Finite impulse response (FIR) filter, also known as moving average (MA) filter
- Infinite impulse response (IIR) filter, also known as autoregressive moving-average (ARMA) filter
- Nonlinear filter

Traditional filter classification begins with classifying a filter according to its impulse response.

Impulse Response

An impulse is a short duration signal that goes from zero to a maximum value and back to zero again in a short time. Equation 3-1 provides the mathematical definition of an impulse.

$$x_0 = 1 \qquad (3\text{-}1)$$

$$x_i = 0 \qquad \text{for all } i \neq 0$$

The impulse response of a filter is the response of the filter to an impulse and depends on the values upon which the filter operates. Figure 3-1 illustrates impulse response.

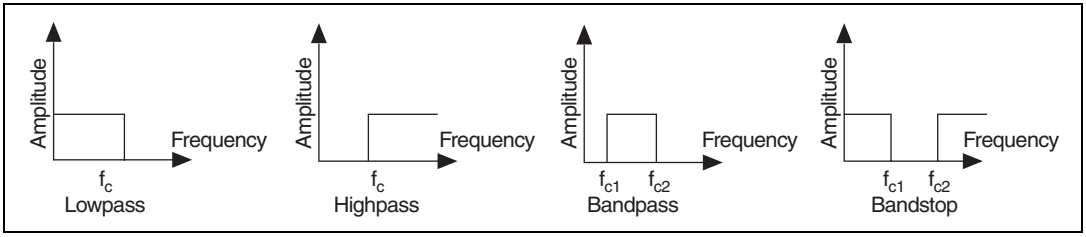


Figure 3-1. Impulse Response

The Fourier transform of the impulse response is the frequency response of the filter. The frequency response of a filter provides information about the output of the filter at different frequencies. In other words, the frequency response of a filter reflects the gain of the filter at different frequencies. For an ideal filter, the gain is one in the passband and zero in the stopband. An ideal filter passes all frequencies in the passband to the output unchanged but passes none of the frequencies in the stopband to the output.

Classifying Filters by Impulse Response

The impulse response of a filter determines whether the filter is an FIR or IIR filter. The output of an FIR filter depends only on the current and past input values. The output of an IIR filter depends on the current and past input values and the current and past output values.

The operation of a cash register can serve as an example to illustrate the difference between FIR and IIR filter operations. For this example, the following conditions are true:

- $x[k]$ is the cost of the current item entered into the cash register.
- $x[k-1]$ is the price of the past item entered into the cash register.
- $1 \le k \le N$
- N is the total number of items entered into the cash register.

The following statements describe the operation of the cash register:

- The cash register adds the cost of each item to produce the running total $y[k]$.
- The following equation computes $y[k]$ up to the k^{th} item.

$$y[k] = x[k] + x[k-1] + x[k-2] + x[k-3] + \ldots + x[1] \qquad (3\text{-}2)$$

Therefore, the total for N items is $y[N]$.

- $y[k]$ equals the total up to the k^{th} item.

  $y[k-1]$ equals the total up to the $(k-1)$ item.

  Therefore, Equation 3-2 can be rewritten as the following equation.

$$y[k] = y[k-1] + x[k] \quad (3\text{-}3)$$

- Add a tax of 8.25% and rewrite Equations 3-2 and 3-3 as the following equations.

$$y[k] = 1.0825x[k] + 1.0825x[k-1] + 1.0825x[k-2] + 1.0825x[k-3] + \ldots + 1.0825x[1] \quad (3\text{-}4)$$

$$y[k] = y[k-1] + 1.0825x[k] \quad (3\text{-}5)$$

Equations 3-4 and 3-5 identically describe the behavior of the cash register. However, Equation 3-4 describes the behavior of the cash register only in terms of the input, while Equation 3-5 describes the behavior in terms of both the input and the output. Equation 3-4 represents a nonrecursive, or FIR, operation. Equation 3-5 represents a recursive, or IIR, operation.

Equations that describe the operation of a filter and have the same form as Equations 3-2, 3-3, 3-4, and 3-5 are difference equations.

FIR filters are the simplest filters to design. If a single impulse is present at the input of an FIR filter and all subsequent inputs are zero, the output of an FIR filter becomes zero after a finite time. Therefore, FIR filters are finite. The time required for the filter output to reach zero equals the number of filter coefficients. Refer to the *FIR Filters* section of this chapter for more information about FIR filters.

Because IIR filters operate on current and past input values and current and past output values, the impulse response of an IIR filter never reaches zero and is an infinite response. Refer to the *IIR Filters* section of this chapter for more information about IIR filters.

Filter Coefficients

In Equation 3-4, the multiplying constant for each term is 1.0825. In Equation 3-5, the multiplying constants are 1 for $y[k-1]$ and 1.0825 for $x[k]$. The multiplying constants are the coefficients of the filter. For an IIR filter, the coefficients multiplying the inputs are the forward coefficients. The coefficients multiplying the outputs are the reverse coefficients.

Characteristics of an Ideal Filter

In practical applications, ideal filters are not realizable.

Ideal filters allow a specified frequency range of interest to pass through while attenuating a specified unwanted frequency range. The following filter classifications are based on the frequency range a filter passes or blocks:

- Lowpass filters pass low frequencies and attenuate high frequencies.
- Highpass filters pass high frequencies and attenuate low frequencies.
- Bandpass filters pass a certain band of frequencies.
- Bandstop filters attenuate a certain band of frequencies.

Figure 3-2 shows the ideal frequency response of each of the preceding filter types.

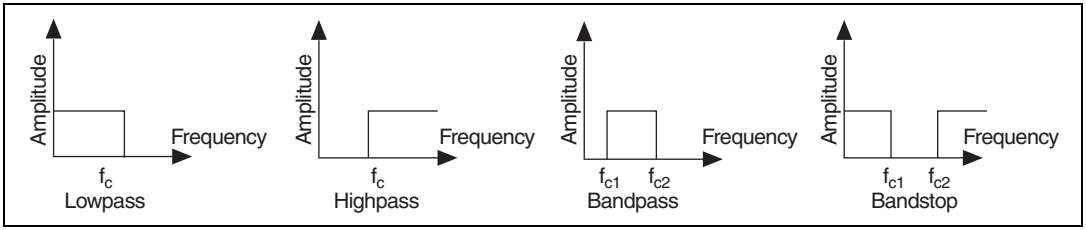


Figure 3-2. Ideal Frequency Response

In Figure 3-2, the filters exhibit the following behavior:

- The lowpass filter passes all frequencies below f_c.
- The highpass filter passes all frequencies above f_c.
- The bandpass filter passes all frequencies between f_{c1} and f_{c2}.
- The bandstop filter attenuates all frequencies between f_{c1} and f_{c2}.

The frequency points f_c, f_{c1}, and f_{c2} specify the cut-off frequencies for the different filters. When designing filters, you must specify the cut-off frequencies.

The passband of the filter is the frequency range that passes through the filter. An ideal filter has a gain of one (0 dB) in the passband so the amplitude of the signal neither increases nor decreases. The stopband of the filter is the range of frequencies that the filter attenuates. Figure 3-3 shows the passband (PB) and the stopband (SB) for each filter type.

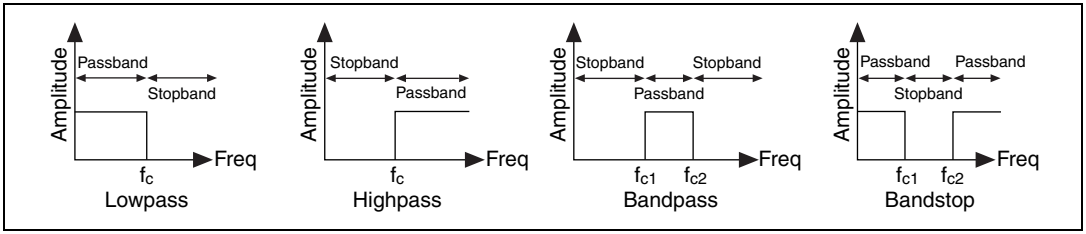


Figure 3-3. Passband and Stopband

The filters in Figure 3-3 have the following passband and stopband characteristics:

- The lowpass and highpass filters have one passband and one stopband.
- The bandpass filter has one passband and two stopbands.
- The bandstop filter has two passbands and one stopband.

Practical (Nonideal) Filters

Ideally, a filter has a unit gain (0 dB) in the passband and a gain of zero (–∞ dB) in the stopband. However, real filters cannot fulfill all the criteria of an ideal filter. In practice, a finite transition band always exists between the passband and the stopband. In the transition band, the gain of the filter changes gradually from one (0 dB) in the passband to zero (–∞ dB) in the stopband.

Transition Band

Figure 3-4 shows the passband, the stopband, and the transition band for each type of practical filter.

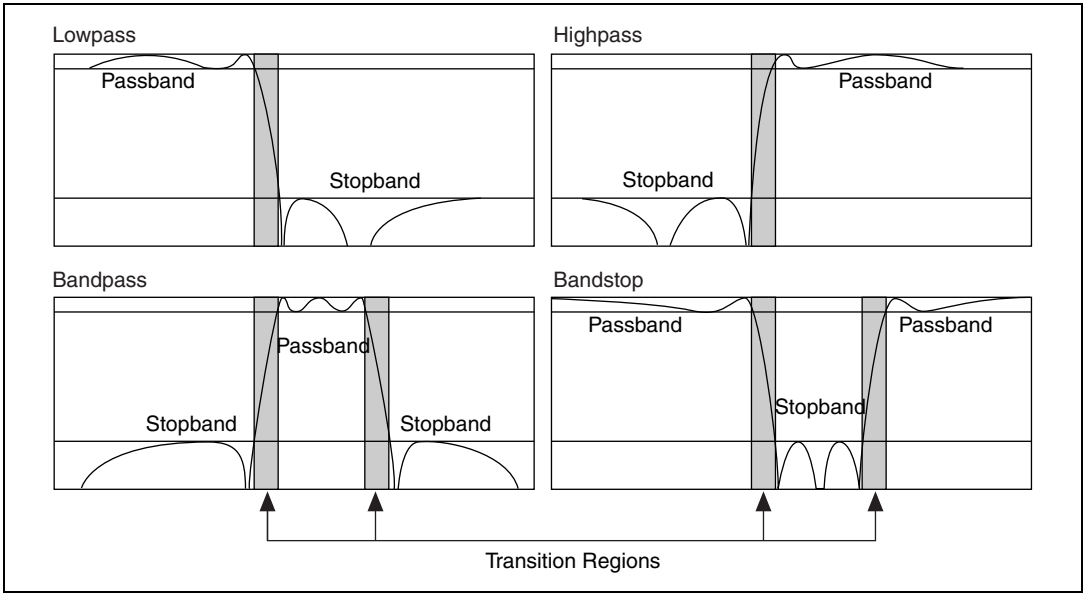


Figure 3-4. Nonideal Filters

In each plot in Figure 3-4, the *x*-axis represents frequency, and the *y*-axis represents the magnitude of the filter in dB. The passband is the region within which the gain of the filter varies from 0 dB to –3 dB.

Passband Ripple and Stopband Attenuation

In many applications, you can allow the gain in the passband to vary slightly from unity. This variation in the passband is the passband ripple, or the difference between the actual gain and the desired gain of unity. In practice, the stopband attenuation cannot be infinite, and you must specify a value with which you are satisfied. Measure both the passband ripple and the stopband attenuation in decibels (dB). Equation 3-6 defines a decibel.

$$\mathrm{dB} = 20 \log\left(\frac{A_o(f)}{A_i(f)}\right) \tag{3-6}$$

where log denotes the base 10 logarithm, $A_i(f)$ is the amplitude at a particular frequency f before filtering, and $A_o(f)$ is the amplitude at a particular frequency f after filtering.

When you know the passband ripple or stopband attenuation, you can use Equation 3-6 to determine the ratio of input and output amplitudes.

The ratio of the amplitudes shows how close the passband or stopband is to the ideal. For example, for a passband ripple of –0.02 dB, Equation 3-6 yields the following set of equations.

$$-0.02 = 20 \log\left(\frac{A_o(f)}{A_i(f)}\right) \tag{3-7}$$

$$\frac{A_o(f)}{A_i(f)} = 10^{-0.001} = 0.9977 \tag{3-8}$$

Equations 3-7 and 3-8 show that the ratio of input and output amplitudes is close to unity, which is the ideal for the passband.

Practical filter design attempts to approximate the ideal desired magnitude response, subject to certain constraints. Table 3-1 compares the characteristics of ideal filters and practical filters.

Table 3-1. Characteristics of Ideal and Practical Filters

Characteristic	Ideal Filters	Practical Filters
Passband	Flat and constant	Might contain ripples
Stopband	Flat and constant	Might contain ripples
Transition band	None	Have transition regions

Practical filter design involves compromise, allowing you to emphasize a desirable filter characteristic at the expense of a less desirable characteristic. The compromises you can make depend on whether the filter is an FIR or IIR filter and the design algorithm.

Sampling Rate

The sampling rate is important to the success of a filtering operation. The maximum frequency component of the signal of interest usually determines the sampling rate. In general, choose a sampling rate 10 times higher than the highest frequency component of the signal of interest.

Make exceptions to the previous sampling rate guideline when filter cut-off frequencies must be very close to either DC or the Nyquist frequency. Filters with cut-off frequencies close to DC or the Nyquist frequency might

have a slow rate of convergence. You can take the following actions to overcome the slow convergence:

- If the cut-off is too close to the Nyquist frequency, increase the sampling rate.
- If the cut-off is too close to DC, reduce the sampling rate.

In general, adjust the sampling rate only if you encounter problems.

FIR Filters

Finite impulse response (FIR) filters are digital filters that have a finite impulse response. FIR filters operate only on current and past input values and are the simplest filters to design. FIR filters also are known as nonrecursive filters, convolution filters, and moving average (MA) filters. FIR filters perform a convolution of the filter coefficients with a sequence of input values and produce an equally numbered sequence of output values. Equation 3-9 defines the finite convolution an FIR filter performs.

$$y_i = \sum_{k=0}^{n-1} h_k x_{i-k} \tag{3-9}$$

where x is the input sequence to filter, y is the filtered sequence, and h is the FIR filter coefficients.

FIR filters have the following characteristics:

- FIR filters can achieve linear phase because of filter coefficient symmetry in the realization.
- FIR filters are always stable.
- FIR filters allow you to filter signals using the convolution. Therefore, you generally can associate a delay with the output sequence, as shown in the following equation.

$$\text{delay} = \frac{n-1}{2}$$

where n is the number of FIR filter coefficients.

Figure 3-5 shows a typical magnitude and phase response of an FIR filter compared to normalized frequency.

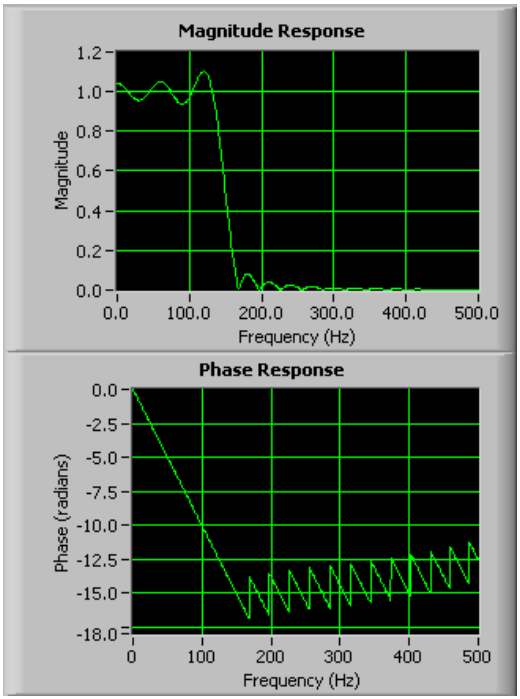


Figure 3-5. FIR Filter Magnitude and Phase Response Compared to Normalized Frequency

In Figure 3-5, the discontinuities in the phase response result from the discontinuities introduced when you use the absolute value to compute the magnitude response. The discontinuities in phase are on the order of pi. However, the phase is clearly linear.

Taps

The terms *tap* and *taps* frequently appear in descriptions of FIR filters, FIR filter design, and FIR filtering operations. Figure 3-6 illustrates the process of tapping.

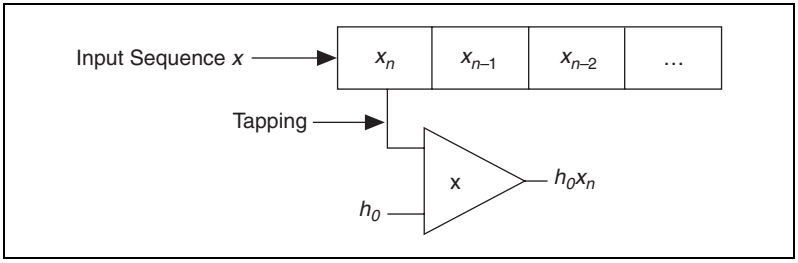


Figure 3-6. Tapping

Figure 3-6 represents an n-sample shift register containing the input samples $[x_i, x_{i-1}, \ldots]$. The term *tap* comes from the process of tapping off of the shift register to form each $h_k x_{i-k}$ term in Equation 3-9. *Taps* usually refers to the number of filter coefficients for an FIR filter.

Designing FIR Filters

You design FIR filters by approximating the desired frequency response of a discrete-time system. The most common techniques approximate the desired magnitude response while maintaining a linear-phase response. Linear-phase response implies that all frequencies in the system have the same propagation delay.

Figure 3-7 shows the block diagram of a VI that returns the frequency response of a bandpass equiripple FIR filter.

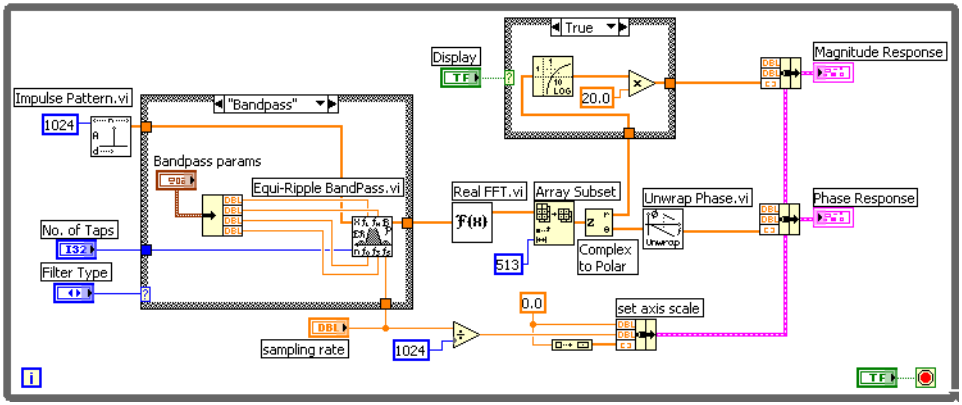


Figure 3-7. Frequency Response of a Bandpass Equiripple FIR Filter

The VI in Figure 3-7 completes the following steps to compute the frequency response of the filter.

1. Pass an impulse signal through the filter.
2. Pass the filtered signal out of the Case structure to the FFT VI. The Case structure specifies the filter type—lowpass, highpass, bandpass, or bandstop. The signal passed out of the Case structure is the impulse response of the filter.
3. Use the FFT VI to perform a Fourier transform on the impulse response and to compute the frequency response of the filter, such that the impulse response and the frequency response comprise the Fourier transform pair $h(t) \Leftrightarrow H(f)$. $h(t)$ is the impulse response. $H(f)$ is the frequency response.
4. Use the Array Subset function to reduce the data returned by the FFT VI. Half of the real FFT result is redundant so the VI needs to process only half of the data returned by the FFT VI.
5. Use the Complex To Polar function to obtain the magnitude-and-phase form of the data returned by the FFT VI. The magnitude-and-phase form of the complex output from the FFT VI is easier to interpret than the rectangular component of the FFT.
6. Unwrap the phase and convert it to degrees.
7. Convert the magnitude to decibels.

Figure 3-8 shows the magnitude and phase responses returned by the VI in Figure 3-7.

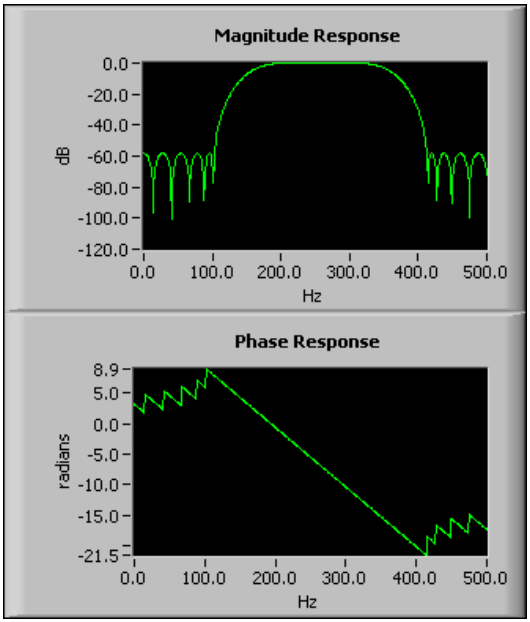


Figure 3-8. Magnitude and Phase Response of a Bandpass Equiripple FIR Filter

In Figure 3-8, the discontinuities in the phase response result from the discontinuities introduced when you use the absolute value to compute the magnitude response. However, the phase response is a linear response because all frequencies in the system have the same propagation delay.

Because FIR filters have ripple in the magnitude response, designing FIR filters has the following design challenges:

- Designing a filter with a magnitude response as close to the ideal as possible
- Designing a filter that distributes the ripple in a desired fashion

For example, a lowpass filter has an ideal characteristic magnitude response. A particular application might allow some ripple in the passband and more ripple in the stopband. The filter design algorithm must balance the relative ripple requirements while producing the sharpest transition band.

The most common techniques for designing FIR filters are windowing and the Parks-McClellan algorithm, also known as Remez Exchange.

Designing FIR Filters by Windowing

Windowing is the simplest technique for designing FIR filters because of its conceptual simplicity and ease of implementation. Designing FIR filters by windowing takes the inverse FFT of the desired magnitude response and applies a smoothing window to the result. The smoothing window is a time domain window.

Complete the following steps to design a FIR filter by windowing.

1. Decide on an ideal frequency response.
2. Calculate the impulse response of the ideal frequency response.
3. Truncate the impulse response to produce a finite number of coefficients. To meet the linear-phase constraint, maintain symmetry about the center point of the coefficients.
4. Apply a symmetric smoothing window.

Truncating the ideal impulse response results in the Gibbs phenomenon. The Gibbs phenomenon appears as oscillatory behavior near cut-off frequencies in the FIR filter frequency response. You can reduce the effects of the Gibbs phenomenon by using a smoothing window to smooth the truncation of the ideal impulse response. By tapering the FIR coefficients at each end, you can decrease the height of the side lobes in the frequency response. However, decreasing the side lobe heights causes the main lobe to widen, resulting in a wider transition band at the cut-off frequencies.

Selecting a smoothing window requires a trade-off between the height of the side lobes near the cut-off frequencies and the width of the transition band. Decreasing the height of the side lobes near the cut-off frequencies increases the width of the transition band. Decreasing the width of the transition band increases the height of the side lobes near the cut-off frequencies.

Designing FIR filters by windowing has the following disadvantages:

- Inefficiency
  - Windowing results in unequal distribution of ripple.
  - Windowing results in a wider transition band than other design techniques.

- Difficulty in specification
  - Windowing increases the difficulty of specifying a cut-off frequency that has a specific attenuation.
  - Filter designers must specify the ideal cut-off frequency.
  - Filter designers must specify the sampling frequency.
  - Filter designers must specify the number of taps.
  - Filter designers must specify the window type.

Designing FIR filters by windowing does not require a large amount of computational resources. Therefore, windowing is the fastest technique for designing FIR filters. However, windowing is not necessarily the best technique for designing FIR filters.

Designing Optimum FIR Filters Using the Parks-McClellan Algorithm

The Parks-McClellan algorithm, or Remez Exchange, uses an iterative technique based on an error criterion to design FIR filter coefficients. You can use the Parks-McClellan algorithm to design optimum, linear-phase, FIR filter coefficients. Filters you design with the Parks-McClellan algorithm are optimal because they minimize the maximum error between the actual magnitude response of the filter and the ideal magnitude response of the filter.

Designing optimum FIR filters reduces adverse effects at the cut-off frequencies. Designing optimum FIR filters also offers more control over the approximation errors in different frequency bands than other FIR filter design techniques, such as designing FIR filters by windowing, which provides no control over the approximation errors in different frequency bands.

Optimum FIR filters you design using the Parks-McClellan algorithm have the following characteristics:

- A magnitude response with the weighted ripple evenly distributed over the passband and stopband
- A sharp transition band

FIR filters you design using the Parks-McClellan algorithm have an optimal response. However, the design process is complex, requires a large amount of computational resources, and is much longer than designing FIR filters by windowing.

Designing Equiripple FIR Filters Using the Parks-McClellan Algorithm

You can use the Parks-McClellan algorithm to design equiripple FIR filters. Equiripple design equally weights the passband and stopband ripple and produces filters with a linear phase characteristic.

You must specify the following filter characteristics to design an equiripple FIR filter:

- Cut-off frequency
- Number of taps
- Filter type, such as lowpass, highpass, bandpass, or bandstop
- Pass frequency
- Stop frequency

The cut-off frequency for equiripple filters specifies the edge of the passband, the stopband, or both. The ripple in the passband and stopband of equiripple filters causes the following magnitude responses:

- **Passband**—a magnitude response greater than or equal to 1
- **Stopband**—a magnitude response less than or equal to the stopband attenuation

For example, if you specify a lowpass filter, the passband cut-off frequency is the highest frequency for which the passband conditions are true. Similarly, the stopband cut-off frequency is the lowest frequency for which the stopband conditions are true.

Designing Narrowband FIR Filters

Using conventional techniques to design FIR filters with especially narrow bandwidths can result in long filter lengths. FIR filters with long filter lengths often require long design and implementation times and are susceptible to numerical inaccuracy. In some cases, conventional filter design techniques, such as the Parks-McClellan algorithm, might not produce an acceptable narrow bandwidth FIR filter.

The interpolated finite impulse response (IFIR) filter design technique offers an efficient algorithm for designing narrowband FIR filters. Using the IFIR technique produces narrowband filters that require fewer coefficients and computations than filters you design by directly applying the Parks-McClellan algorithm. The FIR Narrowband Coefficients VI uses the IFIR technique to generate narrowband FIR filter coefficients.

You must specify the following parameters when developing narrowband filter specifications:

- Filter type, such as lowpass, highpass, bandpass, or bandstop
- Passband ripple on a linear scale
- Sampling frequency
- Passband frequency, which refers to passband width for bandpass and bandstop filters
- Stopband frequency, which refers to stopband width for bandpass and bandstop filters
- Center frequency for bandpass and bandstop filters
- Stopband attenuation in decibels

Figure 3-9 shows the block diagram of a VI that estimates the frequency response of a narrowband FIR bandpass filter by transforming the impulse response into the frequency domain.

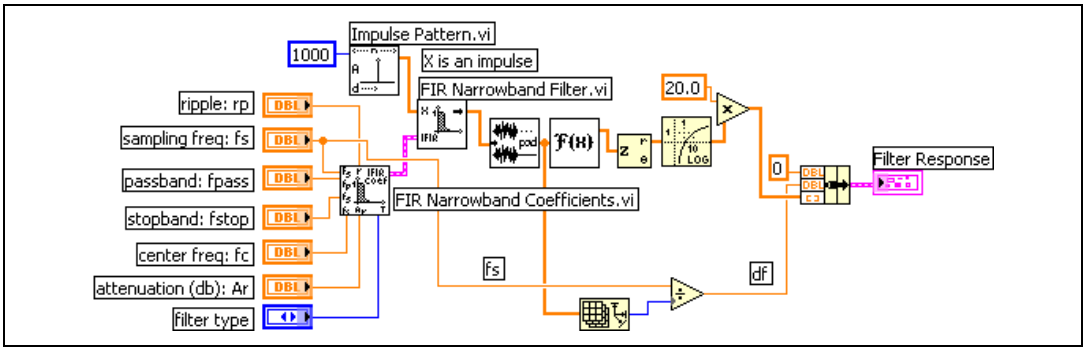


Figure 3-9. Estimating the Frequency Response of a Narrowband FIR Bandpass Filter

Figure 3-10 shows the filter response from zero to the Nyquist frequency that the VI in Figure 3-9 returns.

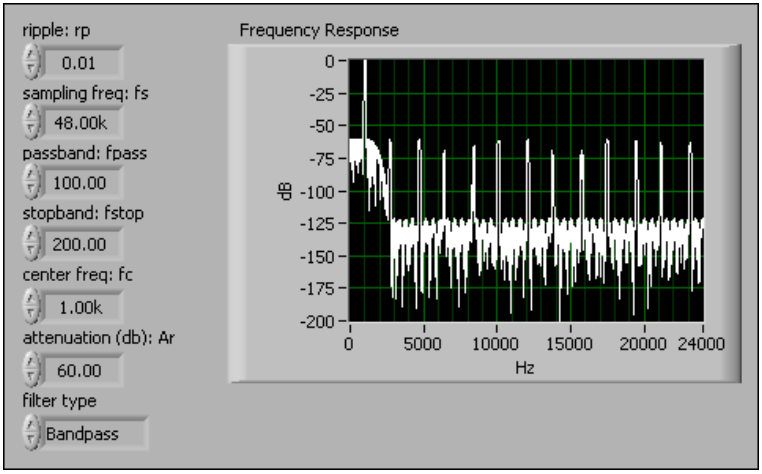


Figure 3-10. Estimated Frequency Response of a Narrowband FIR Bandpass Filter from Zero to Nyquist

In Figure 3-10, the narrow passband centers around 1 kHz. The narrow passband center at 1 kHz is the response of the filter specified by the front panel controls in Figure 3-10.

Figure 3-11 shows the filter response in detail.

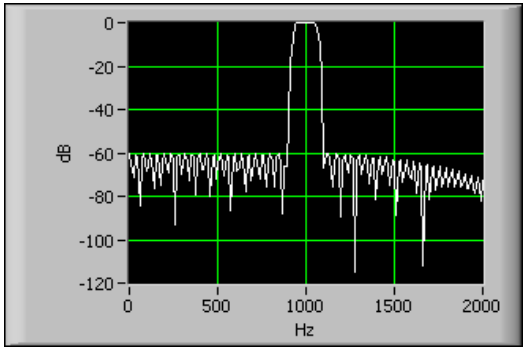


Figure 3-11. Detail of the Estimated Frequency Response of a Narrowband FIR Bandpass Filter

In Figure 3-11, the narrow passband clearly centers around 1 kHz and the attenuation of the signal at 60 dB below the passband.

Refer to the works of Vaidyanathan, P. P. and Neuvo, Y. et al. in Appendix A, *References*, for more information about designing IFIR filters.

Designing Wideband FIR Filters

You also can use the IFIR technique to produce wideband FIR lowpass filters and wideband FIR highpass filters. A wideband FIR lowpass filter has a cut-off frequency near the Nyquist frequency. A wideband FIR highpass filter has a cut-off frequency near zero. You can use the FIR Narrowband Coefficients VI to design wideband FIR lowpass filters and wideband FIR highpass filters. Figure 3-12 shows the frequency response that the VI in Figure 3-9 returns when you use it to estimate the frequency response of a wideband FIR lowpass filter.

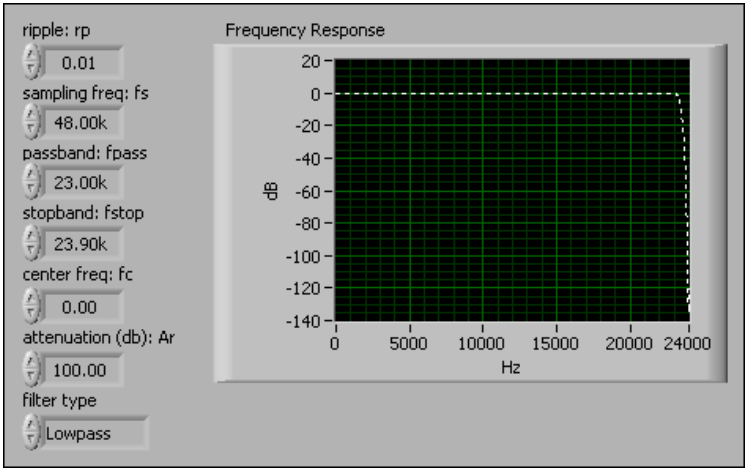


Figure 3-12. Frequency Response of a Wideband FIR Lowpass Filter from Zero to Nyquist

In Figure 3-12, the front panel controls define a narrow bandwidth between the stopband at 23.9 kHz and the Nyquist frequency at 24 kHz. However, the frequency response of the filter runs from zero to 23.9 kHz, which makes the filter a wideband filter.

IIR Filters

Infinite impulse response (IIR) filters, also known as recursive filters and autoregressive moving-average (ARMA) filters, operate on current and past input values and current and past output values. The impulse response

of an IIR filter is the response of the general IIR filter to an impulse, as Equation 3-1 defines impulse. Theoretically, the impulse response of an IIR filter never reaches zero and is an infinite response.

The following general difference equation characterizes IIR filters.

$$y_i = \frac{1}{a_0}\left(\sum_{j=0}^{N_b-1} b_j x_{i-j} - \sum_{k=1}^{N_a-1} a_k y_{i-k} \right) \quad (3\text{-}10)$$

where b_j is the set of forward coefficients, N_b is the number of forward coefficients, a_k is the set of reverse coefficients, and N_a is the number of reverse coefficients.

Equation 3-10 describes a filter with an impulse response of theoretically infinite length for nonzero coefficients. However, in practical filter applications, the impulse response of a stable IIR filter decays to near zero in a finite number of samples.

In most IIR filter designs and all of the LabVIEW IIR filters, coefficient a_0 is 1. The output sample at the current sample index i is the sum of scaled current and past inputs and scaled past outputs, as shown by Equation 3-11.

$$y_i = \sum_{j=0}^{N_b-1} b_j x_{i-j} - \sum_{k=1}^{N_a-1} a_k y_{i-k}, \quad (3\text{-}11)$$

where x_i is the current input, x_{i-j} is the past inputs, and y_{i-k} is the past outputs.

IIR filters might have ripple in the passband, the stopband, or both. IIR filters have a nonlinear-phase response.

Cascade Form IIR Filtering

Equation 3-12 defines the direct-form transfer function of an IIR filter.

$$H(z) = \frac{b_0 + b_1 z^{-1} + \ldots + b_{N_b-1} z^{-(N_b-1)}}{1 + a_1 z^{-1} + \ldots + a_{N_a-1} z^{-(N_a-1)}} \quad (3\text{-}12)$$

A filter implemented by directly using the structure defined by Equation 3-12 after converting it to the difference equation in

Equation 3-11 is a direct-form IIR filter. A direct-form IIR filter often is sensitive to errors introduced by coefficient quantization and by computational precision limits. Also, a filter with an initially stable design can become unstable with increasing coefficient length. The filter order is proportional to the coefficient length. As the coefficient length increases, the filter order increases. As filter order increases, the filter becomes more unstable.

You can lessen the sensitivity of a filter to error by writing Equation 3-12 as a ratio of z transforms, which divides the direct-form transfer function into lower order sections, or filter stages.

By factoring Equation 3-12 into second-order sections, the transfer function of the filter becomes a product of second-order filter functions, as shown in Equation 3-13.

$$H(z) = \prod_{k=1}^{N_s} \frac{b_{0k} + b_{1k} z^{-1} + b_{2k} z^{-2}}{1 + a_{1k} z^{-1} + a_{2k} z^{-2}} \tag{3-13}$$

where N_s is the number of stages, $N_s = \left\lfloor \frac{N_a}{2} \right\rfloor$ is the largest integer $\leq \frac{N_a}{2}$, and $N_a \geq N_b$.

You can describe the filter structure defined by Equation 3-13 as a cascade of second-order filters. Figure 3-13 illustrates cascade filtering.

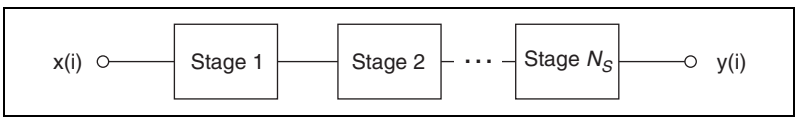


Figure 3-13. Stages of Cascade Filtering

You implement each individual filter stage in Figure 3-13 with the direct-form II filter structure. You use the direct-form II filter structure to implement each filter stage for the following reasons:

- The direct-form II filter structure requires a minimum number of arithmetic operations.
- The direct-form II filter structure requires a minimum number of delay elements, or internal filter states.
- Each k^{th} stage has one input, one output, and two past internal states, $s_k[i-1]$ and $s_k[i-2]$.

If n is the number of samples in the input sequence, the filtering operation proceeds as shown in the following equations.

$$y_0[i] = x[i]$$

$$s_k[i] = y_{k-1}[i-1] - a_{1k}s_k[i-1] - a_{2k}s_k[i-2] \qquad k = 1, 2, \ldots, N_s$$

$$y_k[i] = b_{0k}s_k[i] + b_{1k}s_k[i-1] + b_{2k}s_k[i-2] \qquad k = 1, 2, \ldots, N_s$$

for each sample $i = 0, 1, 2, \ldots, n-1$.

Second-Order Filtering

For lowpass and highpass filters, which have a single cut-off frequency, you can design second-order filter stages directly. The resulting IIR lowpass or highpass filter contains cascaded second-order filters.

Each second-order filter stage has the following characteristics:

- $k = 1, 2, \ldots, N_s$, where k is the second-order filter stage number and N_s is the total number of second-order filter stages.
- Each second-order filter stage has two reverse coefficients, (a_{1k}, a_{2k}).
- The total number of reverse coefficients equals $2N_s$.
- Each second-order filter stage has three forward coefficients, (b_{0k}, b_{1k}, b_{2k}).
- The total number of forward coefficients equals $3N_s$.

In Signal Processing VIs with **Reverse Coefficients** and **Forward Coefficients** parameters, the **Reverse Coefficients** and the **Forward Coefficients** arrays contain the coefficients for one second-order filter stage, followed by the coefficients for the next second-order filter stage, and so on. For example, an IIR filter with two second-order filter stages must have a total of four reverse coefficients and six forward coefficients, as shown in the following equations.

$$\text{Total number of reverse coefficients} = 2N_s = 2 \times 2 = 4$$
$$\text{Reverse Coefficients} = \{a_{11}, a_{21}, a_{12}, a_{22}\}$$

$$\text{Total number of forward coefficients} = 3N_s = 3 \times 2 = 6$$
$$\text{Forward Coefficients} = \{b_{01}, b_{11}, b_{21}, b_{02}, b_{12}, b_{22}\}$$

Fourth-Order Filtering

For bandpass and bandstop filters, which have two cut-off frequencies, fourth-order filter stages are a more direct form of filter design than second-order filter stages. IIR bandpass or bandstop filters resulting from fourth-order filter design contain cascaded fourth-order filters.

Each fourth-order filter stage has the following characteristics:

- $k = 1, 2, \ldots, N_s$, where k is the fourth-order filter stage number and N_s is the total number of fourth-order filter stages.
- $N_s = \left\lfloor \frac{N_a + 1}{4} \right\rfloor$.
- Each fourth-order filter stage has four reverse coefficients, $(a_{1k}, a_{2k}, a_{3k}, a_{4k})$.
- The total number of reverse coefficients equals $4N_s$.
- Each fourth-order filter stage has five forward coefficients, $(b_{0k}, b_{1k}, b_{2k}, b_{3k}, b_{4k})$.
- The total number of forward coefficients equals $5N_s$.

You implement cascade stages in fourth-order filtering in the same manner as in second-order filtering. The following equations show how the filtering operation for fourth-order stages proceeds.

$$y_0[i] = x[i]$$

$$s_k[i] = y_{k-1}[i-1] - a_{1k}s_k[i-1] - a_{2k}s_k[i-2] - a_{3k}s_k[i-3] - a_{4k}s_k[i-4]$$

$$y_k[i] = b_{0k}s_k[i] + b_{1k}s_k[i-1] + b_{2k}s_k[i-2] - b_{3k}s_k[i-3] - b_{4k}s_k[i-4]$$

where $k = 1, 2, \ldots, N_s$.

IIR Filter Types

Digital IIR filter designs come from the classical analog designs and include the following filter types:

- Butterworth filters
- Chebyshev filters
- Chebyshev II filters, also known as inverse Chebyshev and Type II Chebyshev filters
- Elliptic filters, also known as Cauer filters
- Bessel filters

The IIR filter designs differ in the sharpness of the transition between the passband and the stopband and where they exhibit their various characteristics—in the passband or the stopband.

Minimizing Peak Error

The Chebyshev, Chebyshev II, and elliptic filters minimize peak error by accounting for the maximum tolerable error in their frequency response. The maximum tolerable error is the maximum absolute value of the difference between the ideal filter frequency response and the actual filter frequency response. The amount of ripple, in dB, allowed in the frequency response of the filter determines the maximum tolerable error. Depending on the type, the filter minimizes peak error in the passband, stopband, or both.

Butterworth Filters

Butterworth filters have the following characteristics:

- Smooth response at all frequencies
- Monotonic decrease from the specified cut-off frequencies
- Maximal flatness, with the ideal response of unity in the passband and zero in the stopband
- Half-power frequency, or 3 dB down frequency, that corresponds to the specified cut-off frequencies

The advantage of Butterworth filters is their smooth, monotonically decreasing frequency response. Figure 3-14 shows the frequency response of a lowpass Butterworth filter.

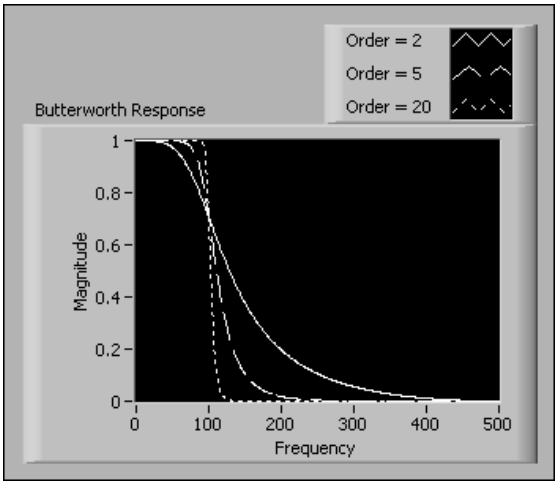


Figure 3-14. Frequency Response of a Lowpass Butterworth Filter

As shown in Figure 3-14, after you specify the cut-off frequency of a Butterworth filter, LabVIEW sets the steepness of the transition proportional to the filter order. Higher order Butterworth filters approach the ideal lowpass filter response.

Butterworth filters do not always provide a good approximation of the ideal filter response because of the slow rolloff between the passband and the stopband.

Chebyshev Filters

Chebyshev filters have the following characteristics:

- Minimization of peak error in the passband
- Equiripple magnitude response in the passband
- Monotonically decreasing magnitude response in the stopband
- Sharper rolloff than Butterworth filters

Compared to a Butterworth filter, a Chebyshev filter can achieve a sharper transition between the passband and the stopband with a lower order filter. The sharp transition between the passband and the stopband of a Chebyshev filter produces smaller absolute errors and faster execution speeds than a Butterworth filter.

Figure 3-15 shows the frequency response of a lowpass Chebyshev filter.

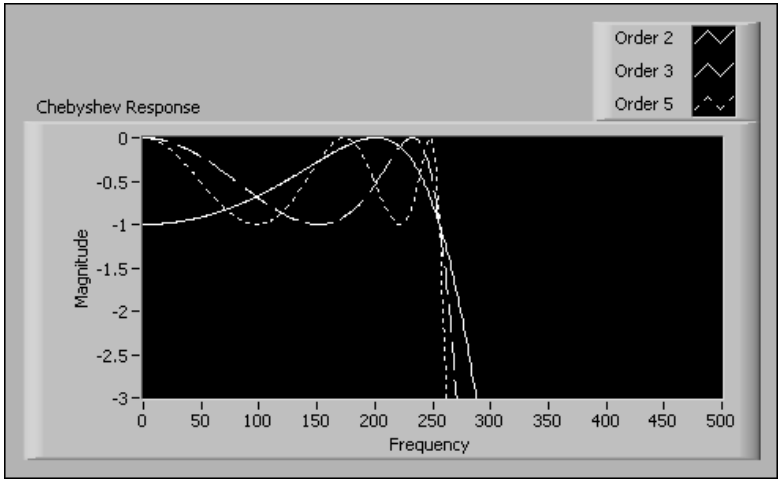


Figure 3-15. Frequency Response of a Lowpass Chebyshev Filter

In Figure 3-15, the maximum tolerable error constrains the equiripple response in the passband. Also, the sharp rolloff appears in the stopband.

Chebyshev II Filters

Chebyshev II filters have the following characteristics:

- Minimization of peak error in the stopband
- Equiripple magnitude response in the stopband
- Monotonically decreasing magnitude response in the passband
- Sharper rolloff than Butterworth filters

Chebyshev II filters are similar to Chebyshev filters. However, Chebyshev II filters differ from Chebyshev filters in the following ways:

- Chebyshev II filters minimize peak error in the stopband instead of the passband. Minimizing peak error in the stopband instead of the passband is an advantage of Chebyshev II filters over Chebyshev filters.
- Chebyshev II filters have an equiripple magnitude response in the stopband instead of the passband.
- Chebyshev II filters have a monotonically decreasing magnitude response in the passband instead of the stopband.

Figure 3-16 shows the frequency response of a lowpass Chebyshev II filter.

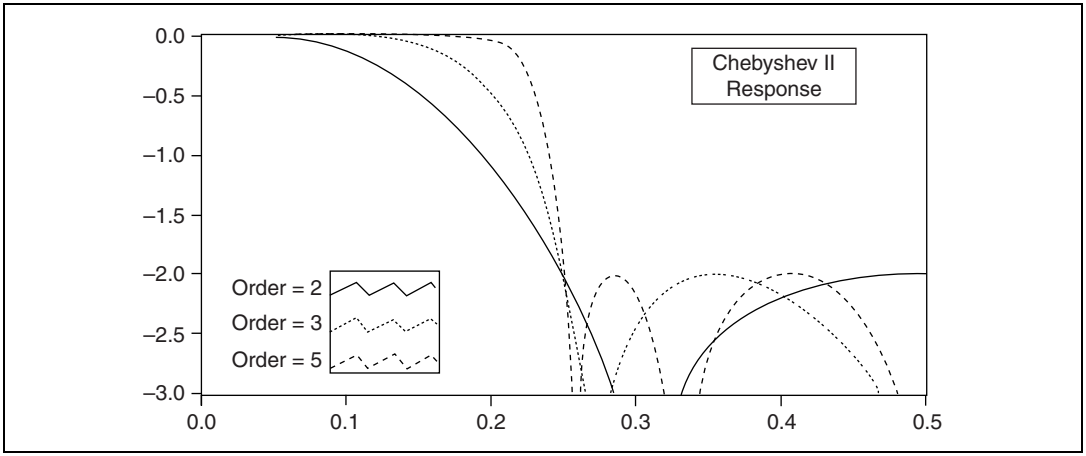


Figure 3-16. Frequency Response of a Lowpass Chebyshev II Filter

In Figure 3-16, the maximum tolerable error constrains the equiripple response in the stopband. Also, the smooth monotonic rolloff appears in the stopband.

Chebyshev II filters have the same advantage over Butterworth filters that Chebyshev filters have—a sharper transition between the passband and the stopband with a lower order filter, resulting in a smaller absolute error and faster execution speed.

Elliptic Filters

Elliptic filters have the following characteristics:

- Minimization of peak error in the passband and the stopband
- Equiripples in the passband and the stopband

Compared with the same order Butterworth or Chebyshev filters, the elliptic filters provide the sharpest transition between the passband and the stopband, which accounts for their widespread use.

Figure 3-17 shows the frequency response of a lowpass elliptic filter.

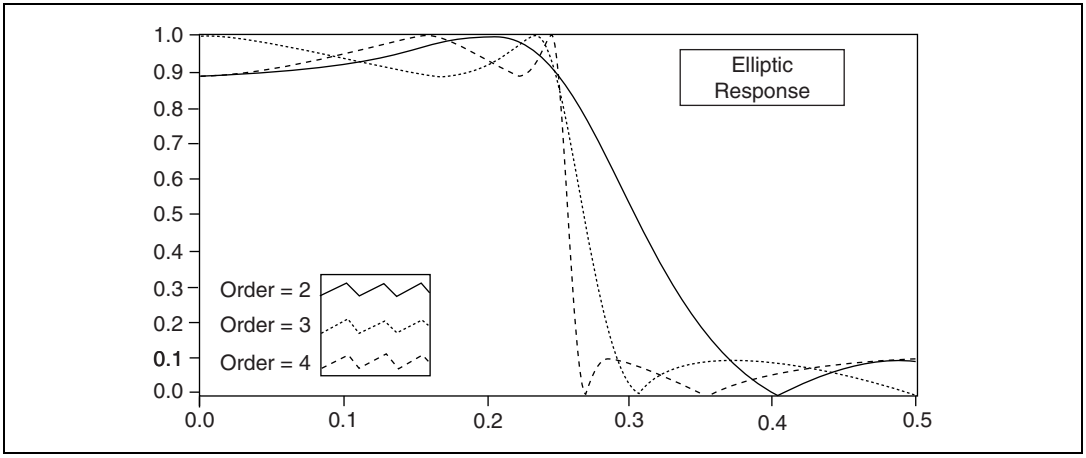


Figure 3-17. Frequency Response of a Lowpass Elliptic Filter

In Figure 3-17, the same maximum tolerable error constrains the ripple in both the passband and the stopband. Also, even low-order elliptic filters have a sharp transition edge.

Bessel Filters

Bessel filters have the following characteristics:

- Maximally flat response in both magnitude and phase
- Nearly linear-phase response in the passband

You can use Bessel filters to reduce nonlinear-phase distortion inherent in all IIR filters. High-order IIR filters and IIR filters with a steep rolloff have a pronounced nonlinear-phase distortion, especially in the transition regions of the filters. You also can obtain linear-phase response with FIR filters.

Figure 3-18 shows the magnitude and phase responses of a lowpass Bessel filter.

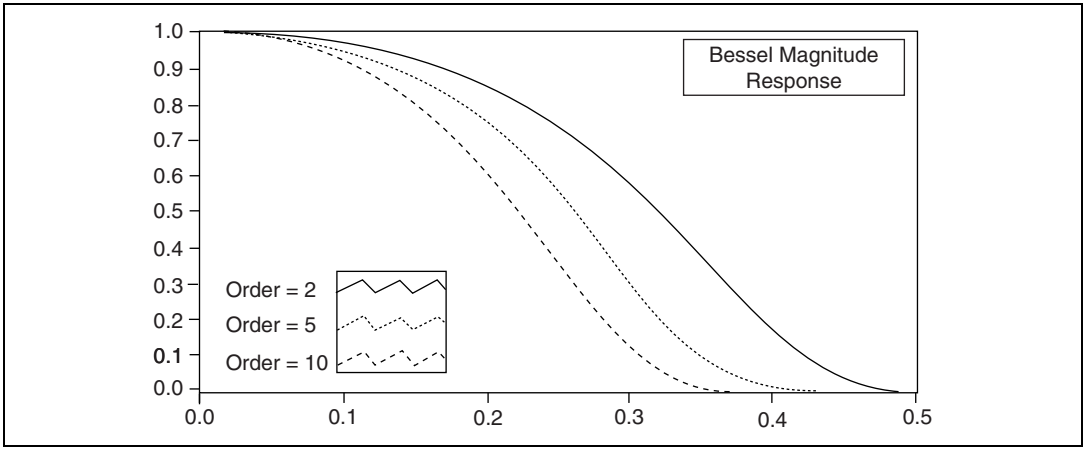


Figure 3-18. Magnitude Response of a Lowpass Bessel Filter

In Figure 3-18, the magnitude is smooth and monotonically decreasing at all frequencies.

Figure 3-19 shows the phase response of a lowpass Bessel filter.

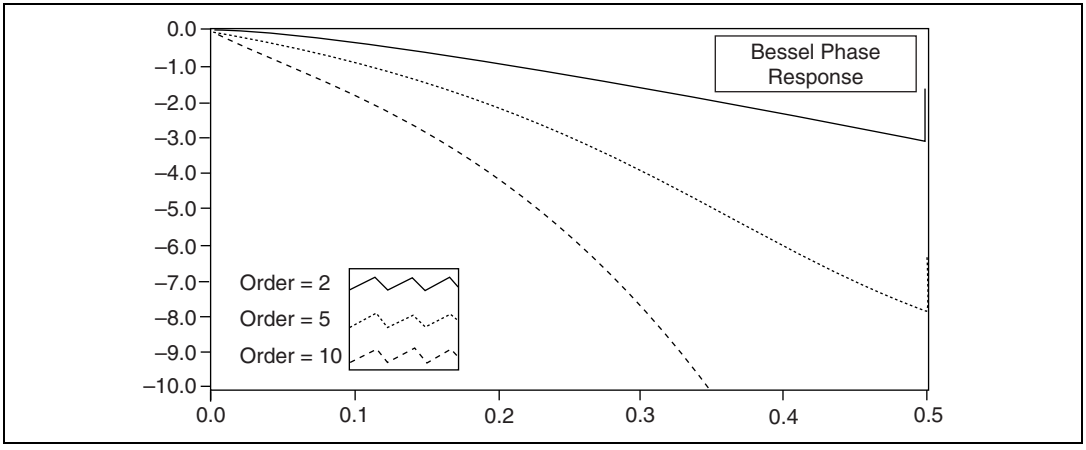


Figure 3-19. Phase Response of a Lowpass Bessel Filter

Figure 3-19 shows the nearly linear phase in the passband. Also, the phase monotonically decreases at all frequencies.

Like Butterworth filters, Bessel filters require high-order filters to minimize peak error, which accounts for their limited use.

Designing IIR Filters

When choosing an IIR filter for an application, you must know the response of the filter. Figure 3-20 shows the block diagram of a VI that returns the frequency response of an IIR filter.

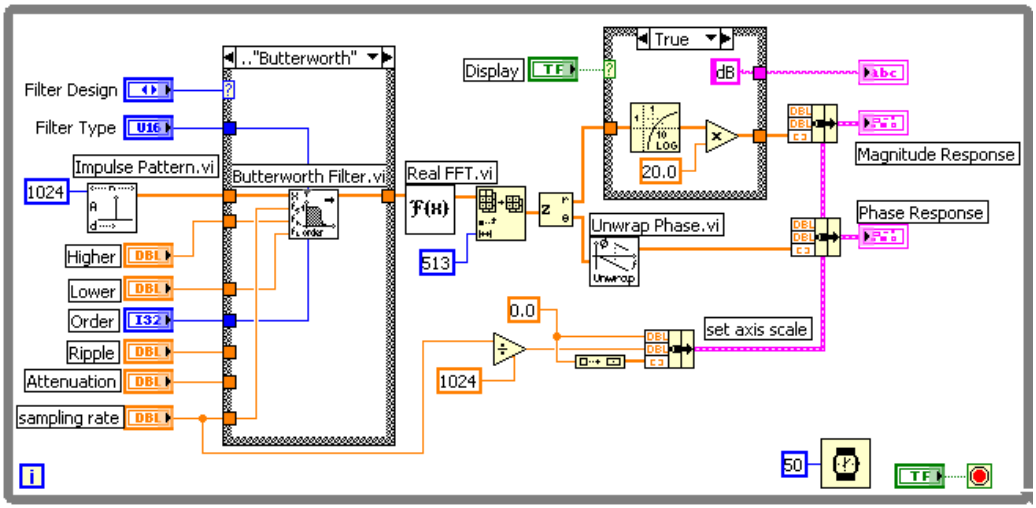


Figure 3-20. Frequency Response of an IIR Filter

Because the same mathematical theory applies to designing IIR and FIR filters, the block diagram in Figure 3-20 of a VI that returns the frequency response of an IIR filter and the block diagram in Figure 3-7 of a VI that returns the frequency response of an FIR filter share common design elements. The main difference between the two VIs is that the Case structure on the left side of Figure 3-20 specifies the IIR filter design and filter type instead of specifying only the filter type. The VI in Figure 3-20 computes the frequency response of an IIR filter by following the same steps outlined in the *Designing FIR Filters* section of this chapter.

Figure 3-21 shows the magnitude and the phase responses of a bandpass elliptic IIR filter.

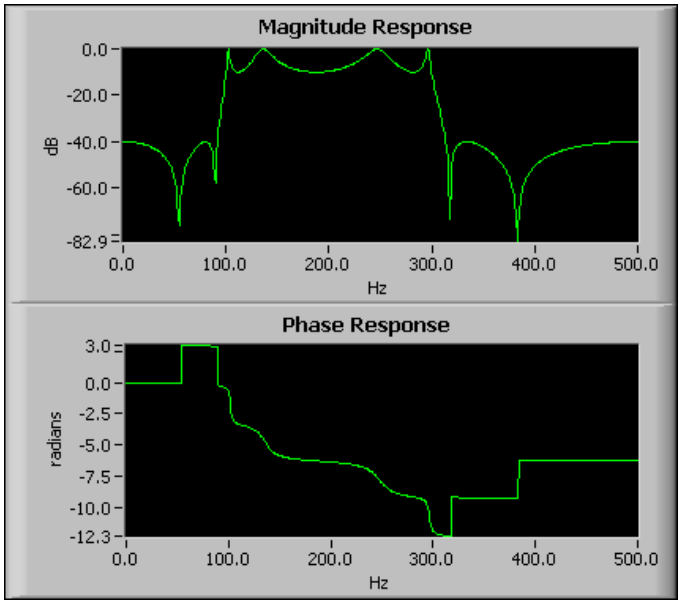


Figure 3-21. Magnitude and Phase Responses of a Bandpass Elliptic IIR Filter

In Figure 3-21, the phase information is clearly nonlinear. When deciding whether to use an IIR or FIR filter to process data, remember that IIR filters provide nonlinear phase information. Refer to the *Comparing FIR and IIR Filters* section and the *Selecting a Digital Filter Design* section of this chapter for information about differences between FIR and IIR filters and selecting an appropriate filter design.

IIR Filter Characteristics in LabVIEW

IIR filters in LabVIEW have the following characteristics:

- IIR filter VIs interpret values at negative indexes in Equation 3-10 as zero the first time you call the VI.
- A transient response, or delay, proportional to the filter order occurs before the filter reaches a steady state. Refer to the *Transient Response* section of this chapter for information about the transient response.
- The number of elements in the filtered sequence equals the number of elements in the input sequence.
- The filter retains the internal filter state values when the filtering process finishes.

Transient Response

The transient response occurs because the initial filter state is zero or has values at negative indexes. The duration of the transient response depends on the filter type.

The duration of the transient response for lowpass and highpass filters equals the filter order.

$$\text{delay} = \text{order}$$

The duration of the transient response for bandpass and bandstop filters equals twice the filter order.

$$\text{delay} = 2 \times \text{order}$$

You can eliminate the transient response on successive calls to an IIR filter VI by enabling state memory. To enable state memory for continuous filtering, wire a value of TRUE to the **init/cont** input of the IIR filter VI.

Figure 3-22 shows the transient response and the steady state for an IIR filter.

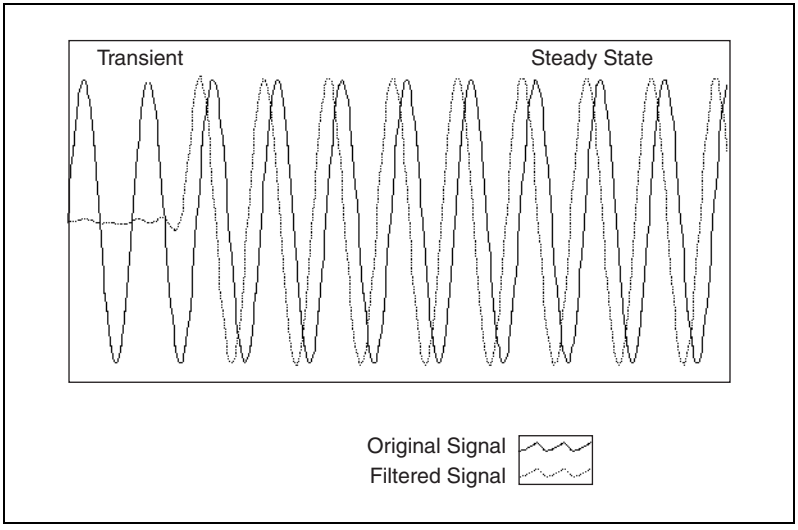


Figure 3-22. Transient Response and Steady State for an IIR Filter

Comparing FIR and IIR Filters

Because designing digital filters involves making compromises to emphasize a desirable filter characteristic over a less desirable characteristic, comparing FIR and IIR filters can help guide you in selecting the appropriate filter design for a particular application.

IIR filters can achieve the same level of attenuation as FIR filters but with far fewer coefficients. Therefore, an IIR filter can provide a significantly faster and more efficient filtering operation than an FIR filter.

You can design FIR filters to provide a linear-phase response. IIR filters provide a nonlinear-phase response. Use FIR filters for applications that require linear-phase responses. Use IIR filters for applications that do not require phase information, such as signal monitoring applications.

Refer to the *Selecting a Digital Filter Design* section of this chapter for more information about selecting a digital filter type.

Nonlinear Filters

Smoothing windows, IIR filters, and FIR filters are linear because they satisfy the superposition and proportionality principles, as shown in Equation 3-14.

$$L\{ax(t) + by(t)\} = aL\{x(t)\} + bL\{y(t)\} \quad (3\text{-}14)$$

where a and b are constants, $x(t)$ and $y(t)$ are signals, $L\{\bullet\}$ is a linear filtering operation, and inputs and outputs are related through the convolution operation, as shown in Equations 3-9 and 3-11.

A nonlinear filter does not satisfy Equation 3-14. Also, you cannot obtain the output signals of a nonlinear filter through the convolution operation because a set of coefficients cannot characterize the impulse response of the filter. Nonlinear filters provide specific filtering characteristics that are difficult to obtain using linear techniques.

The median filter, a nonlinear filter, combines lowpass filter characteristics and high-frequency characteristics. The lowpass filter characteristics allow the median filter to remove high-frequency noise. The high-frequency characteristics allow the median filter to detect edges, which preserves edge information.

Example: Analyzing Noisy Pulse with a Median Filter

The Pulse Parameters VI analyzes an input sequence for a pulse pattern and determines the best set of pulse parameters that describes the pulse. After the VI completes modal analysis to determine the baseline and the top of the input sequence, discriminating between noise and signal becomes difficult without more information. Therefore, to accurately determine the pulse parameters, the peak amplitude of the noise portion of the input sequence must be less than or equal to 50% of the expected pulse amplitude. In some practical applications, a 50% pulse-to-noise ratio is difficult to achieve. Achieving the necessary pulse-to-noise ratio requires a preprocessing operation to extract pulse information.

If the pulse is buried in noise whose expected peak amplitude exceeds 50% of the expected pulse amplitude, you can use a lowpass filter to remove some of the unwanted noise. However, the filter also shifts the signal in time and smears the edges of the pulse because the transition edges contain high-frequency information. A median filter can extract the pulse more effectively than a lowpass filter because the median filter removes high-frequency noise while preserving edge information.

Figure 3-23 shows the block diagram of a VI that generates and analyzes a noisy pulse.

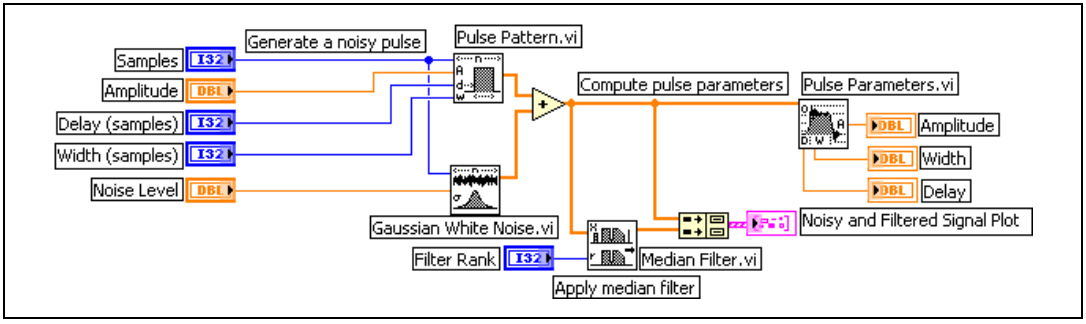


Figure 3-23. Using a Median Filter to Extract Pulse Information

The VI in Figure 3-23 generates a noisy pulse with an expected peak noise amplitude greater than 100% of the expected pulse amplitude. The signal the VI in Figure 3-23 generates has the following ideal pulse values:

- Amplitude of 5.0 V
- Delay of 64 samples
- Width of 32 samples

Figure 3-24 shows the noisy pulse, the filtered pulse, and the estimated pulse parameters returned by the VI in Figure 3-23.

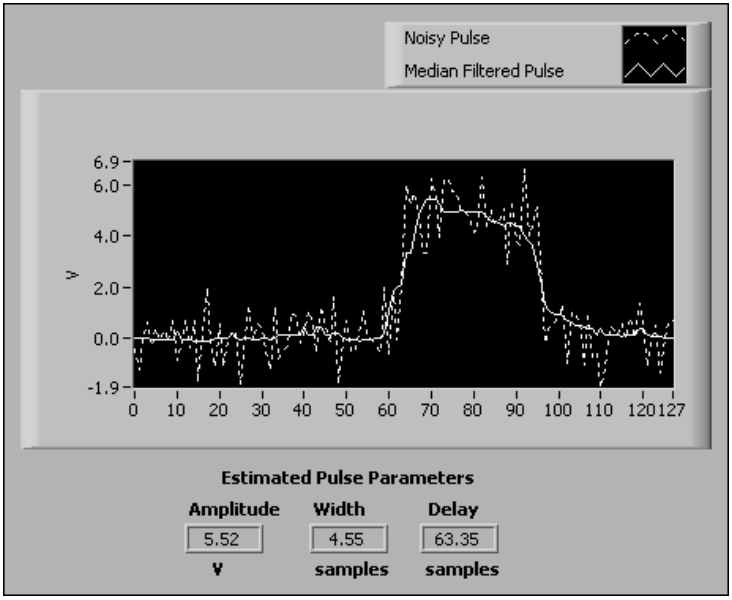


Figure 3-24. Noisy Pulse and Pulse Filtered with Median Filter

In Figure 3-24, you can track the pulse signal produced by the median filter, even though noise obscures the pulse.

You can remove the high-frequency noise with the Median Filter VI to achieve the 50% pulse-to-noise ratio the Pulse Parameters VI needs to complete the analysis accurately.

Selecting a Digital Filter Design

Answer the following questions to select a filter for an application:

- Does the analysis require a linear-phase response?
- Can the analysis tolerate ripples?
- Does the analysis require a narrow transition band?

Use Figure 3-25 as a guideline for selecting the appropriate filter for an analysis application.

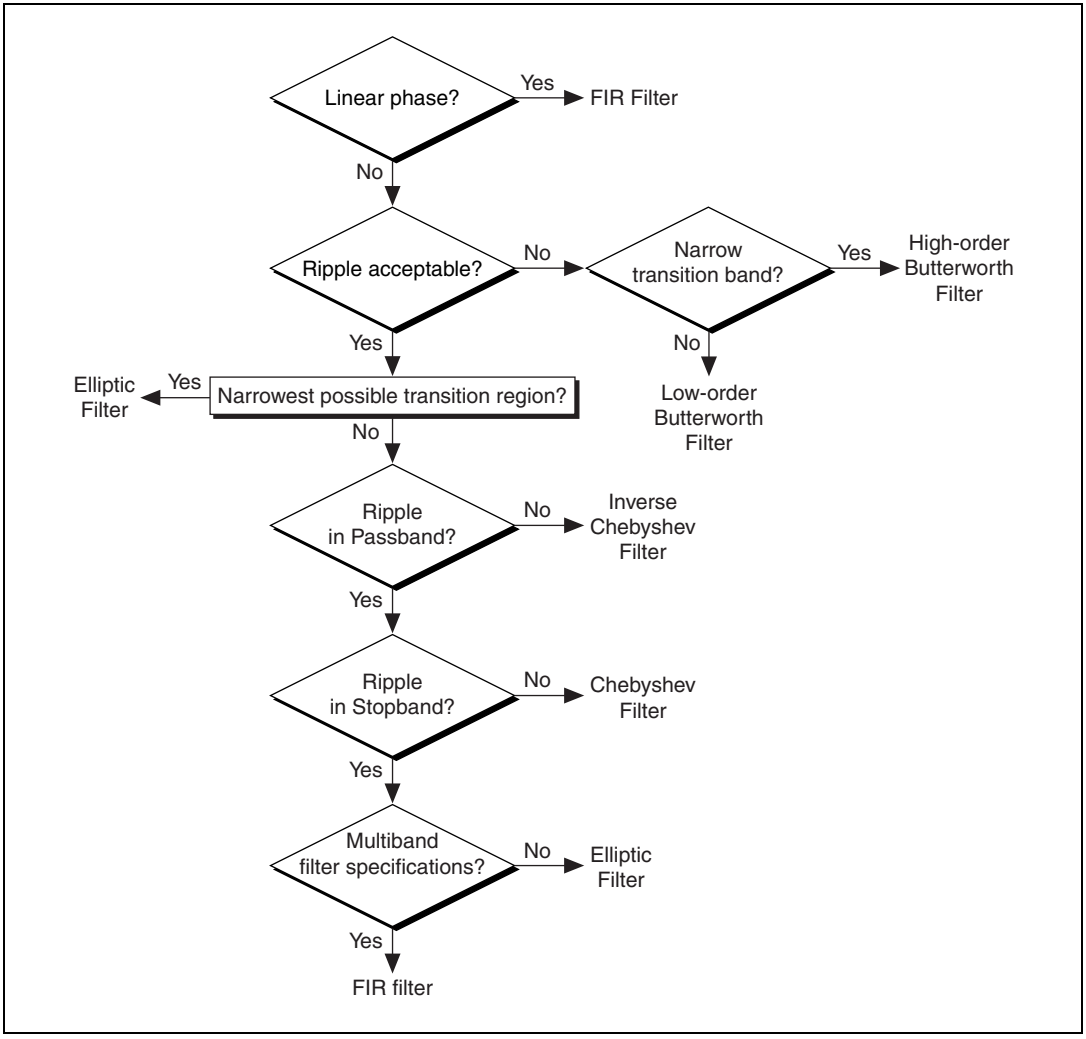


Figure 3-25. Filter Flowchart

Figure 3-25 can provide guidance for selecting an appropriate filter type. However, you might need to experiment with several filter types to find the best type.

4

Frequency Analysis

This chapter describes the fundamentals of the discrete Fourier transform (DFT), the fast Fourier transform (FFT), basic signal analysis computations, computations performed on the power spectrum, and how to use FFT-based functions for network measurement. Use the NI Example Finder to find examples of using the digital signal processing VIs and the measurement analysis VIs to perform FFT and frequency analysis.

Differences between Frequency Domain and Time Domain

The time-domain representation gives the amplitudes of the signal at the instants of time during which it was sampled. However, in many cases you need to know the frequency content of a signal rather than the amplitudes of the individual samples.

Fourier's theorem states that any waveform in the time domain can be represented by the weighted sum of sines and cosines. The same waveform then can be represented in the frequency domain as a pair of amplitude and phase values at each component frequency.

You can generate any waveform by adding sine waves, each with a particular amplitude and phase. Figure 4-1 shows the original waveform, labeled *sum*, and its component frequencies. The fundamental frequency is shown at the frequency f_0, the second harmonic at frequency $2f_0$, and the third harmonic at frequency $3f_0$.

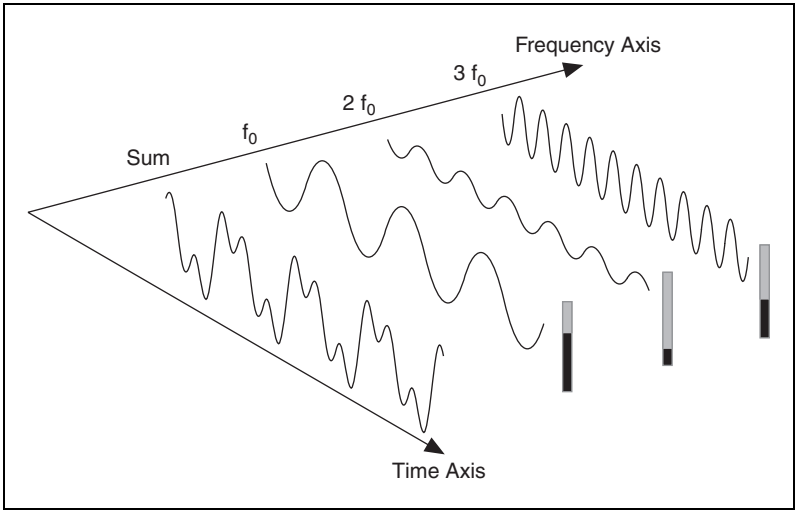


Figure 4-1. Signal Formed by Adding Three Frequency Components

In the frequency domain, you can separate conceptually the sine waves that add to form the complex time-domain signal. Figure 4-1 shows single frequency components, which spread out in the time domain, as distinct impulses in the frequency domain. The amplitude of each frequency line is the amplitude of the time waveform for that frequency component. The representation of a signal in terms of its individual frequency components is the frequency-domain representation of the signal. The frequency-domain representation might provide more insight about the signal and the system from which it was generated.

The samples of a signal obtained from a DAQ device constitute the time-domain representation of the signal. Some measurements, such as harmonic distortion, are difficult to quantify by inspecting the time waveform on an oscilloscope. When the same signal is displayed in the frequency domain by an FFT Analyzer, also known as a Dynamic Signal Analyzer, you easily can measure the harmonic frequencies and amplitudes.

Parseval's Relationship

Parseval's Theorem states that the total energy computed in the time domain must equal the total energy computed in the frequency domain. It is a statement of conservation of energy. The following equation defines the continuous form of Parseval's relationship.

$$\int_{-\infty}^{\infty} x(t)x(t)dt = \int_{-\infty}^{\infty} |X(f)|^2 df$$

The following equation defines the discrete form of Parseval's relationship.

$$\sum_{i=0}^{n-1} |x_i|^2 = \frac{1}{n}\sum_{k=0}^{n-1} |X_k|^2 \qquad (4\text{-}1)$$

where $x_i \Leftrightarrow X_k$ is a discrete FFT pair and n is the number of elements in the sequence.

Figure 4-2 shows the block diagram of a VI that demonstrates Parseval's relationship.

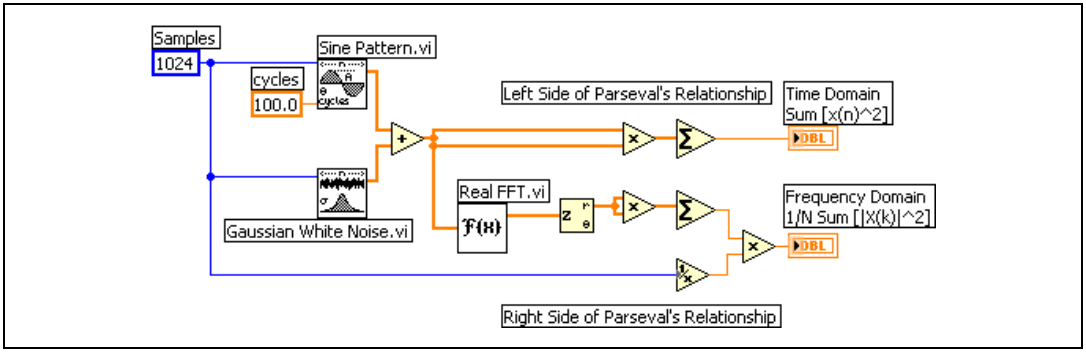


Figure 4-2. VI Demonstrating Parseval's Theorem

The VI in Figure 4-2 produces a real input sequence. The upper branch on the block diagram computes the energy of the time-domain signal using the left side of Equation 4-1. The lower branch on the block diagram converts the time-domain signal to the frequency domain and computes the energy of the frequency-domain signal using the right side of Equation 4-1.

Figure 4-3 shows the results returned by the VI in Figure 4-2.

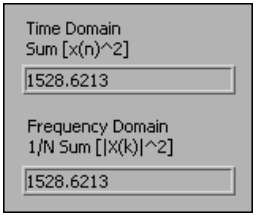


Figure 4-3. Results from Parseval VI

In Figure 4-3, the total computed energy in the time domain equals the total computed energy in the frequency domain.

Fourier Transform

The Fourier transform provides a method for examining a relationship in terms of the frequency domain. The most common applications of the Fourier transform are the analysis of linear time-invariant systems and spectral analysis.

The following equation defines the two-sided Fourier transform.

$$X(f) = F\{x(t)\} = \int_{-\infty}^{\infty} x(t)e^{-j2\pi ft}dt$$

The following equation defines the two-sided inverse Fourier transform.

$$x(t) = F^{-1}\{X(f)\} = \int_{-\infty}^{\infty} X(f)e^{j2\pi ft}df$$

Two-sided means that the mathematical implementation of the forward and inverse Fourier transform considers all negative and positive frequencies and time of the signal. Single-sided means that the mathematical implementation of the transforms considers only the positive frequencies and time history of the signal.

A Fourier transform pair consists of the signal representation in both the time and frequency domain. The following relationship commonly denotes a Fourier transform pair.

$$x(t) \Leftrightarrow X(f)$$

Discrete Fourier Transform (DFT)

The algorithm used to transform samples of the data from the time domain into the frequency domain is the discrete Fourier transform (DFT). The DFT establishes the relationship between the samples of a signal in the time domain and their representation in the frequency domain. The DFT is widely used in the fields of spectral analysis, applied mechanics, acoustics, medical imaging, numerical analysis, instrumentation, and telecommunications. Figure 4-4 illustrates using the DFT to transform data from the time domain into the frequency domain.

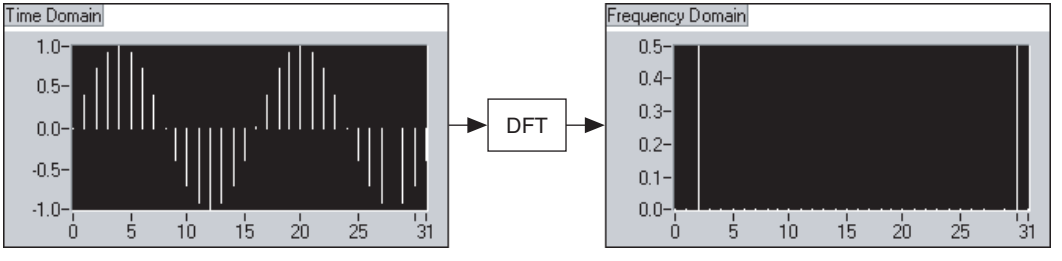


Figure 4-4. Discrete Fourier Transform

Suppose you obtained *N* samples of a signal from a DAQ device. If you apply the DFT to *N* samples of this time-domain representation of the signal, the result also is of length *N* samples, but the information it contains is of the frequency-domain representation.

Relationship between *N* Samples in the Frequency and Time Domains

If a signal is sampled at a given sampling rate, Equation 4-2 defines the time interval between the samples, or the sampling interval.

$$\Delta t = \frac{1}{f_s} \tag{4-2}$$

where Δt is the sampling interval and f_s is the sampling rate in samples per second (S/s).

The sampling interval is the smallest frequency that the system can resolve through the DFT or related routines.

Equation 4-3 defines the DFT. The equation results in $X[k]$, the frequency-domain representation of the sample signal.

$$X[k] = \sum_{i=0}^{N-1} x[i] e^{-j2\pi ik/N} \qquad \text{for } k = 0,1,2, \ldots, N-1, \tag{4-3}$$

where $x[i]$ is the time-domain representation of the sample signal and N is the total number of samples. Both the time domain x and the frequency domain X have a total of N samples.

Similar to the time spacing of Δt between the samples of x in the time domain, you have a frequency spacing, or frequency resolution, between the components of X in the frequency domain, which Equation 4-4 defines.

$$\Delta f = \frac{f_s}{N} = \frac{1}{N\Delta t} \tag{4-4}$$

where Δf is the frequency resolution, f_s is the sampling rate, N is the number of samples, Δt is the sampling interval, and $N\Delta t$ is the total acquisition time.

To improve the frequency resolution, that is, to decrease Δf, you must increase N and keep f_s constant or decrease f_s and keep N constant. Both approaches are equivalent to increasing $N\Delta t$, which is the time duration of the acquired samples.

Example of Calculating DFT

This section provides an example of using Equation 4-3 to calculate the DFT for a DC signal. This example uses the following assumptions:

- $X[0]$ corresponds to the DC component, or the average value, of the signal.
- The DC signal has a constant amplitude of +1 V.
- The number of samples is four samples.
- Each of the samples has a value +1, as shown in Figure 4-5.
- The resulting time sequence for the four samples is given by the following equation.

$$x[0] = x[1] = x[3] = x[4] = 1$$

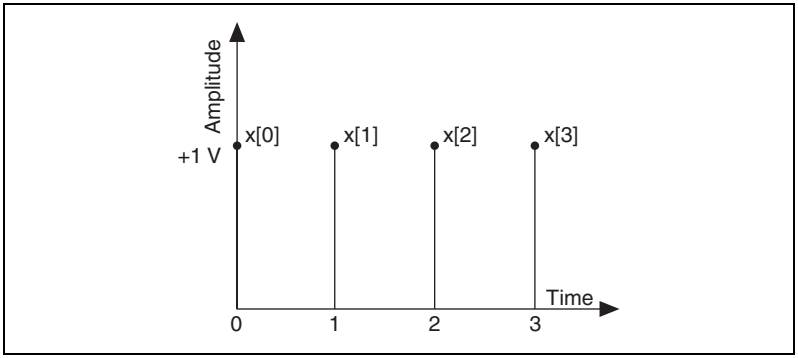


Figure 4-5. Time Sequence for DFT Samples

The DFT calculation makes use of Euler's identity, which is given by the following equation.

$$\exp(-i\theta) = \cos(\theta) - j\sin(\theta)$$

If you use Equation 4-3 to calculate the DFT of the sequence shown in Figure 4-5 and use Euler's identity, you get the following equations.

$$X[0] = \sum_{i=0}^{N-1} x_i e^{-j2\pi i 0/N} = x[0] + x[1] + x[2] + x[3] = 4$$

$$X[1] = x[0] + x[1]\left(\cos\left(\frac{\pi}{2}\right) - j\sin\left(\frac{\pi}{2}\right)\right) + x[2](\cos(\pi) - j\sin(\pi)) +$$
$$x[3]\left(\cos\left(\frac{3\pi}{2}\right) - j\sin\left(\frac{3\pi}{2}\right)\right) = (1 - j - 1 + j) = 0$$

$$X[2] = x[0] + x[1](\cos(\pi) - j\sin(\pi)) + x[2](\cos(2\pi) - j\sin(2\pi)) +$$
$$x[3](\cos(3\pi) - j\sin(3\pi)) = (1 - 1 + 1 - 1) = 0$$

$$X[3] = x[0] + x[1]\left(\cos\left(\frac{3\pi}{2}\right) - j\sin\left(\frac{3\pi}{2}\right)\right) + x[2](\cos(3\pi) - j\sin(3\pi)) +$$
$$x[3]\left(\cos\left(\frac{9\pi}{2}\right) - j\sin\left(\frac{9\pi}{2}\right)\right) = (1 - j - 1 - j) = 0$$

where $X[0]$ is the DC component and N is the number of samples.

Therefore, except for the DC component, all other values for the sequence shown in Figure 4-5 are zero, which is as expected. However, the calculated value of $X[0]$ depends on the value of N. Because in this example $N = 4$, $X[0] = 4$. If $N = 10$, the calculation results in $X[0] = 10$. This dependency of $X[\ ]$ on N also occurs for the other frequency components. Therefore, you usually divide the DFT output by N to obtain the correct magnitude of the frequency component.

Magnitude and Phase Information

N samples of the input signal result in N samples of the DFT. That is, the number of samples in both the time and frequency representations is the same. Equation 4-3 shows that regardless of whether the input signal $x[i]$ is real or complex, $X[k]$ is always complex, although the imaginary part may be zero. In other words, every frequency component has a magnitude and phase.

Normally the magnitude of the spectrum is displayed. The magnitude is the square root of the sum of the squares of the real and imaginary parts.

The phase is relative to the start of the time record or relative to a single-cycle cosine wave starting at the beginning of the time record. Single-channel phase measurements are stable only if the input signal is triggered. Dual-channel phase measurements compute phase differences between channels so if the channels are sampled simultaneously, triggering usually is not necessary.

The phase is the arctangent of the ratio of the imaginary and real parts and is usually between π and $-\pi$ radians, or 180 and –180 degrees.

For real signals ($x[i]$ real), such as those you obtain from the output of one channel of a DAQ device, the DFT is symmetric with properties given by the following equations.

$$|X[k]| = |X[N - k]|$$

$$\text{phase}(X[k]) = -\text{phase}(X[N - k])$$

The magnitude of $X[k]$ is even symmetric, and phase($X[k]$) is odd symmetric. An even symmetric signal is symmetric about the y-axis, and an odd symmetric signal is symmetric about the origin. Figure 4-6 illustrates even and odd symmetry.

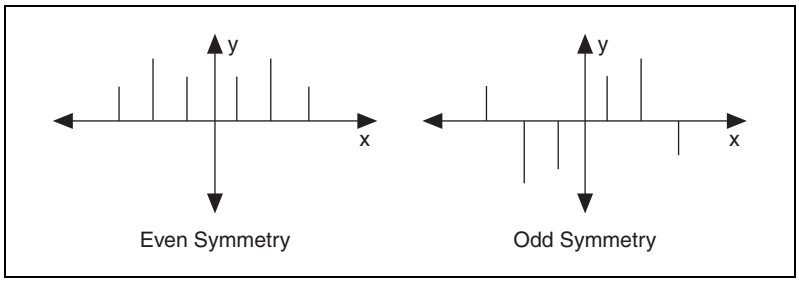


Figure 4-6. Signal Symmetry

Because of this symmetry, the N samples of the DFT contain repetition of information. Because of this repetition of information, only half of the samples of the DFT actually need to be computed or displayed because you can obtain the other half from this repetition. If the input signal is complex, the DFT is asymmetrical, and you cannot use only half of the samples to obtain the other half.

Frequency Spacing between DFT Samples

If the sampling interval is Δt seconds and the first data sample ($k = 0$) is at 0 seconds, the k^{th} data sample, where $k > 0$ and is an integer, is at $k\Delta t$ seconds. Similarly, if the frequency resolution is Δf Hz, the k^{th} sample of the DFT occurs at a frequency of $k\Delta f$ Hz. However, this is valid for only up to the first half of the frequency components. The other half represent negative frequency components.

Depending on whether the number of samples N is even or odd, you can have a different interpretation of the frequency corresponding to the k^{th} sample of the DFT. For example, let $N = 8$ and p represent the index of the Nyquist frequency $p = N/2 = 4$. Table 4-1 shows the Δf to which each format element of the complex output sequence X corresponds.

Table 4-1. $X[p]$ for $N = 8$

$X[p]$	Δf
$X[0]$	DC
$X[1]$	Δf
$X[2]$	$2\Delta f$
$X[3]$	$3\Delta f$
$X[4]$	$4\Delta f$ (Nyquist frequency)
$X[5]$	$-3\Delta f$
$X[6]$	$-2\Delta f$
$X[7]$	$-\Delta f$

The negative entries in the second column beyond the Nyquist frequency represent negative frequencies, that is, those elements with an index value $>p$.

For $N = 8$, $X[1]$ and $X[7]$ have the same magnitude; $X[2]$ and $X[6]$ have the same magnitude; and $X[3]$ and $X[5]$ have the same magnitude. The difference is that $X[1]$, $X[2]$, and $X[3]$ correspond to positive frequency components, while $X[5]$, $X[6]$, and $X[7]$ correspond to negative frequency components. $X[4]$ is at the Nyquist frequency.

Figure 4-7 illustrates the complex output sequence X for $N = 8$.

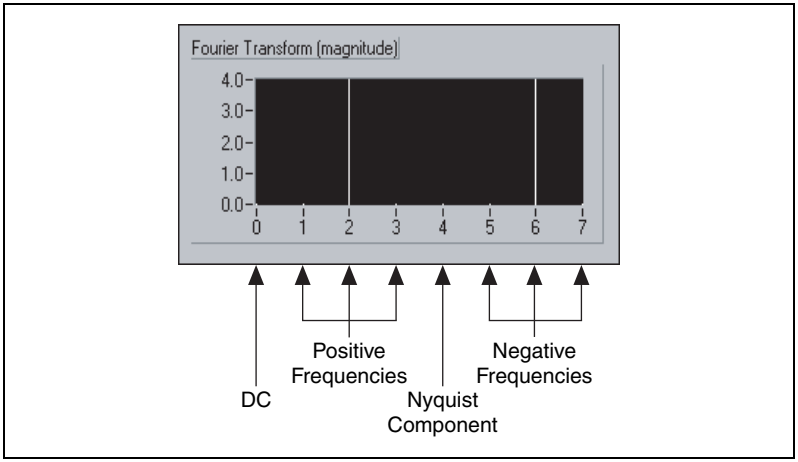


Figure 4-7. Complex Output Sequence X for $N = 8$

A representation where you see the positive and negative frequencies is the two-sided transform.

When N is odd, there is no component at the Nyquist frequency. Table 4-2 lists the values of Δf for $X[p]$ when $N = 7$ and $p = (N-1)/2 = (7-1)/2 = 3$.

Table 4-2. $X[p]$ for $N = 7$

$X[p]$	Δf
$X[0]$	DC
$X[1]$	Δf
$X[2]$	$2\Delta f$
$X[3]$	$3\Delta f$
$X[4]$	$-3\Delta f$
$X[5]$	$-2\Delta f$
$X[6]$	$-\Delta f$

For $N = 7$, $X[1]$ and $X[6]$ have the same magnitude; $X[2]$ and $X[5]$ have the same magnitude; and $X[3]$ and $X[4]$ have the same magnitude. However, $X[1]$, $X[2]$, and $X[3]$ correspond to positive frequencies, while $X[4]$, $X[5]$, and $X[6]$ correspond to negative frequencies. Because N is odd, there is no component at the Nyquist frequency.

Figure 4-8 illustrates the complex output sequence $X[p]$ for $N = 7$.

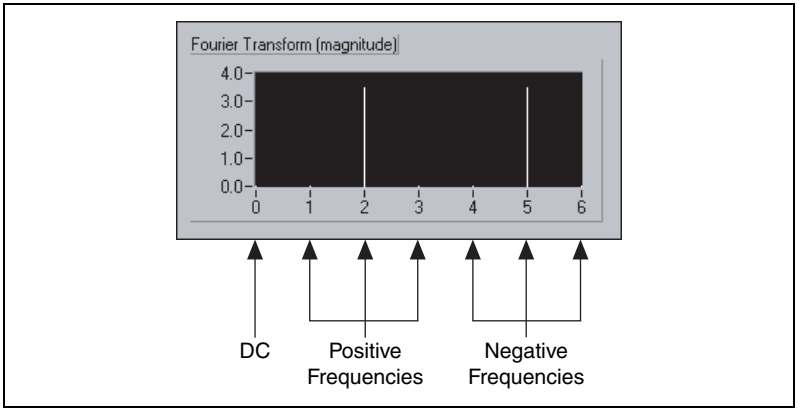


Figure 4-8. Complex Output Sequence $X[p]$ for $N = 7$

Figure 4-8 also shows a two-sided transform because it represents the positive and negative frequencies.

FFT Fundamentals

Directly implementing the DFT on N data samples requires approximately N^2 complex operations and is a time-consuming process. The FFT is a fast algorithm for calculating the DFT. The following equation defines the DFT.

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-j\left(\frac{2\pi nk}{N}\right)}$$

The following measurements comprise the basic functions for FFT-based signal analysis:

- FFT
- Power spectrum
- Cross power spectrum

You can use the basic functions as the building blocks for creating additional measurement functions, such as the frequency response, impulse response, coherence, amplitude spectrum, and phase spectrum.

The FFT and the power spectrum are useful for measuring the frequency content of stationary or transient signals. The FFT produces the average frequency content of a signal over the total acquisition. Therefore, use the FFT for stationary signal analysis or in cases where you need only the average energy at each frequency line.

An FFT is equivalent to a set of parallel filters of bandwidth Δf centered at each frequency increment from DC to $(F_s/2) - (F_s/N)$. Therefore, frequency lines also are known as frequency bins or FFT bins.

Refer to the *Power Spectrum* section of this chapter for more information about the power spectrum.

Computing Frequency Components

Each frequency component is the result of a dot product of the time-domain signal with the complex exponential at that frequency and is given by the following equation.

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-j\left(\frac{2\pi nk}{N}\right)} = \sum_{n=0}^{N-1} x(n)\left[\cos\left(\frac{2\pi nk}{N}\right) - j\sin\left(\frac{2\pi nk}{N}\right)\right]$$

The DC component is the dot product of $x(n)$ with $[\cos(0) - j\sin(0)]$, or with 1.0.

The first bin, or frequency component, is the dot product of $x(n)$ with $\cos(2\pi n/N) - j\sin(2\pi n/N)$. Here, $\cos(2\pi n/N)$ is a single cycle of the cosine wave, and $\sin(2\pi n/N)$ is a single cycle of a sine wave.

In general, bin k is the dot product of $x(n)$ with k cycles of the cosine wave for the real part of $X(k)$ and the sine wave for the imaginary part of $X(k)$.

The use of the FFT for frequency analysis implies two important relationships.

The first relationship links the highest frequency that can be analyzed to the sampling frequency and is given by the following equation.

$$F_{max} = \frac{f_s}{2},$$

where F_{max} is the highest frequency that can be analyzed and f_s is the sampling frequency. Refer to the *Windowing* section of this chapter for more information about F_{max}.

The second relationship links the frequency resolution to the total acquisition time, which is related to the sampling frequency and the block size of the FFT and is given by the following equation.

$$\Delta f = \frac{1}{T} = \frac{f_s}{N},$$

where Δf is the frequency resolution, T is the acquisition time, f_s is the sampling frequency, and N is the block size of the FFT.

Fast FFT Sizes

When the size of the input sequence is a power of two, $N = 2^m$, you can implement the computation of the DFT with approximately $N \log_2(N)$ operations, which makes the calculation of the DFT much faster. DSP literature refers to the algorithms for faster DFT calculation as fast Fourier transforms (FFTs). Common input sequence sizes that are a power of two include 512, 1,024, and 2,048.

When the size of the input sequence is not a power of two but is factorable as the product of small prime numbers, the FFT-based VIs use a mixed radix Cooley-Tukey algorithm to efficiently compute the DFT of the input sequence. For example, Equation 4-5 defines an input sequence size N as the product of small prime numbers.

$$N = 2^m 3^k 5^j \qquad \text{for } m, k, j = 0, 1, 2, 3, \ldots \tag{4-5}$$

For the input sequence size defined by Equation 4-5, the FFT-based VIs can compute the DFT with speeds comparable to an FFT whose input sequence size is a power of two. Common input sequence sizes that are factorable as the product of small prime numbers include 480, 640, 1,000, and 2,000.

Zero Padding

Zero padding is a technique typically employed to make the size of the input sequence equal to a power of two. In zero padding, you add zeros to the end of the input sequence so that the total number of samples is equal to the next higher power of two. For example, if you have 10 samples of a signal, you can add six zeros to make the total number of samples equal to 16, or 2^4, which is a power of two. Figure 4-9 illustrates padding 10 samples of a signal with zeros to make the total number of samples equal 16.

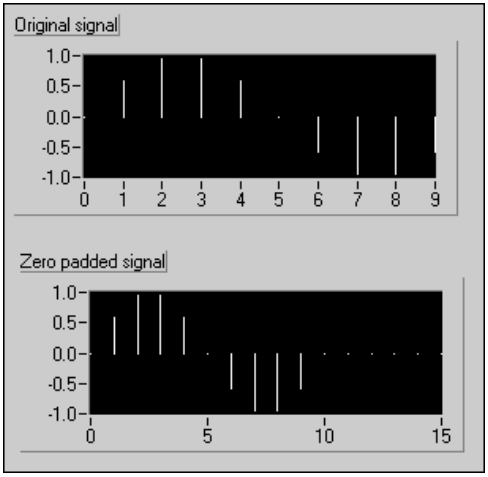


Figure 4-9. Zero Padding

The addition of zeros to the end of the time-domain waveform does not improve the underlying frequency resolution associated with the time-domain signal. The only way to improve the frequency resolution of the time-domain signal is to increase the acquisition time and acquire longer time records.

In addition to making the total number of samples a power of two so that faster computation is made possible by using the FFT, zero padding can lead to an interpolated FFT result, which can produce a higher display resolution.

FFT VI

The polymorphic FFT VI computes the FFT of a signal and has two instances—Real FFT and Complex FFT.

The difference between the two instances is that the Real FFT instance computes the FFT of a real-valued signal, whereas the Complex FFT instance computes the FFT of a complex-valued signal. However, the outputs of both instances are complex.

Most real-world signals are real valued. Therefore, you can use the Real FFT instance for most applications. You also can use the Complex FFT instance by setting the imaginary part of the signal to zero. An example of an application where you use the Complex FFT instance is when the signal consists of both a real and an imaginary component. A signal consisting of a real and an imaginary component occurs frequently

in the field of telecommunications, where you modulate a waveform by a complex exponential. The process of modulation by a complex exponential results in a complex signal, as shown in Figure 4-10.

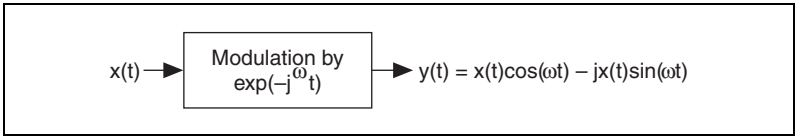


Figure 4-10. Modulation by a Complex Exponential

Displaying Frequency Information from Transforms

The discrete implementation of the Fourier transform maps a digital signal into its Fourier series coefficients, or harmonics. Unfortunately, neither a time nor a frequency stamp is directly associated with the FFT operation. Therefore, you must specify the sampling interval Δt.

Because an acquired array of samples represents a progression of equally spaced samples in time, you can determine the corresponding frequency in hertz. The following equation gives the sampling frequency f_s for Δt.

$$f_s = \frac{1}{\Delta t}$$

Figure 4-11 shows the block diagram of a VI that properly displays frequency information given the sampling interval 1.000E – 3 and returns the value for the frequency interval Δf.

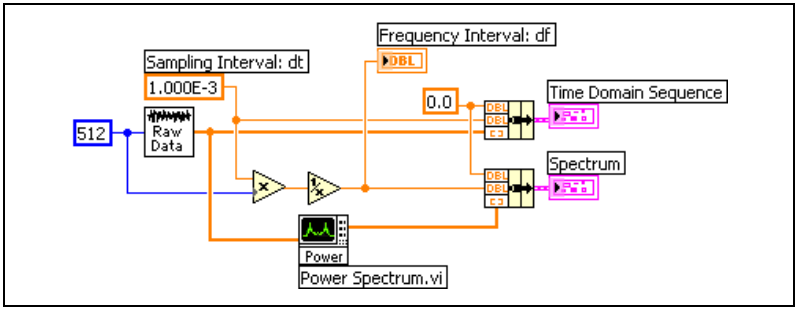


Figure 4-11. Correctly Displaying Frequency Information

Figure 4-12 shows the display and Δf that the VI in Figure 4-11 returns.

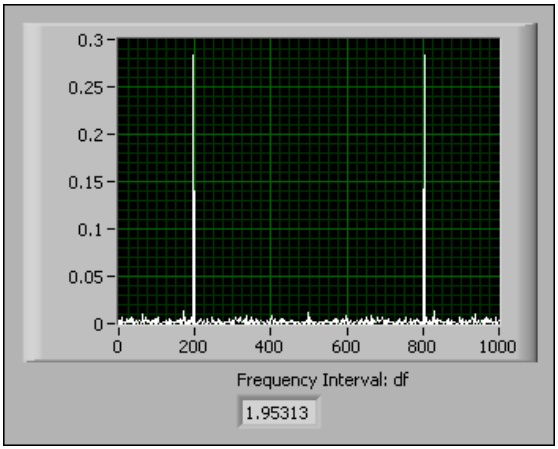


Figure 4-12. Properly Displayed Frequency Information

Two other common ways of presenting frequency information are displaying the DC component in the center and displaying one-sided spectrums. Refer to the *Two-Sided, DC-Centered FFT* section of this chapter for information about displaying the DC component in the center. Refer to the *Power Spectrum* section of this chapter for information about displaying one-sided spectrums.

Two-Sided, DC-Centered FFT

The two-sided, DC-centered FFT provides a method for displaying a spectrum with both positive and negative frequencies. Most introductory textbooks that discuss the Fourier transform and its properties present a table of two-sided Fourier transform pairs. You can use the frequency shifting property of the Fourier transform to obtain a two-sided, DC-centered representation. In a two-sided, DC-centered FFT, the DC component is in the middle of the buffer.

Mathematical Representation of a Two-Sided, DC-Centered FFT

If $x(t) \Leftrightarrow X(f)$ is a Fourier transform pair, then

$$x(t)e^{j2\pi f_0 t} \Leftrightarrow X(f - f_0)$$

Let

$$\Delta t = \frac{1}{f_s}$$

where f_s is the sampling frequency in the discrete representation of the time signal.

Set f_0 to the index corresponding to the Nyquist component f_N, as shown in the following equation.

$$f_0 = f_N = \frac{f_s}{2} = \frac{1}{2\Delta t}$$

f_0 is set to the index corresponding to f_N because causing the DC component to appear in the location of the Nyquist component requires a frequency shift equal to f_N.

Setting f_0 to the index corresponding to f_N results in the discrete Fourier transform pair shown in the following relationship.

$$x_i e^{ji\pi} \Leftrightarrow X_{k-\frac{n}{2}}$$

where n is the number of elements in the discrete sequence, x_i is the time-domain sequence, and X_k is the frequency-domain representation of x_i.

Expanding the exponential term in the time-domain sequence produces the following equation.

$$e^{ji\pi} = \cos(i\pi) + j\sin(i\pi) = \begin{cases} 1 & \text{if } i \text{ is even} \\ -1 & \text{if } i \text{ is odd} \end{cases} \quad (4\text{-}6)$$

Equation 4-6 represents a sequence of alternating +1 and –1. Equation 4-6 means that negating the odd elements of the original time-domain sequence and performing an FFT on the new sequence produces a spectrum whose DC component appears in the center of the sequence.

Therefore, if the original input sequence is

$$X = \{x_0, x_1, x_2, x_3, \ldots, x_{n-1}\}$$

then the sequence

$$Y = \{x_0, -x_1, x_2, -x_3, \ldots, x_{n-1}\} \quad (4\text{-}7)$$

generates a DC-centered spectrum.

Creating a Two-Sided, DC-Centered FFT

You can modulate a signal by the Nyquist frequency in place without extra buffers. Figure 4-13 shows the block diagram of the Nyquist Shift VI located in the `labview\examples\analysis\dspxmpl.llb`, which generates the sequence shown in Equation 4-7.

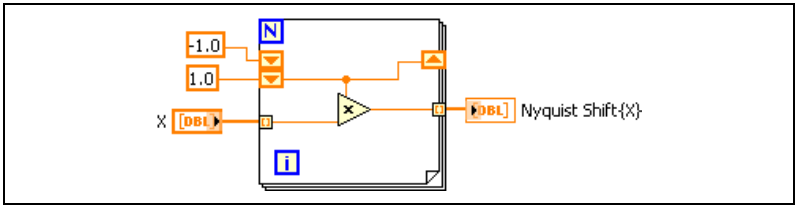


Figure 4-13. Block Diagram of the Nyquist Shift VI

In Figure 4-13, the For Loop iterates through the input sequence, alternately multiplying array elements by 1.0 and –1.0, until it processes the entire input array.

Figure 4-14 shows the block diagram of a VI that generates a time-domain sequence and uses the Nyquist Shift and Power Spectrum VIs to produce a DC-centered spectrum.

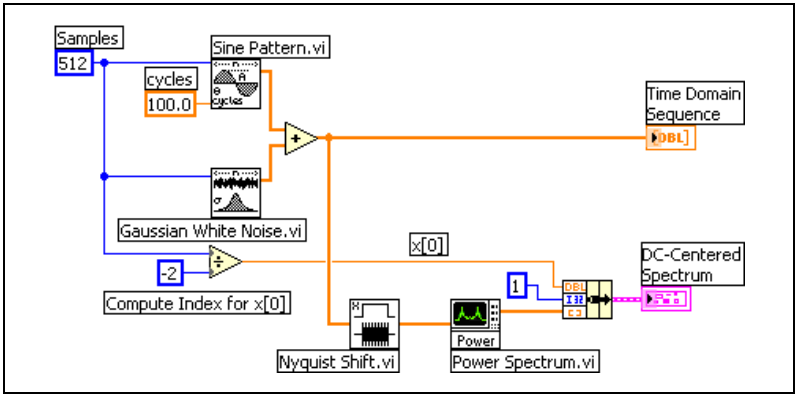


Figure 4-14. Generating Time-Domain Sequence and DC-Centered Spectrum

In the VI in Figure 4-14, the Nyquist Shift VI preprocesses the time-domain sequence by negating every other element in the sequence. The Power Spectrum VI transforms the data into the frequency domain. To display the frequency axis of the processed data correctly, you must supply x_0, which is the x-axis value of the initial frequency bin. For a DC-centered spectrum, the following equation computes x_0.

$$x_0 = -\frac{n}{2}$$

Figure 4-15 shows the time-domain sequence and DC-centered spectrum the VI in Figure 4-14 returns.

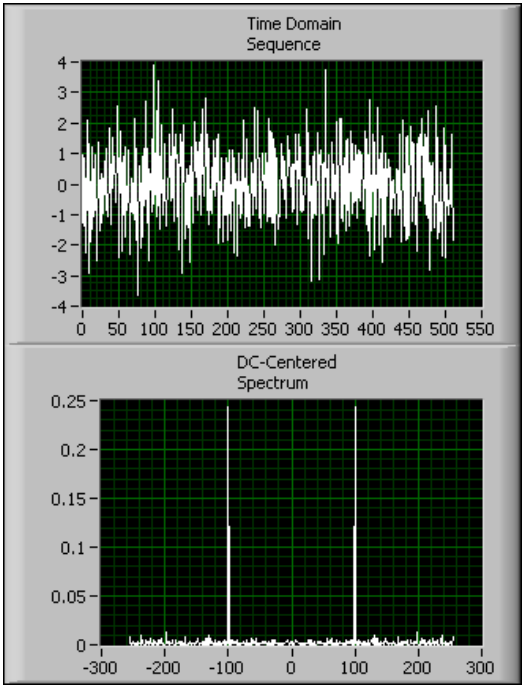


Figure 4-15. Raw Time-Domain Sequence and DC-Centered Spectrum

In the DC-centered spectrum display in Figure 4-15, the DC component appears in the center of the display at $f = 0$. The overall format resembles that commonly found in tables of Fourier transform pairs.

You can create DC-centered spectra for even-sized input sequences by negating the odd elements of the input sequence.

You cannot create DC-centered spectra by directly negating the odd elements of an input time-domain sequence containing an odd number of elements because the Nyquist frequency appears between two frequency bins. To create DC-centered spectra for odd-sized input sequences, you must rotate the FFT arrays by the amount given in the following relationship.

$$\frac{n-1}{2}$$

For a DC-centered spectrum created from an odd-sized input sequence, the following equation computes x_0.

$$x_0 = -\frac{n-1}{2}$$

Power Spectrum

As described in the *Magnitude and Phase Information* section of this chapter, the DFT or FFT of a real signal is a complex number, having a real and an imaginary part. You can obtain the power in each frequency component represented by the DFT or FFT by squaring the magnitude of that frequency component. Thus, the power in the k^{th} frequency component—that is, the k^{th} element of the DFT or FFT—is given by the following equation.

$$\text{power} = |X[k]|^2,$$

where $|X[k]|$ is the magnitude of the frequency component. Refer to the *Magnitude and Phase Information* section of this chapter for information about computing the magnitude of the frequency components.

The power spectrum returns an array that contains the two-sided power spectrum of a time-domain signal and that shows the power in each of the frequency components. You can use Equation 4-8 to compute the two-sided power spectrum from the FFT.

$$\text{Power Spectrum } S_{AA}(f) = \frac{\text{FFT(A)} \times \text{FFT*(A)}}{N} \quad (4\text{-}8)$$

where FFT*(A) denotes the complex conjugate of FFT(A). The complex conjugate of FFT(A) results from negating the imaginary part of FFT(A).

The values of the elements in the power spectrum array are proportional to the magnitude squared of each frequency component making up the time-domain signal. Because the DFT or FFT of a real signal is symmetric, the power at a positive frequency of $k\Delta f$ is the same as the power at the corresponding negative frequency of $-k\Delta f$, excluding DC and Nyquist components. The total power in the DC component is $|X[0]|^2$. The total power in the Nyquist component is $|X[N/2]|^2$.

A plot of the two-sided power spectrum shows negative and positive frequency components at a height given by the following relationship.

$$\frac{A_k^2}{4}$$

where A_k is the peak amplitude of the sinusoidal component at frequency k. The DC component has a height of A_0^2 where A_0 is the amplitude of the DC component in the signal.

Figure 4-16 shows the power spectrum result from a time-domain signal that consists of a 3 V_{rms} sine wave at 128 Hz, a 3 V_{rms} sine wave at 256 Hz, and a DC component of 2 VDC. A 3 V_{rms} sine wave has a peak voltage of 3.0 • $\sqrt{2}$ or about 4.2426 V. The power spectrum is computed from the basic FFT function, as shown in Equation 4-8.

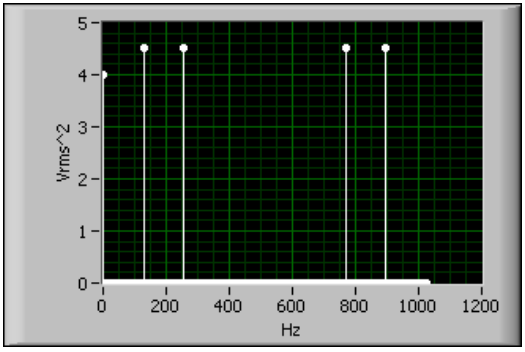


Figure 4-16. Two-Sided Power Spectrum of Signal

Converting a Two-Sided Power Spectrum to a Single-Sided Power Spectrum

Most frequency analysis instruments display only the positive half of the frequency spectrum because the spectrum of a real-world signal is symmetrical around DC. Thus, the negative frequency information is redundant. The two-sided results from the analysis functions include the positive half of the spectrum followed by the negative half of the spectrum, as shown in Figure 4-16.

A two-sided power spectrum displays half the energy at the positive frequency and half the energy at the negative frequency. Therefore, to convert a two-sided spectrum to a single-sided spectrum, you discard the

second half of the array and multiply every point except for DC by two, as shown in the following equations.

$$G_{AA}(i) = S_{AA}(i),\ i = 0\ (\text{DC})$$

$$G_{AA}(i) = (2S_{AA}(i)),\ i = 1 \text{ to } \frac{N}{2} - 1$$

where $S_{AA}(i)$ is the two-sided power spectrum, $G_{AA}(i)$ is the single-sided power spectrum, and N is the length of the two-sided power spectrum. You discard the remainder of the two-sided power spectrum S_{AA}, $N/2$ through $N - 1$.

The non-DC values in the single-sided spectrum have a height given by the following relationship.

$$\frac{A_k^2}{2} \tag{4-9}$$

Equation 4-9 is equivalent to the following relationship.

$$\left(\frac{A_k}{\sqrt{2}}\right)^2$$

where $\frac{A_k}{\sqrt{2}}$ is the root mean square (rms) amplitude of the sinusoidal component at frequency k.

The units of a power spectrum are often quantity squared rms, where quantity is the unit of the time-domain signal. For example, the single-sided power spectrum of a voltage waveform is in volts rms squared, V_{rms}^2.

Figure 4-17 shows the single-sided spectrum of the signal whose two-sided spectrum Figure 4-16 shows.

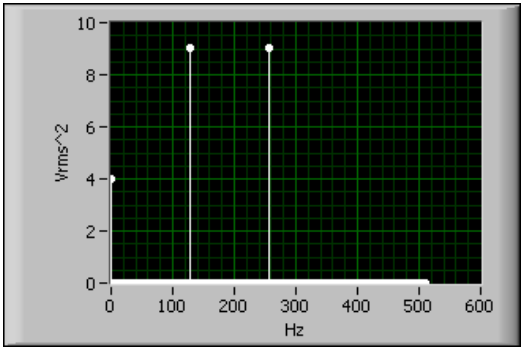


Figure 4-17. Single-Sided Power Spectrum

In Figure 4-17, the height of the non-DC frequency components is twice the height of the non-DC frequency component in Figure 4-16. Also, the spectrum in Figure 4-17 stops at half the frequency of that in Figure 4-16.

Loss of Phase Information

Because the power is obtained by squaring the magnitude of the DFT or FFT, the power spectrum is always real. The disadvantage of obtaining the power by squaring the magnitude of the DFT or FFT is that the phase information is lost. If you want phase information, you must use the DFT or FFT, which gives you a complex output.

You can use the power spectrum in applications where phase information is not necessary, such as calculating the harmonic power in a signal. You can apply a sinusoidal input to a nonlinear system and see the power in the harmonics at the system output.

Computations on the Spectrum

When you have the amplitude or power spectrum, you can compute several useful characteristics of the input signal, such as power and frequency, noise level, and power spectral density.

Estimating Power and Frequency

If a frequency component is between two frequency lines, the frequency component appears as energy spread among adjacent frequency lines with reduced amplitude. The actual peak is between the two frequency lines. You can estimate the actual frequency of a discrete frequency component to a greater resolution than the Δf given by the FFT by performing a

weighted average of the frequencies around a detected peak in the power spectrum, as shown in the following equation.

$$\text{Estimated Frequency} = \frac{\sum_{i=j-3}^{j+3} (\text{Power}(i)(i\Delta f))}{\sum_{i=j-3}^{j+3} \text{Power}(i)}$$

where j is the array index of the apparent peak of the frequency of interest.

The span $j \pm 3$ is reasonable because it represents a spread wider than the main lobes of the smoothing windows listed in Table 5-3, *Correction Factors and Worst-Case Amplitude Errors for Smoothing Windows*, of Chapter 5, *Smoothing Windows*.

You can estimate the power in V^2_{rms} of a discrete peak frequency component by summing the power in the bins around the peak. In other words, you compute the area under the peak. You can use the following equation to estimate the power of a discrete peak frequency component.

$$\text{Estimated Power} = \frac{\sum_{i=j-3}^{j+3} \text{Power}(i)}{\text{noise power bandwidth of window}} \quad (4\text{-}10)$$

Equation 4-10 is valid only for a spectrum made up of discrete frequency components. It is not valid for a continuous spectrum. Also, if two or more frequency peaks are within six lines of each other, they contribute to inflating the estimated powers and skewing the actual frequencies. You can reduce this effect by decreasing the number of lines spanned by Equation 4-10. If two peaks are within six lines of each other, it is likely that they are already interfering with one another because of spectral leakage.

If you want the total power in a given frequency range, sum the power in each bin included in the frequency range and divide by the noise power bandwidth of the smoothing window. Refer to Chapter 5, *Smoothing Windows*, for information about the noise power bandwidth of smoothing windows.

Computing Noise Level and Power Spectral Density

The measurement of noise levels depends on the bandwidth of the measurement. When looking at the noise floor of a power spectrum, you are looking at the narrowband noise level in each FFT bin. Therefore, the noise floor of a given power spectrum depends on the Δf of the spectrum, which is in turn controlled by the sampling rate and the number of points in the data set. In other words, the noise level at each frequency line is equivalent to the noise level obtained using a Δf Hz filter centered at that frequency line. Therefore, for a given sampling rate, doubling the number of data points acquired reduces the noise power that appears in each bin by 3 dB. Theoretically, discrete frequency components have zero bandwidth and therefore do not scale with the number of points or frequency range of the FFT.

To compute the signal-to-noise ratio (SNR), compare the peak power in the frequencies of interest to the broadband noise level. Compute the broadband noise level in V_{rms}^2 by summing all the power spectrum bins, excluding any peaks and the DC component, and dividing the sum by the equivalent noise bandwidth of the window.

Because of noise-level scaling with Δf, spectra for noise measurement often are displayed in a normalized format called power or amplitude spectral density. The power or amplitude spectral density normalizes the power or amplitude spectrum to the spectrum measured by a 1 Hz-wide square filter, a convention for noise-level measurements. The level at each frequency line is equivalent to the level obtained using a 1 Hz filter centered at that frequency line.

You can use the following equation to compute the power spectral density.

$$\text{Power Spectral Density} = \frac{\text{Power Spectrum in } V_{rms}^2}{\Delta f \times \text{Noise Power Bandwidth of Window}}$$

You can use the following equation to compute the amplitude spectral density.

$$\text{Amplitude Spectral Density} = \frac{\text{Amplitude Spectrum in } V_{rms}}{\sqrt{\Delta f \times \text{Noise Power Bandwidth of Window}}}$$

The spectral density format is appropriate for random or noise signals. The spectral density format is not appropriate for discrete frequency components because discrete frequency components theoretically have zero bandwidth.

Computing the Amplitude and Phase Spectrums

The power spectrum shows power as the mean squared amplitude at each frequency line but includes no phase information. Because the power spectrum loses phase information, you might want to use the FFT to view both the frequency and the phase information of a signal.

The phase information the FFT provides is the phase relative to the start of the time-domain signal. Therefore, you must trigger from the same point in the signal to obtain consistent phase readings. A sine wave shows a phase of –90° at the sine wave frequency. A cosine wave shows a 0° phase. Usually, the primary area of interest for analysis applications is either the relative phases between components or the phase difference between two signals acquired simultaneously. You can view the phase difference between two signals by using some of the advanced FFT functions. Refer to the *Frequency Response and Network Analysis* section of this chapter for information about the advanced FFT functions.

The FFT produces a two-sided spectrum in complex form with real and imaginary parts. You must scale and convert the two-sided spectrum to polar form to obtain magnitude and phase. The frequency axis of the polar form is identical to the frequency axis of the two-sided power spectrum. The amplitude of the FFT is related to the number of points in the time-domain signal. Use the following equations to compute the amplitude and phase versus frequency from the FFT.

$$\text{Amplitude spectrum in quantity peak} = \frac{\text{Magnitude}[\text{FFT}(A)]}{N} = \frac{\sqrt{[\text{real}[\text{FFT}(A)]]^2 + [\text{imag}[\text{FFT}(A)]]^2}}{N} \quad (4\text{-}11)$$

$$\text{Phase spectrum in radians} = \text{Phase}[\text{FFT}(A)] = \text{arctangent}\left(\frac{\text{imag}[\text{FFT}(A)]}{\text{real}[\text{FFT}(A)]}\right) \quad (4\text{-}12)$$

where the arctangent function returns values of phase between $-\pi$ and $+\pi$, a full range of 2π radians.

The following relationship defines the rectangular-to-polar conversion function.

$$\frac{\text{FFT}(A)}{N} \quad (4\text{-}13)$$

Using the rectangular-to-polar conversion function to convert the complex spectrum to its magnitude (r) and phase (ϕ) is equivalent to using Equations 4-11 and 4-12.

The two-sided amplitude spectrum actually shows half the peak amplitude at the positive and negative frequencies. To convert to the single-sided form, multiply each frequency, other than DC, by two and discard the second half of the array. The units of the single-sided amplitude spectrum are then in quantity peak and give the peak amplitude of each sinusoidal component making up the time-domain signal.

To obtain the single-sided phase spectrum, discard the second half of the array.

Calculating Amplitude in V_{rms} and Phase in Degrees

To view the amplitude spectrum in volts rms (V_{rms}), divide the non-DC components by the square root of two after converting the spectrum to the single-sided form. Because you multiply the non-DC components by two to convert from the two-sided amplitude spectrum to the single-sided amplitude spectrum, you can calculate the rms amplitude spectrum directly from the two-sided amplitude spectrum by multiplying the non-DC components by the square root of two and discarding the second half of the array. The following equations show the entire computation from a two-sided FFT to a single-sided amplitude spectrum.

$$\text{Amplitude Spectrum } V_{rms} = \sqrt{2}\,\frac{\text{Magnitude}[\text{FFT}(A)]}{N} \qquad \text{for } i = 1 \text{ to } \frac{N}{2} - 1$$

$$\text{Amplitude Spectrum } V_{rms} = \frac{\text{Magnitude}[\text{FFT}(A)]}{N} \qquad \text{for } i = 0 \text{ (DC)}$$

where i is the frequency line number, or array index, of the FFT of A.

The magnitude in V_{rms} gives the rms voltage of each sinusoidal component of the time-domain signal.

The amplitude spectrum is closely related to the power spectrum. You can compute the single-sided power spectrum by squaring the single-sided rms amplitude spectrum. Conversely, you can compute the amplitude spectrum by taking the square root of the power spectrum. Refer to the *Power Spectrum* section of this chapter for information about computing the power spectrum.

Use the following equation to view the phase spectrum in degrees.

$$\text{Phase Spectrum in Degrees} = \frac{180}{\pi}\text{Phase FFT}(A)$$

Frequency Response Function

When analyzing two simultaneously sampled channels, you usually want to know the differences between the two channels rather than the properties of each.

In a typical dual-channel analyzer, as shown in Figure 4-18, the instantaneous spectrum is computed using a window function and the FFT for each channel. The averaged FFT spectrum, auto power spectrum, and cross power spectrum are computed and used in estimating the frequency response function. You also can use the coherence function to check the validity of the frequency response function.

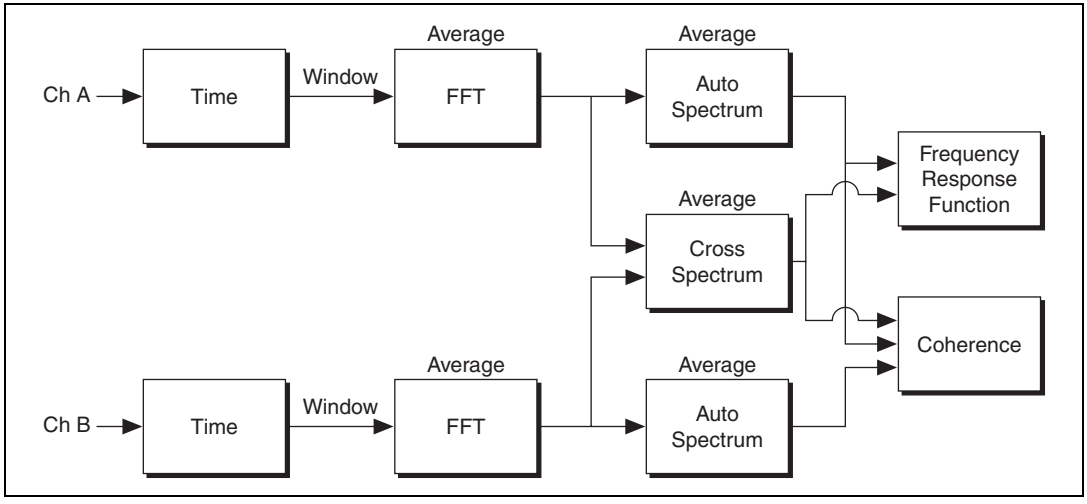


Figure 4-18. Dual-Channel Frequency Analysis

The frequency response of a system is described by the magnitude, |H|, and phase, ∠H, at each frequency. The gain of the system is the same as its magnitude and is the ratio of the output magnitude to the input magnitude at each frequency. The phase of the system is the difference of the output phase and input phase at each frequency.

Cross Power Spectrum

The cross power spectrum is not typically used as a direct measurement but is an important building block for other measurements.

Use the following equation to compute the two-sided cross power spectrum of two time-domain signals *A* and *B*.

$$\text{Cross Power Spectrum } S_{AB}(f) = \frac{\text{FFT}(B) \times \text{FFT}^*(A)}{N^2}$$

The cross power spectrum is a two-sided complex form, having real and imaginary parts. To convert the cross power spectrum to magnitude and phase, use the rectangular-to-polar conversion function from Equation 4-13.

To convert the cross power spectrum to a single-sided form, use the methods and equations from the *Converting a Two-Sided Power Spectrum to a Single-Sided Power Spectrum* section of this chapter. The single-sided cross power spectrum yields the product of the rms amplitudes and the phase difference between the two signals *A* and *B*. The units of the single-sided cross power spectrum are in quantity rms squared, for example, V^2_{rms}.

The power spectrum is equivalent to the cross power spectrum when signals *A* and *B* are the same signal. Therefore, the power spectrum is often referred to as the auto power spectrum or the auto spectrum.

Frequency Response and Network Analysis

You can use the following functions to characterize the frequency response of a network:

- Frequency response function
- Impulse response function
- Coherence function

Frequency Response Function

Figure 4-19 illustrates the method for measuring the frequency response of a network.

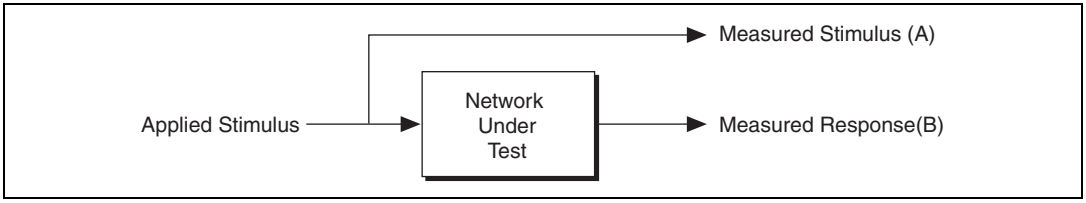


Figure 4-19. Configuration for Network Analysis

In Figure 4-19, you apply a stimulus to the network under test and measure the stimulus and response signals. From the measured stimulus and response signals, you compute the frequency response function. The frequency response function gives the gain and phase versus frequency of a network. You use Equation 4-14 to compute the response function.

$$H(f) = \frac{S_{AB}(f)}{S_{AA}(f)} \tag{4-14}$$

where $H(f)$ is the response function, A is the stimulus signal, B is the response signal, $S_{AB}(f)$ is the cross power spectrum of A and B, and $S_{AA}(f)$ is the power spectrum of A.

The frequency response function is a two-sided complex form, having real and imaginary parts. To convert to the frequency response gain and the frequency response phase, use the rectangular-to-polar conversion function from Equation 4-13. To convert to single-sided form, discard the second half of the response function array.

You might want to take several frequency response function readings and compute the average. Complete the following steps to compute the average frequency response function.

1. Compute the average $S_{AB}(f)$ by finding the sum in the complex form and dividing the sum by the number of measurements.
2. Compute the average $S_{AA}(f)$ by finding the sum and dividing the sum by the number of measurements.
3. Substitute the average $S_{AB}(f)$ and the average $S_{AA}(f)$ in Equation 4-14.

Impulse Response Function

The impulse response function of a network is the time-domain representation of the frequency response function of the network. The impulse response function is the output time-domain signal generated by applying an impulse to the network at time $t = 0$.

To compute the impulse response of the network, perform an inverse FFT on the two-sided complex frequency response function from Equation 4-14. To compute the average impulse response, perform an inverse FFT on the average frequency response function.

Coherence Function

The coherence function provides an indication of the quality of the frequency response function measurement and of how much of the response energy is correlated to the stimulus energy. If there is another signal present in the response, either from excessive noise or from another signal, the quality of the network response measurement is poor. You can use the coherence function to identify both excessive noise and which of the multiple signal sources are contributing to the response signal. Use Equation 4-15 to compute the coherence function.

$$\gamma^2(f) = \frac{(\text{Magnitude of the Average } S_{AB}(f))^2}{(\text{Average } S_{AA}(f))(\text{Average } S_{BB}(f))} \quad (4\text{-}15)$$

where S_{AB} is the cross power spectrum, S_{AA} is the power spectrum of A, and S_{BB} is the power spectrum of B.

Equation 4-15 yields a coherence factor with a value between zero and one versus frequency. A value of zero for a given frequency line indicates no correlation between the response and the stimulus signal. A value of one for a given frequency line indicates that 100% of the response energy is due to the stimulus signal and that no interference is occurring at that frequency.

For a valid result, the coherence function requires an average of two or more readings of the stimulus and response signals. For only one reading, the coherence function registers unity at all frequencies.

Windowing

In practical applications, you obtain only a finite number of samples of the signal. The FFT assumes that this time record repeats. If you have an integral number of cycles in your time record, the repetition is smooth at the boundaries. However, in practical applications, you usually have a nonintegral number of cycles. In the case of a nonintegral number of cycles, the repetition results in discontinuities at the boundaries. These artificial discontinuities were not originally present in your signal and result in a smearing or leakage of energy from your actual frequency to all other frequencies. This phenomenon is spectral leakage. The amount of leakage depends on the amplitude of the discontinuity, with a larger amplitude causing more leakage.

A signal that is exactly periodic in the time record is composed of sine waves with exact integral cycles within the time record. Such a perfectly periodic signal has a spectrum with energy contained in exact frequency bins.

A signal that is not periodic in the time record has a spectrum with energy split or spread across multiple frequency bins. The FFT spectrum models the time domain as if the time record repeated itself forever. It assumes that the analyzed record is just one period of an infinitely repeating periodic signal.

Because the amount of leakage is dependent on the amplitude of the discontinuity at the boundaries, you can use windowing to reduce the size of the discontinuity and reduce spectral leakage. Windowing consists of multiplying the time-domain signal by another time-domain waveform, known as a window, whose amplitude tapers gradually and smoothly towards zero at edges. The result is a windowed signal with very small or no discontinuities and therefore reduced spectral leakage. You can choose from among many different types of windows. The one you choose depends on your application and some prior knowledge of the signal you are analyzing.

Refer to Chapter 5, *Smoothing Windows*, for more information about windowing.

Averaging to Improve the Measurement

Averaging successive measurements usually improves measurement accuracy. Averaging usually is performed on measurement results or on individual spectra but not directly on the time record.

You can choose from among the following common averaging modes:

- RMS averaging
- Vector averaging
- Peak hold

RMS Averaging

RMS averaging reduces signal fluctuations but not the noise floor. The noise floor is not reduced because RMS averaging averages the energy, or power, of the signal. RMS averaging also causes averaged RMS quantities of single-channel measurements to have zero phase. RMS averaging for dual-channel measurements preserves important phase information. RMS-averaged measurements are computed according to the following equations.

FFT spectrum $\sqrt{\langle X^* \bullet X \rangle}$

power spectrum $\langle X^* \bullet X \rangle$

cross spectrum $\langle X^* \bullet Y \rangle$

frequency response $H1 = \frac{\langle X^* \bullet Y \rangle}{\langle X^* \bullet X \rangle}$

$$H2 = \langle \frac{Y^* \bullet Y}{Y^* \bullet X} \rangle$$

$$H3 = \frac{(H1 + H2)}{2}$$

where X is the complex FFT of signal *x* (stimulus), *Y* is the complex FFT of signal *y* (response), X^* is the complex conjugate of *X*, Y^* is the complex conjugate of *Y*, and $\langle X \rangle$ is the average of *X*, real and imaginary parts being averaged separately.

Vector Averaging

Vector averaging eliminates noise from synchronous signals. Vector averaging computes the average of complex quantities directly. The real part is averaged separately from the imaginary part. Averaging the real part separately from the imaginary part can reduce the noise floor for random signals because random signals are not phase coherent from one time record to the next. The real and imaginary parts are averaged separately, reducing noise but usually requiring a trigger.

FFT spectrum	$\langle X \rangle$
power spectrum	$\langle X^* \rangle \bullet \langle X \rangle$
cross spectrum	$\langle X^* \rangle \bullet \langle Y \rangle$
frequency response	$\frac{\langle Y \rangle}{\langle X \rangle}$ $(H1 = H2 = H3)$

where X is the complex FFT of signal *x* (stimulus), *Y* is the complex FFT of signal *y* (response), X^* is the complex conjugate of *X*, and $\langle X \rangle$ is the average of *X*, real and imaginary parts being averaged separately.

Peak Hold

Peak hold averaging retains the peak levels of the averaged quantities. Peak hold averaging is performed at each frequency line separately, retaining peak levels from one FFT record to the next.

FFT spectrum	$MAX\sqrt{(X^* \bullet X)}$
power spectrum	$MAX(X^* \bullet X)$

where X is the complex FFT of signal *x* (stimulus) and X^* is the complex conjugate of *X*.

Weighting

When performing RMS or vector averaging, you can weight each new spectral record using either linear or exponential weighting.

Linear weighting combines N spectral records with equal weighting. When the number of averages is completed, the analyzer stops averaging and presents the averaged results.

Exponential weighting emphasizes new spectral data more than old and is a continuous process.

Weighting is applied according to the following equation.

$$Y_i = \frac{N-1}{N} Y_{i-1} + \frac{1}{N} X_i,$$

where X_i is the result of the analysis performed on the i^{th} block, Y_i is the result of the averaging process from X_1 to X_i, $N = i$ for linear weighting, and N is a constant for exponential weighting ($N = 1$ for $i = 1$).

Echo Detection

Echo detection using Hilbert transforms is a common measurement for the analysis of modulation systems.

Equation 4-16 describes a time-domain signal. Equation 4-17 yields the Hilbert transform of the time-domain signal.

$$x(t) = Ae^{-t/\tau}\cos(2\pi f_0 t) \tag{4-16}$$

$$x_H(t) = -Ae^{-t/\tau}\sin(2\pi f_0 t) \tag{4-17}$$

where A is the amplitude, f_0 is the natural resonant frequency, and τ is the time decay constant.

Equation 4-18 yields the natural logarithm of the magnitude of the analytic signal $x_A(t)$.

$$\ln\left|x_A(t)\right| = \ln\left|x(t) + jx_H(t)\right| = -\frac{t}{\tau} + \ln A \tag{4-18}$$

The result from Equation 4-18 has the form of a line with slope $m = -\frac{1}{\tau}$. Therefore, you can extract the time constant of the system by graphing $\ln|x_A(t)|$.

Figure 4-20 shows a time-domain signal containing an echo signal.

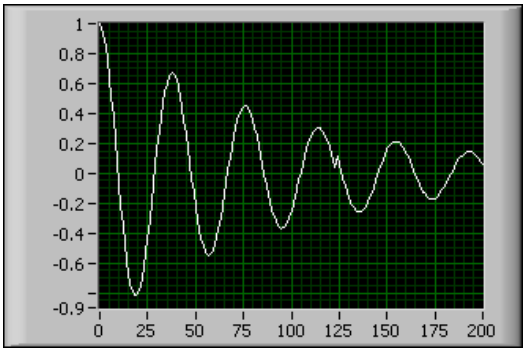


Figure 4-20. Echo Signal

The following conditions make the echo signal difficult to locate in Figure 4-20:

- The time delay between the source and the echo signal is short relative to the time decay constant of the system.
- The echo amplitude is small compared to the source.

You can make the echo signal visible by plotting the magnitude of $x_A(t)$ on a logarithmic scale, as shown in Figure 4-21.

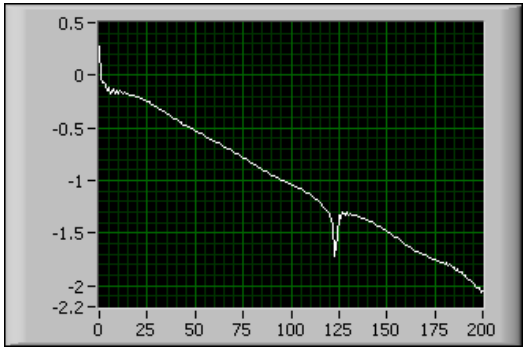


Figure 4-21. Echogram of the Magnitude of $x_A(t)$

In Figure 4-21, the discontinuity is plainly visible and indicates the location of the time delay of the echo.

Figure 4-22 shows a section of the block diagram of the VI used to produce Figures 4-20 and 4-21.

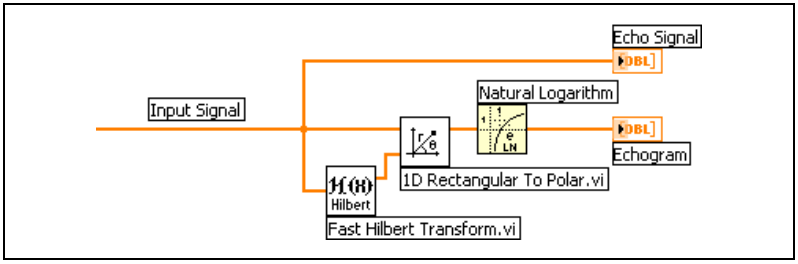


Figure 4-22. Echo Detector Block Diagram

The VI in Figure 4-22 completes the following steps to detect an echo.

1. Processes the input signal with the Fast Hilbert Transform VI to produce the analytic signal $x_A(t)$.
2. Computes the magnitude of $x_A(t)$ with the 1D Rectangular To Polar VI.
3. Computes the natural log of $x_A(t)$ to detect the presence of an echo.

5

Smoothing Windows

This chapter describes spectral leakage, how to use smoothing windows to decrease spectral leakage, the different types of smoothing windows, how to choose the correct type of smoothing window, the differences between smoothing windows used for spectral analysis and smoothing windows used for filter coefficient design, and the importance of scaling smoothing windows.

Applying a smoothing window to a signal is windowing. You can use windowing to complete the following analysis operations:

- Define the duration of the observation.
- Reduce spectral leakage.
- Separate a small amplitude signal from a larger amplitude signal with frequencies very close to each other.
- Design FIR filter coefficients.

The Windows VIs provide a simple method of improving the spectral characteristics of a sampled signal. Use the NI Example Finder to find examples of using the Windows VIs.

Spectral Leakage

According to the Shannon Sampling Theorem, you can completely reconstruct a continuous-time signal from discrete, equally spaced samples if the highest frequency in the time signal is less than half the sampling frequency. Half the sampling frequency equals the Nyquist frequency. The Shannon Sampling Theorem bridges the gap between continuous-time signals and digital-time signals. Refer to Chapter 1, *Introduction to Digital Signal Processing and Analysis in LabVIEW*, for more information about the Shannon Sampling Theorem.

In practical, signal-sampling applications, digitizing a time signal results in a finite record of the signal, even when you carefully observe the Shannon Sampling Theorem and sampling conditions. Even when the data meets the Nyquist criterion, the finite sampling record might cause energy leakage, called spectral leakage. Therefore, even though you use proper signal

acquisition techniques, the measurement might not result in a scaled, single-sided spectrum because of spectral leakage. In spectral leakage, the energy at one frequency appears to leak out into all other frequencies.

Spectral leakage results from an assumption in the FFT and DFT algorithms that the time record exactly repeats throughout all time. Thus, signals in a time record are periodic at intervals that correspond to the length of the time record. When you use the FFT or DFT to measure the frequency content of data, the transforms assume that the finite data set is one period of a periodic signal. Therefore, the finiteness of the sampling record results in a truncated waveform with different spectral characteristics from the original continuous-time signal, and the finiteness can introduce sharp transition changes into the measured data. The sharp transitions are discontinuities. Figure 5-1 illustrates discontinuities.

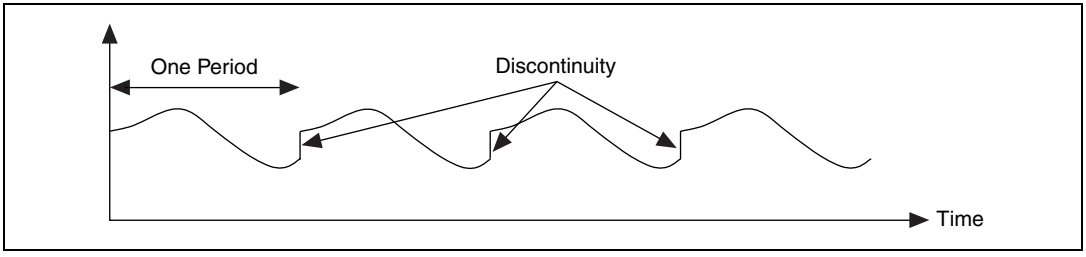


Figure 5-1. Periodic Waveform Created from Sampled Period

The discontinuities shown in Figure 5-1 produce leakage of spectral information. Spectral leakage produces a discrete-time spectrum that appears as a smeared version of the original continuous-time spectrum.

Sampling an Integer Number of Cycles

Spectral leakage occurs only when the sample data set consists of a noninteger number of cycles. Figure 5-2 shows a sine wave sampled at an integer number of cycles and the Fourier transform of the sine wave.

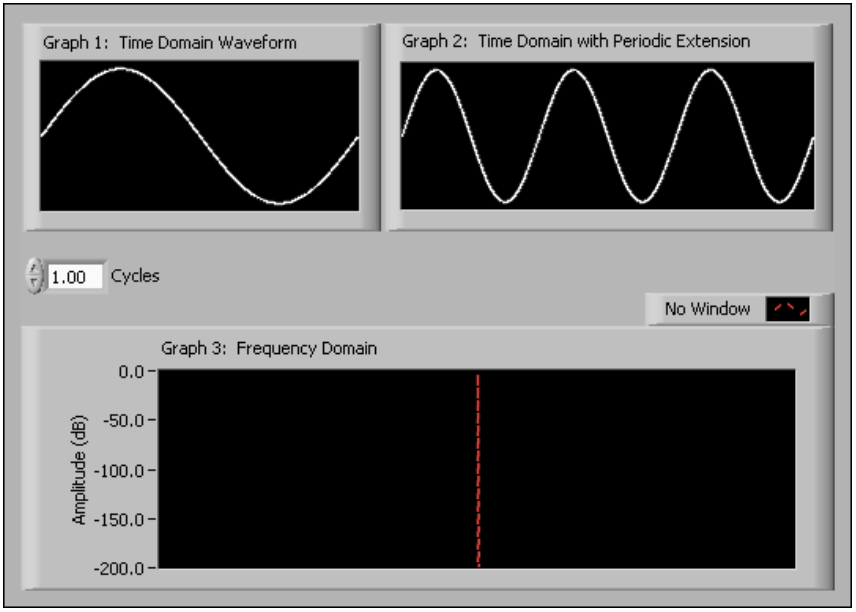


Figure 5-2. Sine Wave and Corresponding Fourier Transform

In Figure 5-2, Graph 1 shows the sampled time-domain waveform. Graph 2 shows the periodic time waveform of the sine wave from Graph 1. In Graph 2, the waveform repeats to fulfill the assumption of periodicity for the Fourier transform. Graph 3 shows the spectral representation of the waveform.

Because the time record in Graph 2 is periodic with no discontinuities, its spectrum appears in Graph 3 as a single line showing the frequency of the sine wave. The waveform in Graph 2 does not have any discontinuities because the data set is from an integer number of cycles—in this case, one.

The following methods are the only methods that guarantee you always acquire an integer number of cycles:

- Sample synchronously with respect to the signal you measure. Therefore, you can acquire an integral number of cycles deliberately.
- Capture a transient signal that fits entirely into the time record.

Sampling a Noninteger Number of Cycles

Usually, an unknown signal you are measuring is a stationary signal. A stationary signal is present before, during, and after data acquisition. When measuring a stationary signal, you cannot guarantee that you are

sampling an integer number of cycles. If the time record contains a noninteger number of cycles, spectral leakage occurs because the noninteger cycle frequency component of the signal does not correspond exactly to one of the spectrum frequency lines. Spectral leakage distorts the measurement in such a way that energy from a given frequency component appears to spread over adjacent frequency lines or bins, resulting in a smeared spectrum. You can use smoothing windows to minimize the effects of performing an FFT over a noninteger number of cycles.

Because of the assumption of periodicity of the waveform, artificial discontinuities between successive periods occur when you sample a noninteger number of cycles. The artificial discontinuities appear as very high frequencies in the spectrum of the signal—frequencies that are not present in the original signal. The high frequencies of the discontinuities can be much higher than the Nyquist frequency and alias somewhere between 0 and $f_s/2$. Therefore, spectral leakage occurs. The spectrum you obtain by using the DFT or FFT is a smeared version of the spectrum and is not the actual spectrum of the original signal.

Figure 5-3 shows a sine wave sampled at a noninteger number of cycles and the Fourier transform of the sine wave.

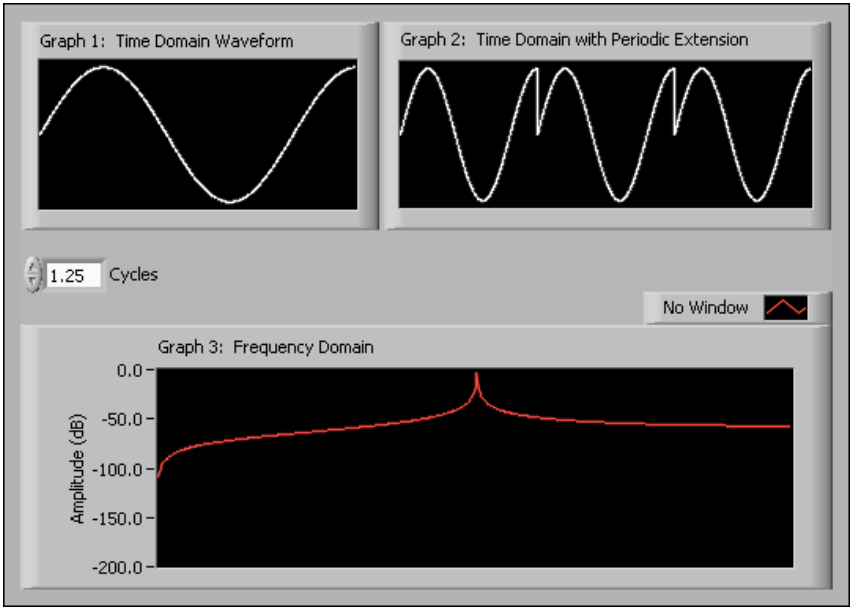


Figure 5-3. Spectral Representation When Sampling a Noninteger Number of Samples

In Figure 5-3, Graph 1 consists of 1.25 cycles of the sine wave. In Graph 2, the waveform repeats periodically to fulfill the assumption of periodicity for the Fourier transform. Graph 3 shows the spectral representation of the waveform. The energy is spread, or smeared, over a wide range of frequencies. The energy has leaked out of one of the FFT lines and smeared itself into all the other lines, causing spectral leakage.

Spectral leakage occurs because of the finite time record of the input signal. To overcome spectral leakage, you can take an infinite time record, from –infinity to +infinity. With an infinite time record, the FFT calculates one single line at the correct frequency. However, waiting for infinite time is not possible in practice. To overcome the limitations of a finite time record, windowing is used to reduce the spectral leakage.

In addition to causing amplitude accuracy errors, spectral leakage can obscure adjacent frequency peaks. Figure 5-4 shows the spectrum for two close frequency components when no smoothing window is used and when a Hanning window is used.

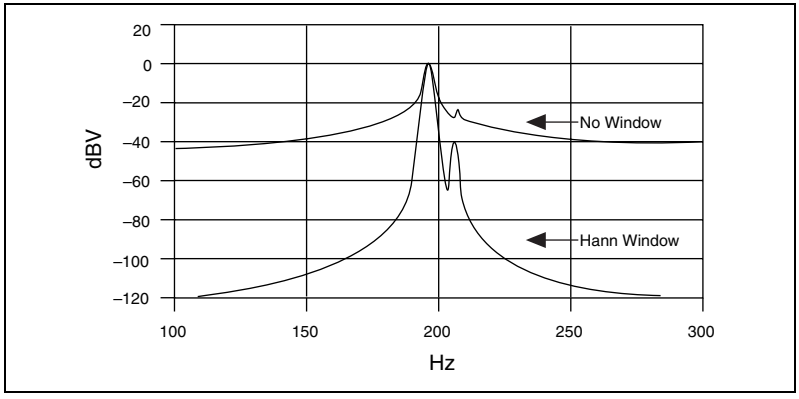


Figure 5-4. Spectral Leakage Obscuring Adjacent Frequency Components

In Figure 5-4, the second peak stands out more prominently in the windowed signal than it does in the signal with no smoothing window applied.

Windowing Signals

Use smoothing windows to improve the spectral characteristics of a sampled signal. When performing Fourier or spectral analysis on finite-length data, you can use smoothing windows to minimize the discontinuities of truncated waveforms, thus reducing spectral leakage.

The amount of spectral leakage depends on the amplitude of the discontinuity. As the discontinuity becomes larger, spectral leakage increases, and vice versa. Smoothing windows reduce the amplitude of the discontinuities at the boundaries of each period and act like predefined, narrowband, lowpass filters.

The process of windowing a signal involves multiplying the time record by a smoothing window of finite length whose amplitude varies smoothly and gradually towards zero at the edges. The length, or time interval, of a smoothing window is defined in terms of number of samples. Multiplication in the time domain is equivalent to convolution in the frequency domain. Therefore, the spectrum of the windowed signal is a convolution of the spectrum of the original signal with the spectrum of the smoothing window. Windowing changes the shape of the signal in the time domain, as well as affecting the spectrum that you see.

Figure 5-5 illustrates convolving the original spectrum of a signal with the spectrum of a smoothing window.

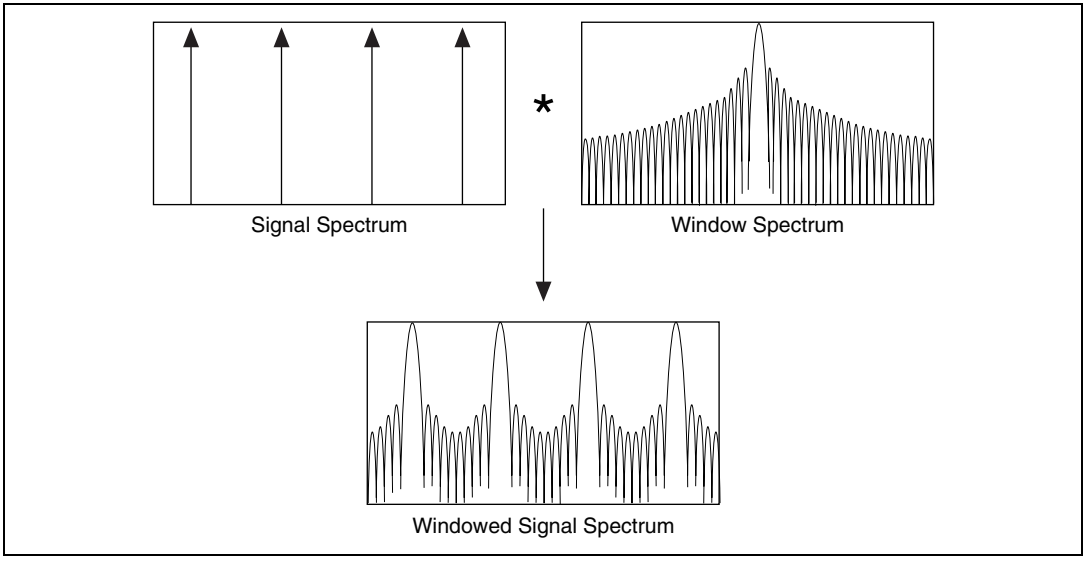


Figure 5-5. Frequency Characteristics of a Windowed Spectrum

Even if you do not apply a smoothing window to a signal, a windowing effect still occurs. The acquisition of a finite time record of an input signal produces the effect of multiplying the signal in the time domain by a uniform window. The uniform window has a rectangular shape and uniform height. The multiplication of the input signal in the time domain by the uniform window is equivalent to convolving the spectrum of the signal with

the spectrum of the uniform window in the frequency domain, which has a sinc function characteristic.

Figure 5-6 shows the result of applying a Hamming window to a time-domain signal.

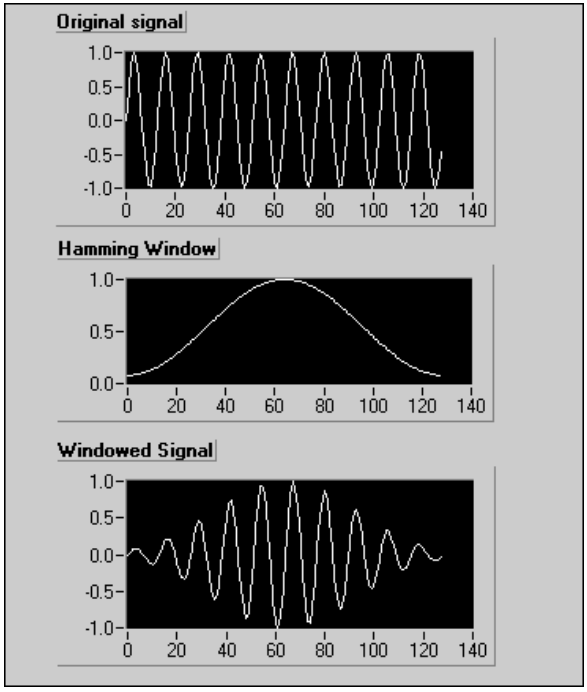


Figure 5-6. Time Signal Windowed Using a Hamming Window

In Figure 5-6, the time waveform of the windowed signal gradually tapers to zero at the ends because the Hamming window minimizes the discontinuities along the transition edges of the waveform. Applying a smoothing window to time-domain data before the transform of the data into the frequency domain minimizes spectral leakage.

Figure 5-7 shows the effects of the following smoothing windows on a signal:

- None (uniform)
- Hanning
- Flat top

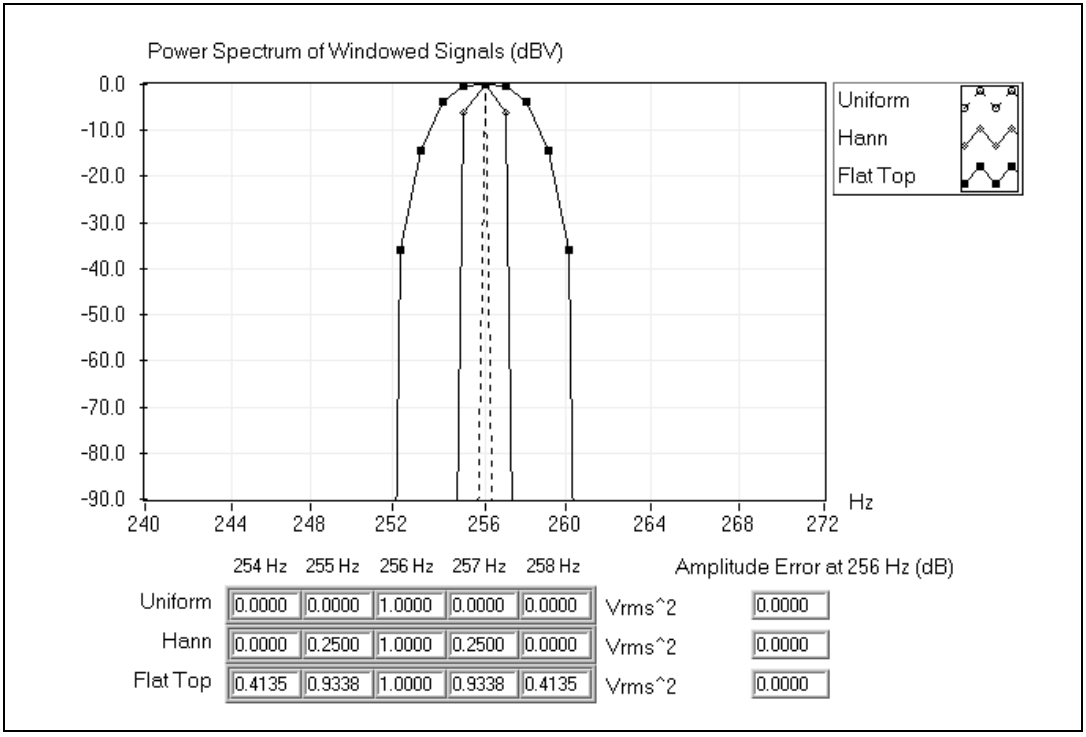


Figure 5-7. Power Spectrum of 1 V_{rms} Signal at 256 Hz with Uniform, Hanning, and Flat Top Windows

The data set for the signal in Figure 5-7 consists of an integer number of cycles, 256, in a 1,024-point record. If the frequency components of the original signal match a frequency line exactly, as is the case when you acquire an integer number of cycles, you see only the main lobe of the spectrum. The smoothing windows have a main lobe around the frequency of interest. The main lobe is a frequency-domain characteristic of windows. The uniform window has the narrowest lobe. The Hanning and flat top windows introduce some spreading. The flat top window has a broader main lobe than the uniform or Hanning windows. For an integer number of cycles, all smoothing windows yield the same peak amplitude reading and have excellent amplitude accuracy. Side lobes do not appear because the spectrum of the smoothing window approaches zero at Δf intervals on either side of the main lobe.

Figure 5-7 also shows the values at frequency lines of 254 Hz through 258 Hz for each smoothing window. The amplitude error at 256 Hz equals 0 dB for each smoothing window. The graph shows the spectrum values between 240 Hz and 272 Hz. The actual values in the resulting spectrum

array for each smoothing window at 254 Hz through 258 Hz are shown below the graph. Δf equals 1 Hz.

If a time record does not contain an integer number of cycles, the continuous spectrum of the smoothing window shifts from the main lobe center at a fraction of Δf that corresponds to the difference between the frequency component and the FFT line frequencies. This shift causes the side lobes to appear in the spectrum. In addition, amplitude error occurs at the frequency peak because sampling of the main lobe is off center and smears the spectrum. Figure 5-8 shows the effect of spectral leakage on a signal whose data set consists of 256.5 cycles.

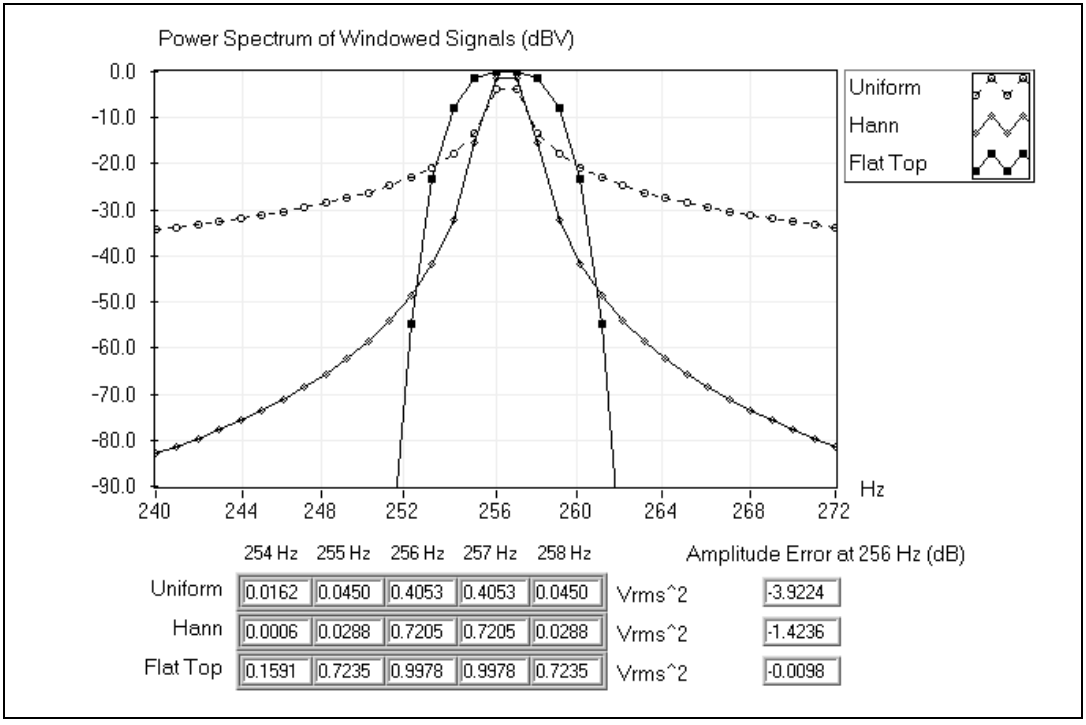


Figure 5-8. Power Spectrum of 1 V_{rms} Signal at 256.5 Hz with Uniform, Hanning, and Flat Top Windows

In Figure 5-8, for a noninteger number of cycles, the Hanning and flat top windows introduce much less spectral leakage than the uniform window. Also, the amplitude error is better with the Hanning and flat top windows. The flat top window demonstrates very good amplitude accuracy and has a wider spread and higher side lobes than the Hanning window.

Figure 5-9 shows the block diagram of a VI that measures the windowed and nonwindowed spectrums of a signal composed of the sum of two sinusoids.

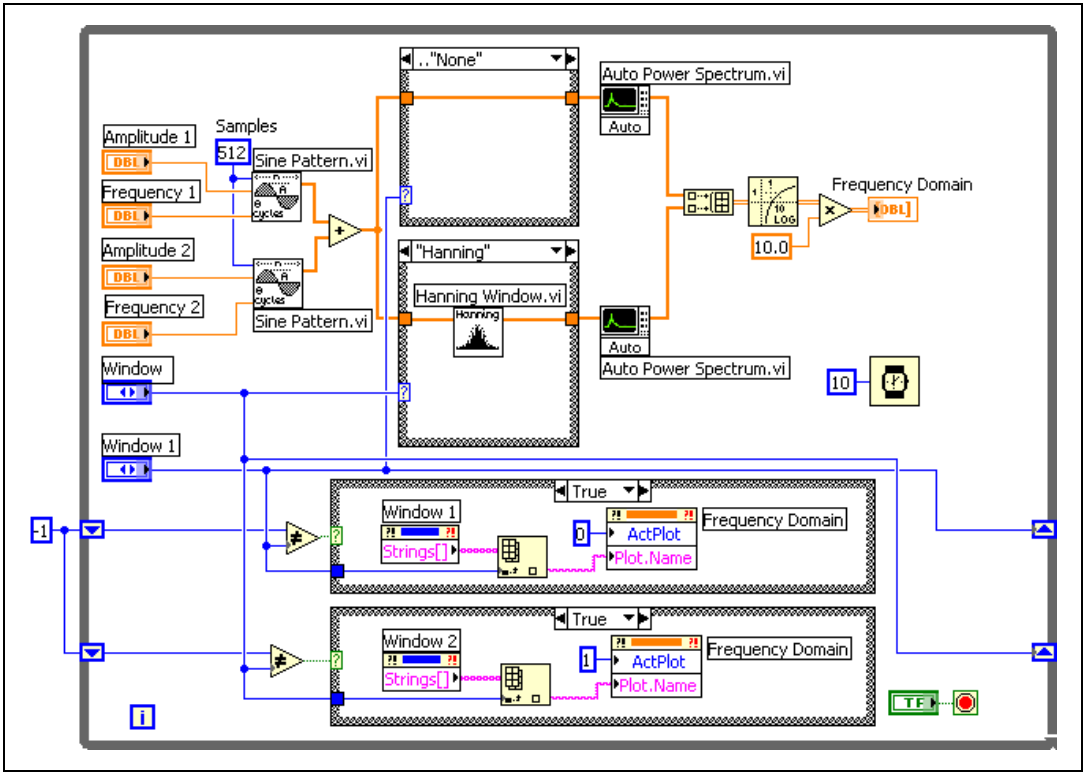


Figure 5-9. Measuring the Spectrum of a Signal Composed of the Sum of Two Sinusoids

Figure 5-10 shows the amplitudes and frequencies of the two sinusoids and the measurement results. The frequencies shown are in units of cycles.

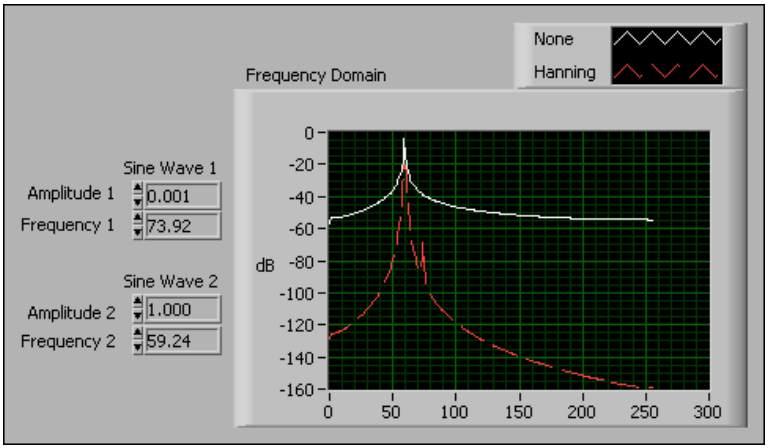


Figure 5-10. Windowed and Nonwindowed Spectrums of the Sum of Two Sinusoids

In Figure 5-10, the nonwindowed spectrum shows leakage that is more than 20 dB at the frequency of the smaller sinusoid.

You can apply more sophisticated techniques to get a more accurate description of the original time-continuous signal in the frequency domain. However, in most applications, applying a smoothing window is sufficient to obtain a better frequency representation of the signal.

Characteristics of Different Smoothing Windows

To simplify choosing a smoothing window, you need to define various characteristics so that you can make comparisons between smoothing windows. An actual plot of a smoothing window shows that the frequency characteristic of the smoothing window is a continuous spectrum with a main lobe and several side lobes. Figure 5-11 shows the spectrum of a typical smoothing window.

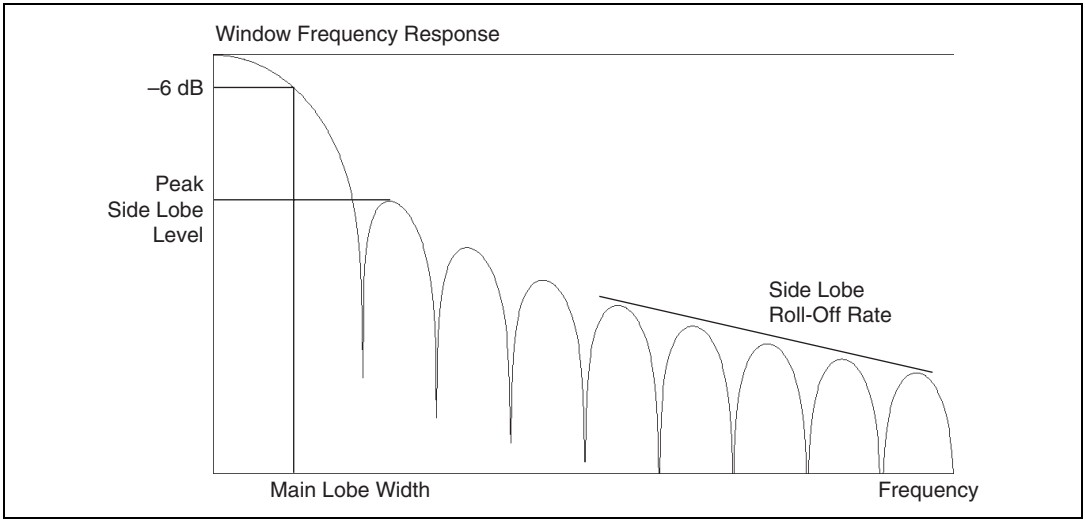


Figure 5-11. Frequency Response of a Smoothing Window

Main Lobe

The center of the main lobe of a smoothing window occurs at each frequency component of the time-domain signal. By convention, to characterize the shape of the main lobe, the widths of the main lobe at –3 dB and –6 dB below the main lobe peak describe the width of the main lobe. The unit of measure for the main lobe width is FFT bins or frequency lines.

The width of the main lobe of the smoothing window spectrum limits the frequency resolution of the windowed signal. Therefore, the ability to distinguish two closely spaced frequency components increases as the main lobe of the smoothing window narrows. As the main lobe narrows and spectral resolution improves, the window energy spreads into its side lobes, increasing spectral leakage and decreasing amplitude accuracy. A trade-off occurs between amplitude accuracy and spectral resolution.

Side Lobes

Side lobes occur on each side of the main lobe and approach zero at multiples of f_s/N from the main lobe. The side lobe characteristics of the smoothing window directly affect the extent to which adjacent frequency components leak into adjacent frequency bins. The side lobe response of a strong sinusoidal signal can overpower the main lobe response of a nearby weak sinusoidal signal.

Maximum side lobe level and side lobe roll-off rate characterize the side lobes of a smoothing window. The maximum side lobe level is the largest side lobe level in decibels relative to the main lobe peak gain. The side lobe roll-off rate is the asymptotic decay rate in decibels per decade of frequency of the peaks of the side lobes. Table 5-1 lists the characteristics of several smoothing windows.

Table 5-1. Characteristics of Smoothing Windows

Smoothing Window	–3 dB Main Lobe Width (bins)	–6 dB Main Lobe Width (bins)	Maximum Side Lobe Level (dB)	Side Lobe Roll-Off Rate (dB/decade)
Uniform (none)	0.88	1.21	–13	20
Hanning	1.44	2.00	–32	60
Hamming	1.30	1.81	–43	20
Blackman-Harris	1.62	2.27	–71	20
Exact Blackman	1.61	2.25	–67	20
Blackman	1.64	2.30	–58	60
Flat Top	2.94	3.56	–44	20

Rectangular (None)

The rectangular window has a value of one over its length. The following equation defines the rectangular window.

$$w(n) = 1.0 \qquad \text{for } n = 0, 1, 2, \ldots, N - 1$$

where N is the length of the window and w is the window value.

Applying a rectangular window is equivalent to not using any window because the rectangular function just truncates the signal to within a finite time interval. The rectangular window has the highest amount of spectral leakage.

Figure 5-12 shows the rectangular window for $N = 32$.

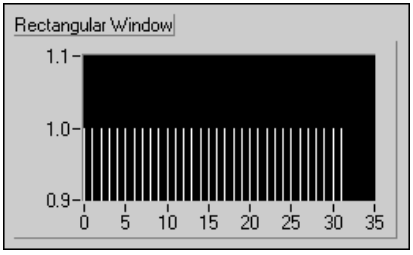


Figure 5-12. Rectangular Window

The rectangular window is useful for analyzing transients that have a duration shorter than that of the window. Transients are signals that exist only for a short time duration. The rectangular window also is used in order tracking, where the effective sampling rate is proportional to the speed of the shaft in rotating machines. In order tracking, the rectangular window detects the main mode of vibration of the machine and its harmonics.

Hanning

The Hanning window has a shape similar to that of half a cycle of a cosine wave. The following equation defines the Hanning window.

$$w(n) = 0.5 - 0.5\cos\frac{2\pi n}{N} \qquad \text{for } n = 0, 1, 2, \ldots, N-1$$

where N is the length of the window and w is the window value.

Figure 5-13 shows a Hanning window with $N = 32$.

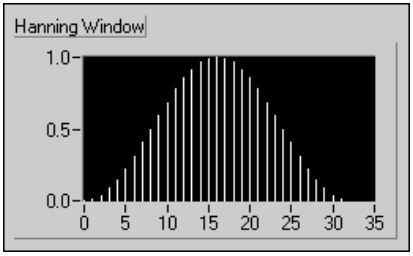


Figure 5-13. Hanning Window

The Hanning window is useful for analyzing transients longer than the time duration of the window and for general-purpose applications.

Hamming

The Hamming window is a modified version of the Hanning window. The shape of the Hamming window is similar to that of a cosine wave. The following equation defines the Hamming window.

$$w(n) = 0.54 - 0.46\cos\frac{2\pi n}{N} \qquad \text{for } n = 0, 1, 2, \ldots, N-1$$

where N is the length of the window and w is the window value.

Figure 5-14 shows a Hamming window with $N = 32$.

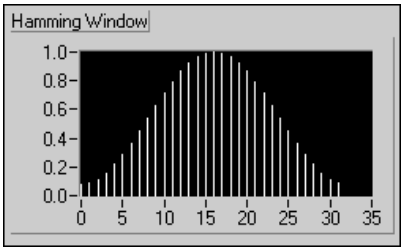


Figure 5-14. Hamming Window

The Hanning and Hamming windows are similar, as shown in Figures 5-13 and 5-14. However, in the time domain, the Hamming window does not get as close to zero near the edges as does the Hanning window.

Kaiser-Bessel

The Kaiser-Bessel window is a flexible smoothing window whose shape you can modify by adjusting the **beta** input. Thus, depending on your application, you can change the shape of the window to control the amount of spectral leakage.

Figure 5-15 shows the Kaiser-Bessel window for different values of **beta**.

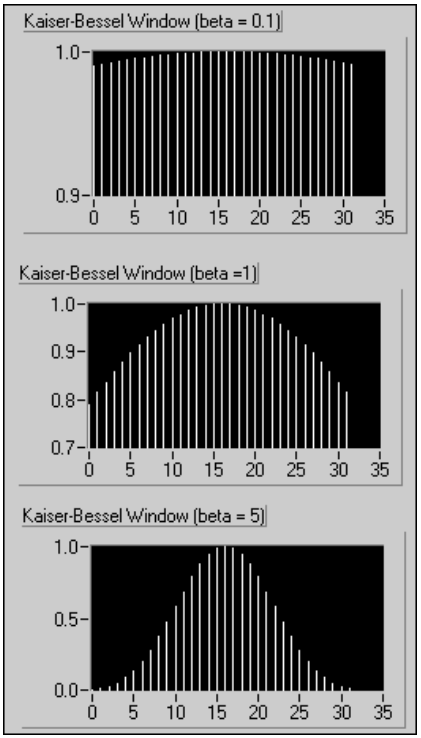


Figure 5-15. Kaiser-Bessel Window

For small values of **beta**, the shape is close to that of a rectangular window. Actually, for **beta** = 0.0, you do get a rectangular window. As you increase **beta**, the window tapers off more to the sides.

The Kaiser-Bessel window is useful for detecting two signals of almost the same frequency but with significantly different amplitudes.

Triangle

The shape of the triangle window is that of a triangle. The following equation defines the triangle window.

$$w(n) = 1 - \left|\frac{2n - N}{N}\right| \qquad \text{for } n = 0, 1, 2, \ldots, N - 1$$

where N is the length of the window and w is the window value.

Figure 5-16 shows a triangle window for $N = 32$.

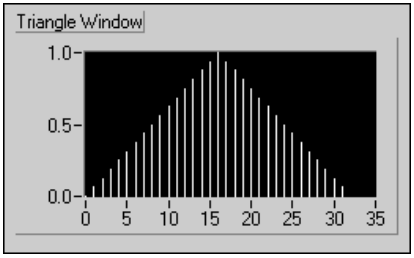


Figure 5-16. Triangle Window

Flat Top

The flat top window has the best amplitude accuracy of all the smoothing windows at ±0.02 dB for signals exactly between integral cycles. Because the flat top window has a wide main lobe, it has poor frequency resolution. The following equation defines the flat top window.

$$w(n) = \sum_{k=0}^{4} (-1)^k a_k \cos(k\omega)$$

where $\omega = \frac{2\pi n}{N}$

$a_0 = 0.215578948$

$a_1 = 0.416631580$

$a_2 = 0.277263158$

$a_3 = 0.083578947$

$a_4 = 0.006947368$

Figure 5-17 shows a flat top window.

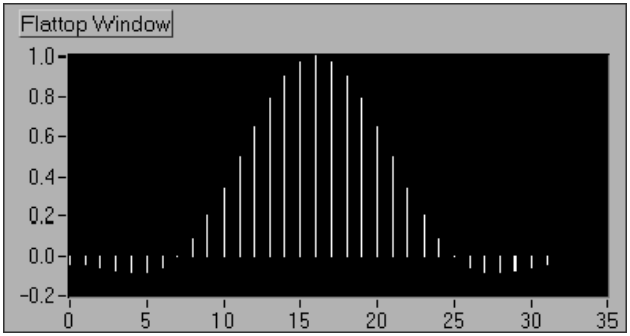


Figure 5-17. Flat Top Window

The flat top window is most useful in accurately measuring the amplitude of single frequency components with little nearby spectral energy in the signal.

Exponential

The shape of the exponential window is that of a decaying exponential. The following equation defines the exponential window.

$$w[n] = e^{\left(\frac{n \ln(f)}{N-1}\right)} = f^{\left(\frac{n}{N-1}\right)} \quad \text{for } n = 0, 1, 2, \ldots, N-1$$

where N is the length of the window, w is the window value, and f is the final value.

The initial value of the window is one and gradually decays toward zero. You can adjust the final value of the exponential window to between 0 and 1.

Figure 5-18 shows the exponential window for $N = 32$, with the final value specified as 0.1.

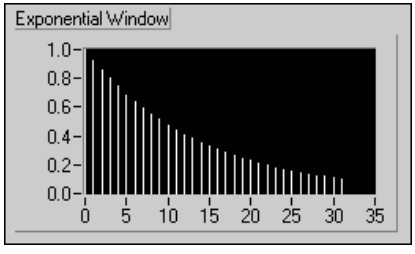


Figure 5-18. Exponential Window

The exponential window is useful for analyzing transient response signals whose duration is longer than the length of the window. The exponential window damps the end of the signal, ensuring that the signal fully decays by the end of the sample block. You can apply the exponential window to signals that decay exponentially, such as the response of structures with light damping that are excited by an impact, such as the impact of a hammer.

Windows for Spectral Analysis versus Windows for Coefficient Design

Spectral analysis and filter coefficient design place different requirements on a window. Spectral analysis requires a DFT-even window, while filter coefficient design requires a window symmetric about its midpoint.

Spectral Analysis

The smoothing windows designed for spectral analysis must be DFT even. A smoothing window is DFT even if its dot product, or inner product, with integral cycles of sine sequences is identically zero. In other words, the DFT of a DFT-even sequence has no imaginary component.

Figures 5-19 and 5-20 show the Hanning window for a sample size of 8 and one cycle of a sine pattern for a sample size of 8.

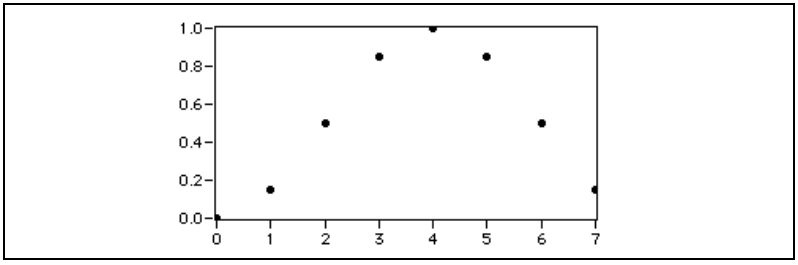


Figure 5-19. Hanning Window for Sample Size 8

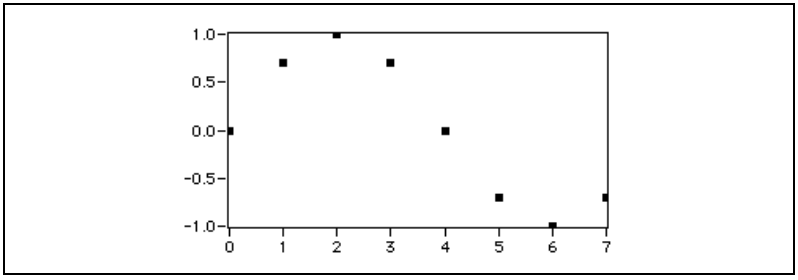


Figure 5-20. Sine Pattern for Sample Size 8

In Figure 5-19, the DFT-even Hanning window is not symmetric about its midpoint. The last point of the window is not equal to its first point, similar to one complete cycle of the sine pattern shown in Figure 5-20.

Smoothing windows for spectral analysis are spectral windows and include the following window types:

- Scaled time-domain window
- Hanning window
- Hamming window
- Triangle window
- Blackman window
- Exact Blackman window
- Blackman-Harris window
- Flat top window
- Kaiser-Bessel window
- General cosine window
- Cosine tapered window

Windows for FIR Filter Coefficient Design

Designing FIR filter coefficients requires a window that is symmetric about its midpoint.

Equations 5-1 and 5-2 illustrate the difference between a spectral window and a symmetrical window for filter coefficient design.

Equation 5-1 defines the Hanning window for spectral analysis.

$$w[i] = 0.5\left(1 - \cos\left(\frac{2\pi i}{N}\right)\right) \qquad \text{for } i = 0, 1, 2, \ldots, N-1 \qquad (5\text{-}1)$$

where N is the length of the window and w is the window value.

Equation 5-2 defines a symmetrical Hanning window for filter coefficient design.

$$w[i] = 0.5\left(1 - \cos\left(\frac{2\pi i}{N-1}\right)\right) \qquad \text{for } i = 0, 1, 2, \ldots, N-1 \qquad (5\text{-}2)$$

where N is the length of the window and w is the window value.

By modifying a spectral window, as shown in Equation 5-2, you can define a symmetrical window for designing filter coefficients. Refer to Chapter 3, *Digital Filtering*, for more information about designing digital filters.

Choosing the Correct Smoothing Window

Selecting a smoothing window is not a simple task. Each smoothing window has its own characteristics and suitability for different applications. To choose a smoothing window, you must estimate the frequency content of the signal. If the signal contains strong interfering frequency components distant from the frequency of interest, choose a smoothing window with a high side lobe roll-off rate. If the signal contains strong interfering signals near the frequency of interest, choose a smoothing window with a low maximum side lobe level. Refer to Table 5-1 for information about side lobe roll-off rates and maximum side lobe levels for various smoothing windows.

If the frequency of interest contains two or more signals very near to each other, spectral resolution is important. In this case, it is best to choose a smoothing window with a very narrow main lobe. If the amplitude accuracy of a single frequency component is more important than the exact location

of the component in a given frequency bin, choose a smoothing window with a wide main lobe. If the signal spectrum is rather flat or broadband in frequency content, use the uniform window, or no window. In general, the Hanning window is satisfactory in 95% of cases. It has good frequency resolution and reduced spectral leakage. If you do not know the nature of the signal but you want to apply a smoothing window, start with the Hanning window.

Table 5-2 lists different types of signals and the appropriate windows that you can use with them.

Table 5-2. Signals and Windows

Type of Signal	Window
Transients whose duration is shorter than the length of the window	Rectangular
Transients whose duration is longer than the length of the window	Exponential, Hanning
General-purpose applications	Hanning
Spectral analysis (frequency-response measurements)	Hanning (for random excitation), Rectangular (for pseudorandom excitation)
Separation of two tones with frequencies very close to each other but with widely differing amplitudes	Kaiser-Bessel
Separation of two tones with frequencies very close to each other but with almost equal amplitudes	Rectangular
Accurate single-tone amplitude measurements	Flat top
Sine wave or combination of sine waves	Hanning
Sine wave and amplitude accuracy is important	Flat top
Narrowband random signal (vibration data)	Hanning
Broadband random (white noise)	Uniform
Closely spaced sine waves	Uniform, Hamming
Excitation signals (hammer blow)	Force
Response signals	Exponential
Unknown content	Hanning

Initially, you might not have enough information about the signal to select the most appropriate smoothing window for the signal. You might need to experiment with different smoothing windows to find the best one. Always compare the performance of different smoothing windows to find the best one for the application.

Scaling Smoothing Windows

Applying a smoothing window to a time-domain signal multiplies the time-domain signal by the length of the smoothing window and introduces distortion effects due to the smoothing window. The smoothing window changes the overall amplitude of the signal. When applying multiple smoothing windows to the same signal, scaling each smoothing window by dividing the windowed array by the coherent gain of the window results in each window yielding the same spectrum amplitude result within the accuracy constraints of the window. The plots in Figures 5-7 and 5-8 are the result of applying scaled smoothing windows to the time-domain signal.

An FFT is equivalent to a set of parallel filters with each filter having a bandwidth equal to Δf. Because of the spreading effect of a smoothing window, the smoothing window increases the effective bandwidth of an FFT bin by an amount known as the equivalent noise-power bandwidth (ENBW) of the smoothing window. The power of a given frequency peak equals the sum of the adjacent frequency bins around the peak increased by a scaling factor equal to the ENBW of the smoothing window. You must take the scaling factor into account when you perform computations based on the power spectrum. Refer to Chapter 4, *Frequency Analysis*, for information about performing computations on the power spectrum.

Table 5-3 lists the scaling factor, also known as coherent gain, the ENBW, and the worst-case peak amplitude accuracy caused by off-center components for several popular smoothing windows.

Table 5-3. Correction Factors and Worst-Case Amplitude Errors for Smoothing Windows

Window	Scaling Factor (Coherent Gain)	ENBW	Worst-Case Amplitude Error (dB)
Uniform (none)	1.00	1.00	3.92
Hanning	0.50	1.50	1.42
Hamming	0.54	1.36	1.75

Table 5-3. Correction Factors and Worst-Case Amplitude Errors for Smoothing Windows (Continued)

Window	Scaling Factor (Coherent Gain)	ENBW	Worst-Case Amplitude Error (dB)
Blackman-Harris	0.42	1.71	1.13
Exact Blackman	0.43	1.69	1.15
Blackman	0.42	1.73	1.10
Flat Top	0.22	3.77	<0.01

6

Distortion Measurements

This chapter describes harmonic distortion, total harmonic distortion (THD), signal noise and distortion (SINAD), and when to use distortion measurements.

Defining Distortion

Applying a pure single-frequency sine wave to a perfectly linear system produces an output signal having the same frequency as that of the input sine wave. However, the output signal might have a different amplitude and/or phase than the input sine wave. Also, when you apply a composite signal consisting of several sine waves at the input, the output signal consists of the same frequencies but different amplitudes and/or phases.

Many real-world systems act as nonlinear systems when their input limits are exceeded, resulting in distorted output signals. If the input limits of a system are exceeded, the output consists of one or more frequencies that did not originally exist at the input. For example, if the input to a nonlinear system consists of two frequencies f_1 and f_2, the frequencies at the output might have the following components:

- f_1 and harmonics, or integer multiples, of f_1
- f_2 and harmonics of f_2
- Sums and differences of f_1, f_2
- Harmonics of f_1 and f_2

The number of new frequencies at the output, their corresponding amplitudes, and their relationships with respect to the original frequencies vary depending on the transfer function. Distortion measurements quantify the degree of nonlinearity of a system. Common distortion measurements include the following measurements:

- Total harmonic distortion (THD)
- Total harmonic distortion + noise (THD + N)
- Signal noise and distortion (SINAD)
- Intermodulation distortion

Application Areas

You can make distortion measurements for many devices, such as A/D and D/A converters, audio processing devices, analog tape recorders, cellular phones, radios, televisions, stereos, and loudspeakers.

Measurements of harmonics often provide a good indication of the cause of the nonlinearity of a system. For example, nonlinearities that are asymmetrical around zero produce mainly even harmonics. Nonlinearities symmetrical around zero produce mainly odd harmonics. You can use distortion measurements to diagnose faults such as bad solder joints, torn speaker cones, and incorrectly installed components.

However, nonlinearities are not always undesirable. For example, many musical sounds are produced specifically by driving a device into its nonlinear region.

Harmonic Distortion

When a signal $x(t)$ of a particular frequency f_1 passes through a nonlinear system, the output of the system consists of f_1 and its harmonics. The following expression describes the relationship between f_1 and its harmonics.

$$f_1, f_2 = 2f_1, f_3 = 3f_1, f_4 = 4f_1, \ldots, f_n = nf_1$$

The degree of nonlinearity of the system determines the number of harmonics and their corresponding amplitudes the system generates. In general, as the nonlinearity of a system increases, the harmonics become higher. As the nonlinearity of a system decreases, the harmonics become lower.

Figure 6-1 illustrates an example of a nonlinear system where the output $y(t)$ is the cube of the input signal $x(t)$.

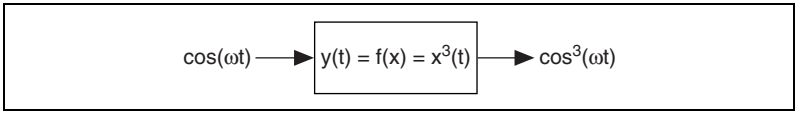


Figure 6-1. Example of a Nonlinear System

The following equation defines the input for the system shown in Figure 6-1.

$$x(t) = \cos(\omega t)$$

Equation 6-1 defines the output of the system shown in Figure 6-1.

$$x^3(t) = 0.5\cos(\omega t) + 0.25[\cos(\omega t) + \cos(3\omega t)] \quad (6\text{-}1)$$

In Equation 6-1, the output contains not only the input fundamental frequency ω but also the third harmonic 3ω.

A common cause of harmonic distortion is clipping. Clipping occurs when a system is driven beyond its capabilities. Symmetrical clipping results in odd harmonics. Asymmetrical clipping creates both even and odd harmonics.

THD

To determine the total amount of nonlinear distortion, also known as total harmonic distortion (THD), a system introduces, measure the amplitudes of the harmonics the system introduces relative to the amplitude of the fundamental frequency. The following equation yields THD.

$$\text{THD} = \frac{\sqrt{A_2^2 + A_3^2 + A_4^2 + \ldots}}{A_1}$$

where A_1 is the amplitude of the fundamental frequency, A_2 is the amplitude of the second harmonic, A_3 is the amplitude of the third harmonic, A_4 is the amplitude of the fourth harmonic, and so on.

You usually report the results of a THD measurement in terms of the highest order harmonic present in the measurement, such as THD through the seventh harmonic.

The following equation yields the percentage total harmonic distortion (%THD).

$$\%\text{THD} = (100)\left(\frac{\sqrt{A_2^2 + A_3^2 + A_4^2 + \ldots}}{A_1}\right)$$

THD + N

Real-world signals usually contain noise. A system can introduce additional noise into the signal. THD + N measures signal distortion while taking into account the amount of noise power present in the signal. Measuring THD + N requires measuring the amplitude of the fundamental frequency and the power present in the remaining signal after removing the fundamental frequency. The following equation yields THD + N.

$$\text{THD + N} = \frac{\sqrt{A_2^2 + A_3^2 + \ldots + N^2}}{\sqrt{A_1^2 + A_2^2 + A_3^2 + \ldots N^2}}$$

where *N* is the noise power.

A low THD + N measurement means that the system has a low amount of harmonic distortion and a low amount of noise from interfering signals, such as AC mains hum and wideband white noise.

As with THD, you usually report the results of a THD + N measurement in terms of the highest order harmonic present in the measurement, such as THD + N through the third harmonic.

The following equation yields percentage total harmonic distortion + noise (%THD + N).

$$\%\text{THD + N} = (100)\left(\frac{\sqrt{A_2^2 + A_3^2 + \ldots + N^2}}{\sqrt{A_1^2 + A_2^2 + A_3^2 + \ldots N^2}}\right)$$

SINAD

Similar to THD + N, SINAD takes into account both harmonics and noise. However, SINAD is the reciprocal of THD + N. The following equation yields SINAD.

$$\text{SINAD} = \frac{Fundamental + Noise + Distortion}{Noise + Distortion}$$

You can use SINAD to characterize the performance of FM receivers in terms of sensitivity, adjacent channel selectivity, and alternate channel selectivity.

7

DC/RMS Measurements

Two of the most common measurements of a signal are its direct current (DC) and root mean square (RMS) levels. This chapter introduces measurement analysis techniques for making DC and RMS measurements of a signal.

What Is the DC Level of a Signal?

You can use DC measurements to define the value of a static or slowly varying signal. DC measurements can be both positive and negative. The DC value usually is constant within a specific time window. You can track and plot slowly moving values, such as temperature, as a function of time using a DC meter. In that case, the observation time that results in the measured value has to be short compared to the speed of change for the signal. Figure 7-1 illustrates an example DC level of a signal.

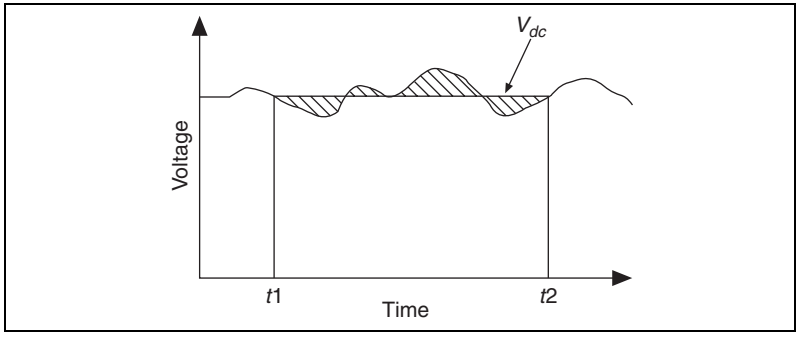


Figure 7-1. DC Level of a Signal

The DC level of a continuous signal $V(t)$ from time $t1$ to time $t2$ is given by the following equation.

$$V_{dc} = \frac{1}{(t2 - t1)} \cdot \int_{t1}^{t2} V(t)dt$$

where $t2 - t1$ is the integration time or measurement time.

For digitized signals, the discrete-time version of the previous equation is given by the following equation.

$$V_{dc} = \frac{1}{N} \cdot \sum_{i=1}^{N} V_i$$

For a sampled system, the DC value is defined as the mean value of the samples acquired in the specified measurement time window.

Between pure DC signals and fast-moving dynamic signals is a *gray zone* where signals become more complex, and measuring the DC level of these signals becomes challenging.

Real-world signals often contain a significant amount of dynamic influence. Often, you do not want the dynamic part of the signal. The DC measurement identifies the static DC signal hidden in the dynamic signal, for example, the voltage generated by a thermocouple in an industrial environment, where external noise or hum from the main power can disturb the DC signal significantly.

What Is the RMS Level of a Signal?

The RMS level of a signal is the square root of the mean value of the squared signal. RMS measurements are always positive. Use RMS measurements when a representation of energy is needed. You usually acquire RMS measurements on dynamic signals—signals with relatively fast changes—such as noise or periodic signals. Refer to Chapter 7, *Measuring AC Voltage*, of the *LabVIEW Measurements Manual* for more information about when to use RMS measurements.

The RMS level of a continuous signal $V(t)$ from time $t1$ to time $t2$ is given by the following equation.

$$V_{rms} = \sqrt{\frac{1}{(t2 - t1)} \cdot \int_{t1}^{t2} V^2(t)dt}$$

where $t2 - t1$ is the integration time or measurement time.

The RMS level of a discrete signal V_i is given by the following equation.

$$V_{rms} = \sqrt{\frac{1}{N} \cdot \sum_{i=1}^{N} V_i^2}$$

One difficulty is encountered when measuring the dynamic part of a signal using an instrument that does not offer an AC-coupling option. A true RMS measurement includes the DC part in the measurement, which is a measurement you might not want.

Averaging to Improve the Measurement

Instantaneous DC measurements of a noisy signal can vary randomly and significantly, as shown in Figure 7-2. You can measure a more accurate value by averaging out the noise that is superimposed on the desired DC level. In a continuous signal, the averaged value between two times, $t1$ and $t2$, is defined as the signal integration between $t1$ and $t2$, divided by the measurement time, $t2 - t1$, as shown in Figure 7-1. The area between the averaged value V_{dc} and the signal that is above V_{dc} is equal to the area between V_{dc} and the signal that is under V_{dc}. For a sampled signal, the average value is the sum of the voltage samples divided by the measurement time in samples, or the mean value of the measurement samples. Refer to Chapter 6, *Measuring DC Voltage*, of the *LabVIEW Measurements Manual* for more information about averaging in LabVIEW.

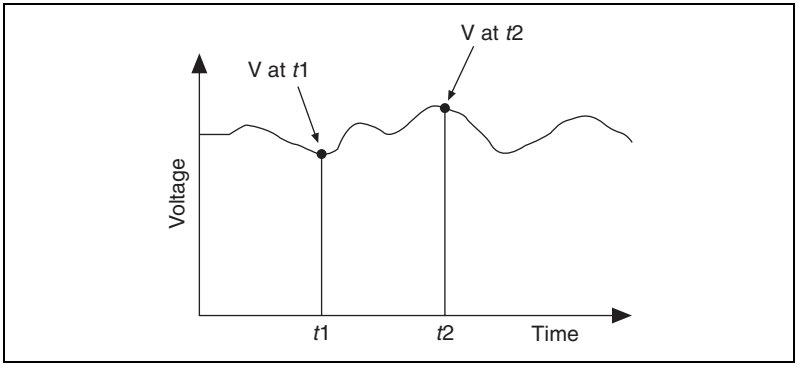


Figure 7-2. Instantaneous DC Measurements

An RMS measurement is an averaged quantity because it is the average energy in the signal over a measurement period. You can improve the RMS measurement accuracy by using a longer averaging time, equivalent to the integration time or measurement time.

There are several different strategies to use for making DC and RMS measurements, each dependent on the type of error or noise sources. When choosing a strategy, you must decide if accuracy or speed of the measurement is more important.

Common Error Sources Affecting DC and RMS Measurements

Some common error sources for DC measurements are single-frequency components (or tones), multiple tones, or random noise. These same error signals can interfere with RMS measurements so in many cases the approach taken to improve RMS measurements is the same as for DC measurements.

DC Overlapped with Single Tone

Consider the case where the signal you measure is composed of a DC signal and a single sine tone. The average of a single period of the sine tone is ideally zero because the positive half-period of the tone cancels the negative half-period.

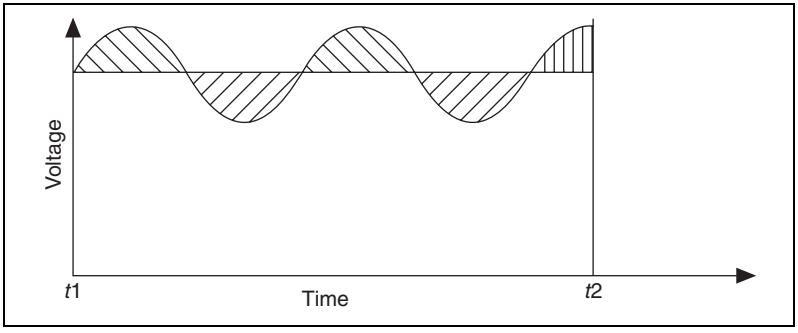


Figure 7-3. DC Signal Overlapped with Single Tone

Any remaining partial period, shown in Figure 7-3 with vertical hatching, introduces an error in the average value and therefore in the DC measurement. Increasing the averaging time reduces this error because the integration is always divided by the measurement time $t2 - t1$. If you know

the period of the sine tone, you can take a more accurate measurement of the DC value by using a measurement period equal to an integer number of periods of the sine tone. The most severe error occurs when the measurement time is a half-period different from an integer number of periods of the sine tone because this is the maximum area under or over the signal curve.

Defining the Equivalent Number of Digits

Defining the Equivalent Number of Digits (ENOD) makes it easier to relate a measurement error to a number of digits, similar to digits of precision. ENOD translates measurement accuracy into a number of digits.

$$\text{ENOD} = \log_{10}(\text{Relative Error})$$

A 1% error corresponds to two digits of accuracy, and a one part per million error corresponds to six digits of accuracy ($\log_{10}(0.000001) = 6$).

ENOD is only an approximation that tells you what order of magnitude of accuracy you can achieve under specific measurement conditions. This accuracy does not take into account any error introduced by the measurement instrument or data acquisition hardware itself. ENOD is only a tool for computing the reliability of a specific measurement technique.

Thus, the ENOD should at least match the accuracy of the measurement instrument or measurement requirements. For example, it is not necessary to use a measurement technique with an ENOD of six digits if your instrument has an accuracy of only 0.1% (three digits). Similarly, you do not get the six digits of accuracy from your six-digit accurate measurement instrument if your measurement technique is limited to an ENOD of only three digits.

DC Plus Sine Tone

Figure 7-4 shows that for a 1.0 VDC signal overlapped with a 0.5 V single sine tone, the worst ENOD increases with measurement time—*x*-axis shown in periods of the additive sine tone—at a rate of approximately one additional digit for 10 times more measurement time. To achieve 10 times more accuracy, you need to increase your measurement time by a factor of 10. In other words, accuracy and measurement time are related through a first-order function.

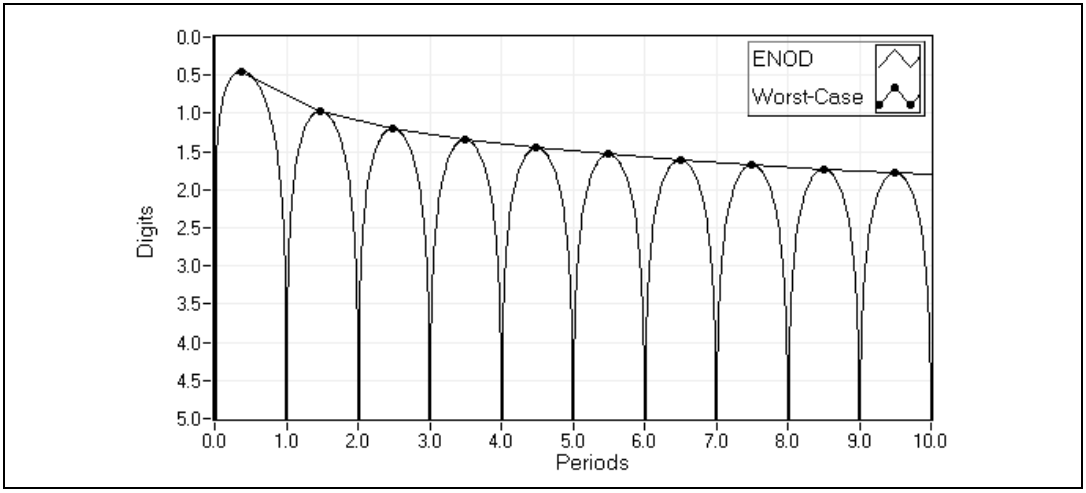


Figure 7-4. Digits versus Measurement Time for 1.0 VDC Signal with 0.5 V Single Tone

Windowing to Improve DC Measurements

The worst ENOD for a DC signal plus a sine tone occurs when the measurement time is at half-periods of the sine tone. You can greatly reduce these errors due to noninteger number of cycles by using a weighting function before integrating to measure the desired DC value. The most common weighting or window function is the Hann window, commonly known as the Hanning window.

Figure 7-5 shows a dramatic increase in accuracy from the use of the Hann window. The accuracy as a function of the number of sine tone periods is improved from a first-order function to a third-order function. In other words, you can achieve one additional digit of accuracy for every $10^{1/3} = 2.15$ times more measurement time using the Hann window instead of one digit for every 10 times more measurement time without using a window. As in the non-windowing case, the DC level is 1.0 V and the single tone peak amplitude is 0.5 V.

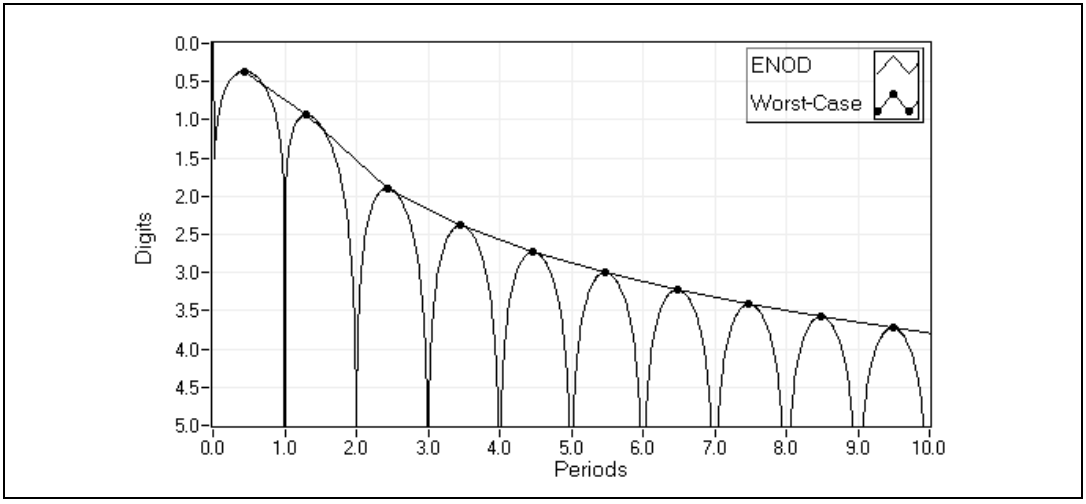


Figure 7-5. Digits versus Measurement Time for DC + Tone Using Hann Window

You can use other types of window functions to further reduce the necessary measurement time or greatly increase the resulting accuracy. Figure 7-6 shows that the Low Sidelobe (LSL) window can achieve more than six ENOD of worst accuracy when averaging your DC signal over only five periods of the sine tone (same test signal).

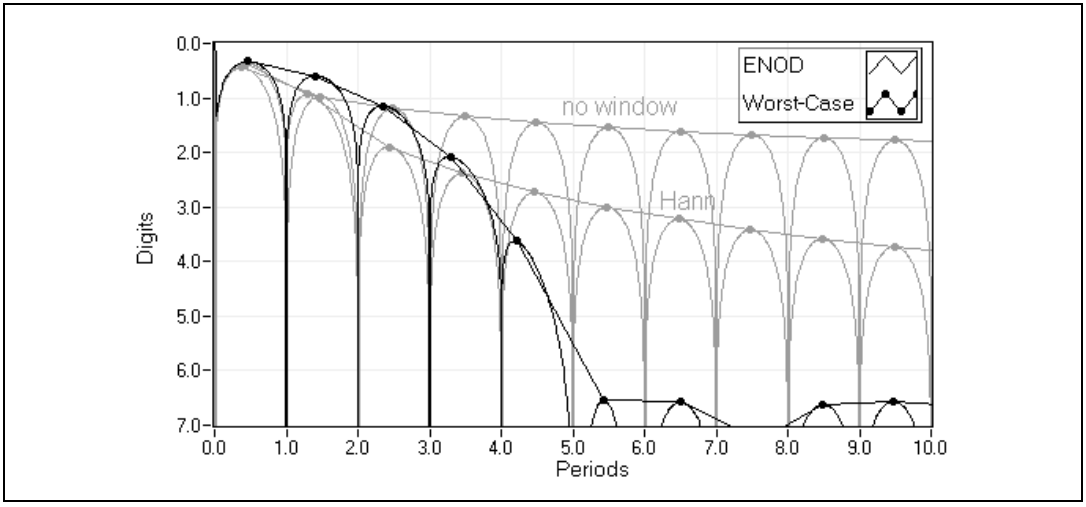


Figure 7-6. Digits versus Measurement Time for DC + Tone Using LSL Window

RMS Measurements Using Windows

Like DC measurements, the worst ENOD for measuring the RMS level of signals sometimes can be improved significantly by applying a window to the signal before RMS integration. For example, if you measure the RMS level of the DC signal plus a single sine tone, the most accurate measurements are made when the measurement time is an integer number of periods of the sine tone. Figure 7-7 shows that the worst ENOD varies with measurement time (in periods of the sine tone) for various window functions. Here, the test signal contains 0.707 VDC with 1.0 V peak sine tone.

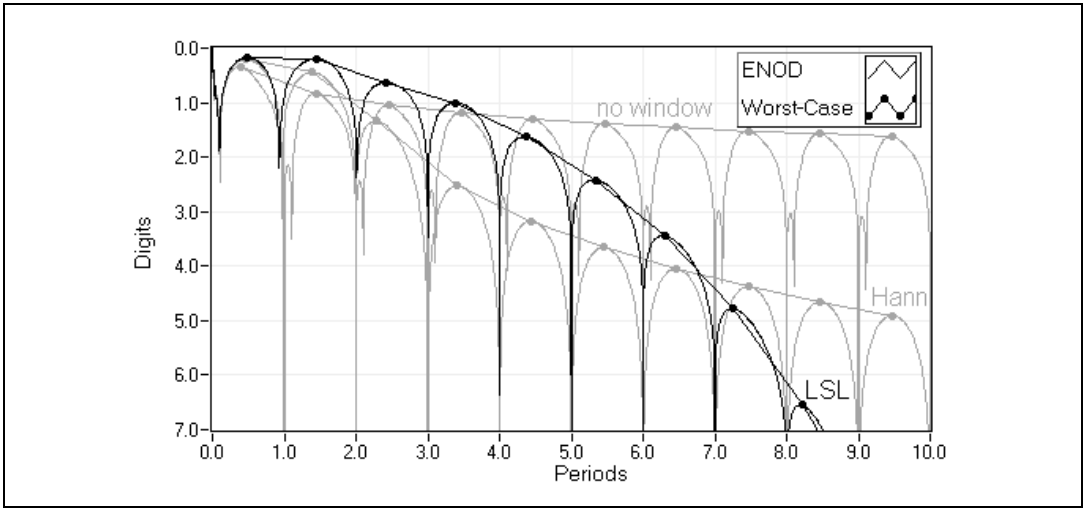


Figure 7-7. Digits versus Measurement Time for RMS Measurements

Applying the window to the signal increases RMS measurement accuracy significantly, but the improvement is not as large as in DC measurements. For this example, the LSL window achieves six digits of accuracy when the measurement time reaches eight periods of the sine tone.

Using Windows with Care

Window functions can be very useful to improve the speed of your measurement, but you must be careful. The Hann window is a general window recommended in most cases. Use more advanced windows such as the LSL window only if you know the window will improve the measurement. For example, you can reduce significantly RMS measurement accuracy if the signal you want to measure is composed of many frequency components close to each other in the frequency domain.

You also must make sure that the window is scaled correctly or that you update scaling after applying the window. The most useful window functions are pre-scaled by their *coherent gain*—the mean value of the window function—so that the resulting mean value of the scaled window function is always 1.00. DC measurements do not need to be scaled when using a properly scaled window function. For RMS measurements, each window has a specific *equivalent noise bandwidth* that you must use to scale integrated RMS measurements. You must scale RMS measurements using windows by the reciprocal of the square root of the equivalent noise bandwidth.

Rules for Improving DC and RMS Measurements

Use the following guidelines when determining a strategy for improving your DC and RMS measurements:

- If your signal is overlapped with a single tone, longer integration times increase the accuracy of your measurement. If you know the exact frequency of the sine tone, use a measurement time that corresponds to an exact number of sine periods. If you do not know the frequency of the sine tone, apply a window, such as a Hann window, to reduce significantly the measurement time needed to achieve a specific accuracy.
- If your signal is overlapped with many independent tones, increasing measurement time increases the accuracy of the measurement. As in the single tone case, using a window significantly reduces the measurement time needed to achieve a specific accuracy.
- If your signal is overlapped with noise, do not use a window. In this case, you can increase the accuracy of your measurement by increasing the integration time or by preprocessing or conditioning your noisy signal with a lowpass (or bandstop) filter.

RMS Levels of Specific Tones

You always can improve the accuracy of an RMS measurement by choosing a specific measurement time to contain an integer number of cycles of your sine tones or by using a window function. The measurement of the RMS value is based only on the time domain knowledge of your signal. You can use advanced techniques when you are interested in a specific frequency or narrow frequency range.

You can use bandpass or bandstop filtering before RMS computations to measure the RMS power in a specific band of frequencies. You also can use the Fast Fourier Transform (FFT) to pick out specific frequencies for RMS processing. Refer to Chapter 4, *Frequency Analysis*, for more information about the FFT.

The RMS level of a specific sine tone that is part of a complex or noisy signal can be extracted very accurately using frequency domain processing, leveraging the power of the FFT, and using the benefits of windowing.

8

Limit Testing

This chapter provides information about setting up an automated system for performing limit testing, specifying limits, and applications for limit testing.

You can use limit testing to monitor a waveform and determine if it always satisfies a set of conditions, usually upper and lower limits. The region bounded by the specified limits is a mask. The result of a limit or mask test is generally a pass or fail.

Setting up an Automated Test System

You can use the same method to create and control many different automated test systems. Complete the following basic steps to set up an automated test system for limit mask testing.

1. Configure the measurement by specifying arbitrary upper and lower limits. This defines your mask or region of interest.
2. Acquire data using a DAQ device.
3. Monitor the data to make sure it always falls within the specified mask.
4. Log the pass/fail results from step 3 to a file or visually inspect the input data and the points that fall outside the mask.
5. Repeat steps 2 through 4 to continue limit mask testing.

The following sections describe steps 1 and 3 in further detail. Assume that the signal to be monitored starts at $x = x_0$ and all the data points are evenly spaced. The spacing between each point is denoted by dx.

Specifying a Limit

Limits are classified into two types—continuous limits and segmented limits, as shown in Figure 8-1. The top graph in Figure 8-1 shows a continuous limit. A continuous limit is specified using a set of x and y points $\{\{x_1, x_2, x_3, \ldots\}, \{y_1, y_2, y_3, \ldots\}\}$. Completing step 1 creates a limit with the first point at x_0 and all other points at a uniform spacing of dx ($x_0 + dx$, $x_0 + 2dx$, …). This is done through a linear interpolation of the x and y values that define the limit. In Figure 8-1, black dots represent the

points at which the limit is defined and the solid line represents the limit you create. Creating the limit in step 1 reduces test times in step 3. If the spacing between the samples changes, you can repeat step 1. The limit is undefined in the region $x_0 < x < x_1$ and for $x > x_4$.

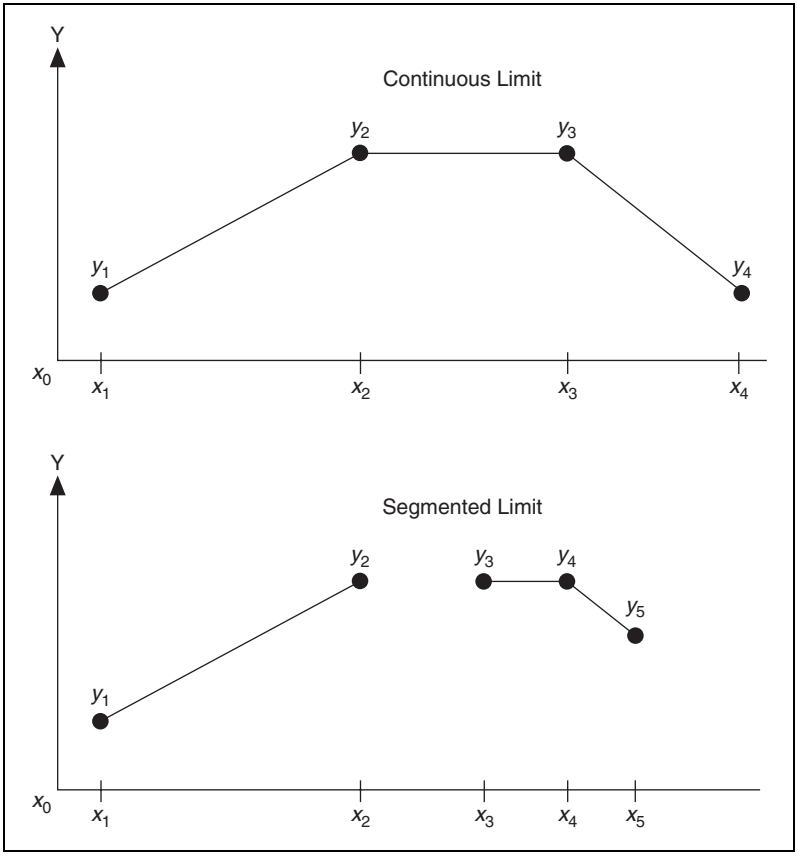


Figure 8-1. Continuous versus Segmented Limit Specification

The bottom graph of Figure 8-1 shows a segmented limit. The first segment is defined using a set of x and y points $\{\{x_1, x_2\}, \{y_1, y_2\}\}$. The second segment is defined using a set of points $\{x_3, x_4, x_5\}$ and $\{y_3, y_4, y_5\}$. You can define any number of such segments. As with continuous limits, step 1 uses linear interpolation to create a limit with the first point at x_0 and all other points with an uniform spacing of dx. The limit is undefined in the region $x_0 < x < x_1$ and in the region $x > x_5$. Also, the limit is undefined in the region $x_2 < x < x_3$.

Specifying a Limit Using a Formula

You can specify limits using formulas. Such limits are best classified as segmented limits. Each segment is defined by start and end frequencies and a formula. For example, the ANSI T1.413 recommendation specifies the limits for the transmit and receive spectrum of an ADSL signal in terms of formula. Table 8-1, which includes only a part of the specification, shows the start and end frequencies and the upper limits of the spectrum for each segment.

Table 8-1. ADSL Signal Recommendations

Start (kHz)	End (kHz)	Maximum (Upper Limit) Value (dBm/Hz)
0.3	4.0	–97.5
4.0	25.9	$-92.5 + 21.5 \log_2(f/4{,}000)$
25.9	138.0	–34.5
138.0	307.0	$-34.5 - 48.0 \log_2(f/138{,}000)$
307.0	1,221.0	–90

The limit is specified as an array of a set of *x* and *y* points, [{0.3, 4.0}{–97.5, –97.5}, {4.0, 25.9}{$-92.5 + 21.5 \log_2(f/4{,}000)$, $-92.5 + 21.5 \log_2(f/4{,}000)$}, …, {307.0, 1,221.0}{–90, –90}]. Each element of the array corresponds to a segment.

Figure 8-2 shows the segmented limit plot specified using the formulas shown in Table 8-1. The *x*-axis is on a logarithmic scale.

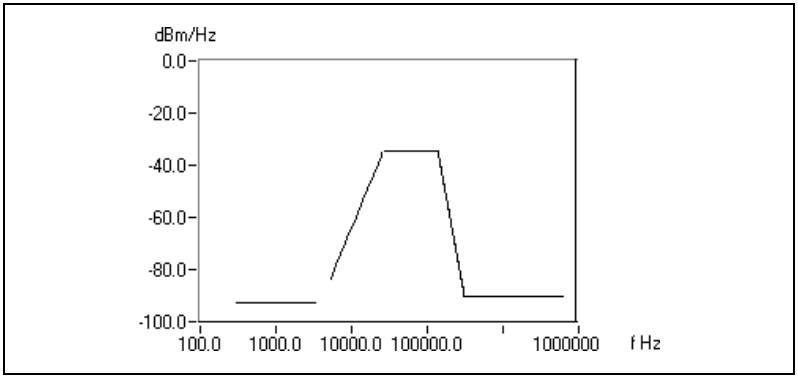


Figure 8-2. Segmented Limit Specified Using Formulas

Limit Testing

After you define your mask, you acquire a signal using a DAQ device. The sample rate is set at 1/*dx* S/s. Compare the signal with the limit. In step 1, you create a limit value at each point where the signal is defined. In step 3, you compare the signal with the limit. For the upper limit, if the data point is less than or equal to the limit point, the test passes. If the data point is greater than the limit point, the test fails. For the lower limit, if the data point is greater than or equal to the limit point, the test passes. If the data point is less than the limit point, the test fails.

Figure 8-3 shows the result of limit testing in a continuous mask case. The test signal falls within the mask at all the points it is sampled, other than points b and c. Thus, the limit test fails. Point d is not tested because it falls outside the mask.

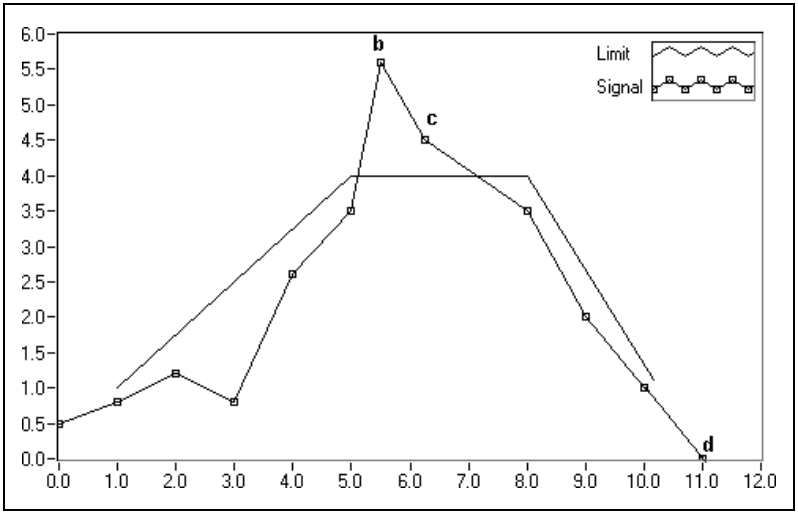


Figure 8-3. Result of Limit Testing with a Continuous Mask

Figure 8-4 shows the result of limit testing in a segmented mask case. All the points fall within the mask. Points b and c are not tested because the mask is undefined at those points. Thus, the limit test passes. Point d is not tested because it falls outside the mask.

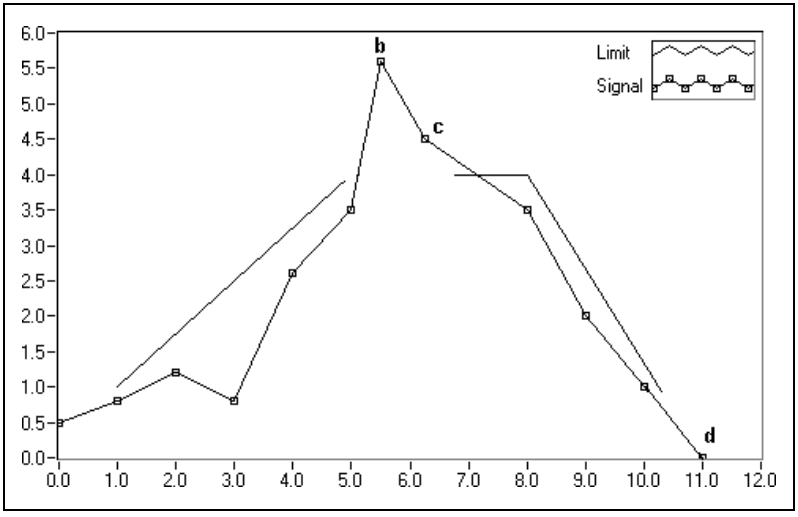


Figure 8-4. Result of Limit Testing with a Segmented Mask

Applications

You can use limit mask testing in a wide range of test and measurement applications. For example, you can use limit mask testing to determine that the power spectral density of ADSL signals meets the recommendations in the ANSI T1.413 specification. Refer to the *Specifying a Limit Using a Formula* section of this chapter for more information about ADSL signal limits.

The following sections provide examples of when you can use limit mask testing. In all these examples, the specifications are recommended by standards-generating bodies, such as the CCITT, ITU-T, ANSI, and IEC, to ensure that all the test and measurement systems conform to a universally accepted standard. In some other cases, the limit testing specifications are proprietary and are strictly enforced by companies for quality control.

Modem Manufacturing Example

Limit testing is used in modem manufacturing to make sure the transmit spectrum of the line signal meets the V.34 modem specification, as shown in Figure 8-5.

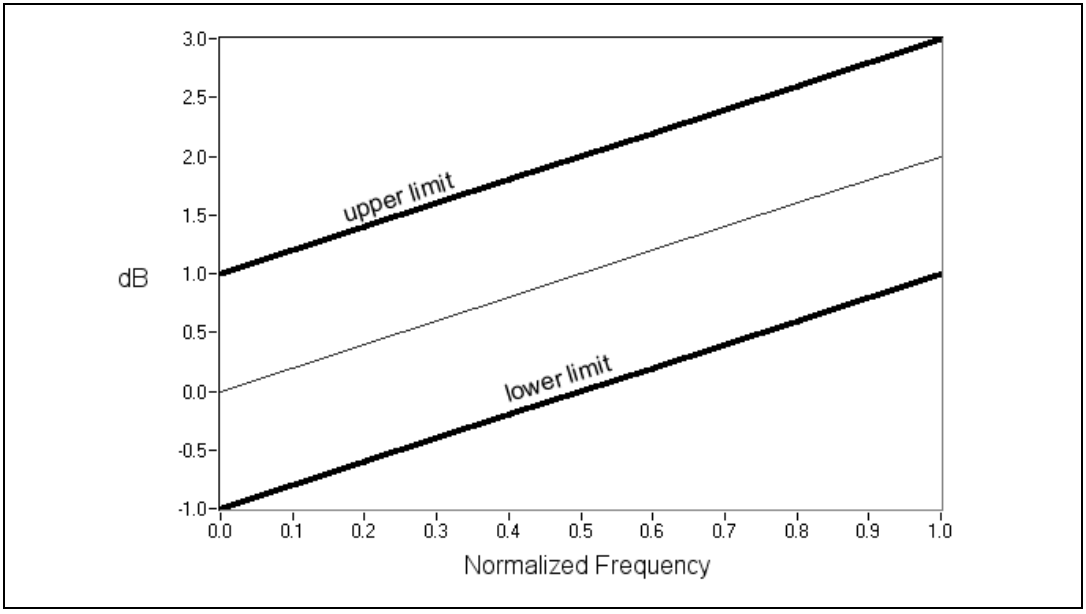


Figure 8-5. Upper and Lower Limit for V.34 Modem Transmitted Spectrum

The ITU-T V.34 recommendation contains specifications for a modem operating at data signaling rates up to 33,600 bits/s. It specifies that the spectrum for the line signal that transmits data conforms to the template shown in Figure 8-5. For example, for a normalized frequency of 1.0, the spectrum must always lie between 3 dB and 1 dB. All the modems must meet this specification. A modem manufacturer can set up an automated test system to monitor the transmit spectrum for the signals that the modem outputs. If the spectrum conforms to the specification, the modem passes the test and is ready for customer use. Recommendations such as the ITU-T V.34 are essential to ensure interoperability between modems from different manufacturers and to provide high-quality service to customers.

Digital Filter Design Example

You also can use limit mask testing in the area of digital filter design. You might want to design lowpass filters with a passband ripple of 10 dB and stopband attenuation of 60 dB. You can use limit testing to make sure the frequency response of the filter always meets these specifications. The first step in this process is to specify the limits. You can specify a lower limit of –10 dB in the passband region and an upper limit of –60 dB in the stopband region, as shown in Figure 8-6. After you specify these limits, you can run the actual test repeatedly to make sure that all the frequency responses of all the filters are designed to meet these specifications.

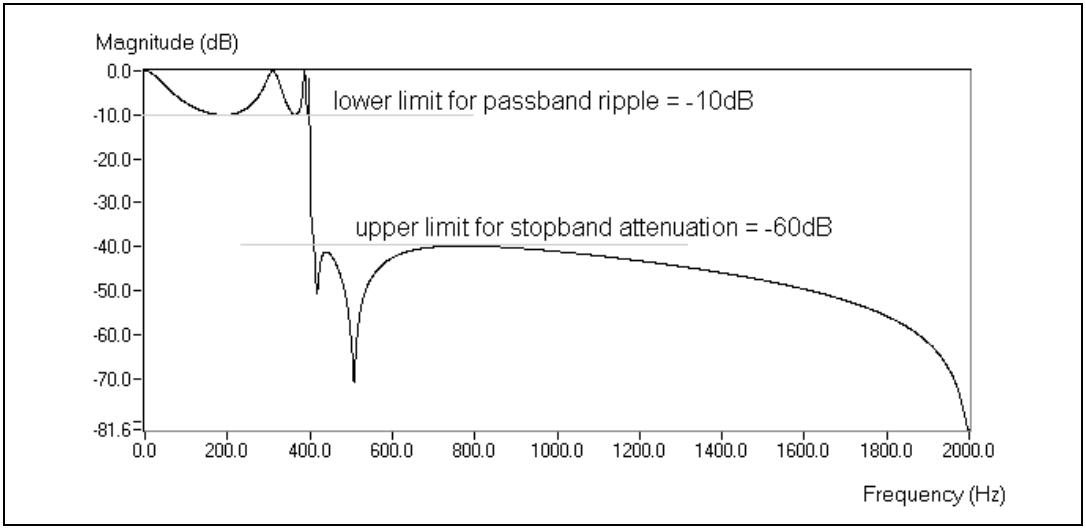


Figure 8-6. Limit Test of a Lowpass Filter Frequency Response

Pulse Mask Testing Example

The ITU-T G.703 recommendation specifies the pulse mask for signals with bit rates, $n \times 64$, where n is between 2 and 31. Figure 8-7 shows the pulse mask for interface at 1,544 kbits/s. Signals with this bit rate also are referred to as T1 signals. T1 signals must lie in the mask specified by the upper and lower limit. These limits are set to properly enable the interconnection of digital network components to form a digital path or connection.

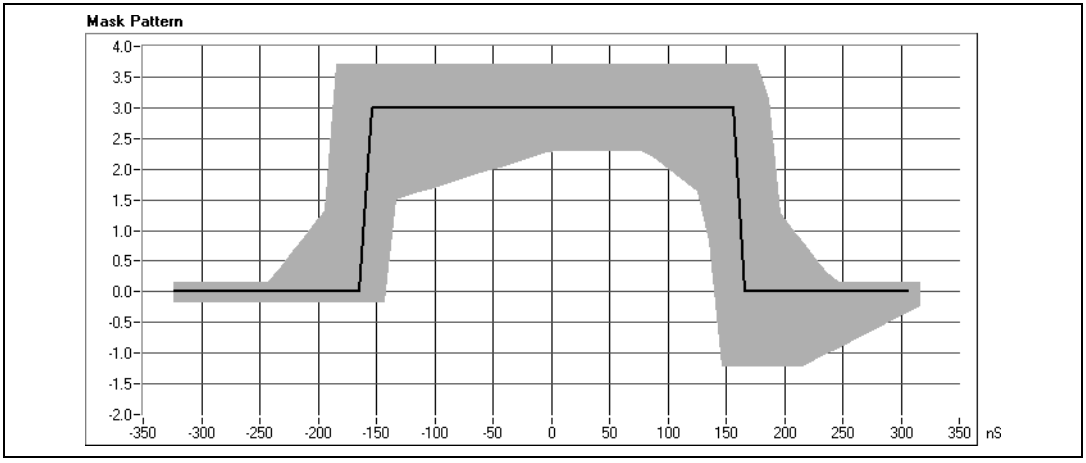


Figure 8-7. Pulse Mask Testing on T1/E1 Signals

Part II

Mathematics

This part provides information about mathematical concepts commonly used in analysis applications.

- Chapter 9, *Curve Fitting*, describes how to extract information from a data set to obtain a functional description.
- Chapter 10, *Probability and Statistics*, describes fundamental concepts of probability and statistics and how to use these concepts to solve real-world problems.
- Chapter 11, *Linear Algebra*, describes how to use the Linear Algebra VIs to perform matrix computation and analysis.
- Chapter 12, *Optimization*, describes basic concepts and methods used to solve optimization problems.
- Chapter 13, *Polynomials*, describes polynomials and operations involving polynomials.

9

Curve Fitting

This chapter describes how to extract information from a data set to obtain a functional description. Use the NI Example Finder to find examples of using the Curve Fitting VIs.

Introduction to Curve Fitting

The technique of curve fitting analysis extracts a set of curve parameters or coefficients from a data set to obtain a functional description of the data set. The least squares method of curve fitting fits a curve to a particular data set. Equation 9-1 defines the least square error.

$$e(a) = [f(x, a) - y(x)]^2 \quad (9\text{-}1)$$

where $e(a)$ is the least square error, $y(x)$ is the observed data set, $f(x, a)$ is the functional description of the data set, and a is the set of curve coefficients that best describes the curve.

For example, if $a = \{a_0, a_1\}$, the following equation yields the functional description.

$$f(x, a) = a_0 + a_1 x$$

The least squares algorithm finds a by solving the system defined by Equation 9-2.

$$\frac{\partial}{\partial a} e(a) = 0 \quad (9\text{-}2)$$

To solve the system defined by Equation 9-2, you set up and solve the Jacobian system generated by expanding Equation 9-2. After you solve the system for a, you can use the functional description $f(x, a)$ to obtain an estimate of the observed data set for any value of x.

The Curve Fitting VIs automatically set up and solve the Jacobian system and return the set of coefficients that best describes the data set. You can concentrate on the functional description of the data without having to solve the system in Equation 9-2.

Applications of Curve Fitting

In some applications, parameters such as humidity, temperature, and pressure can affect data you collect. You can model the statistical data by performing regression analysis and gain insight into the parameters that affect the data.

Figure 9-1 shows the block diagram of a VI that uses the Linear Fit VI to fit a line to a set of data points.

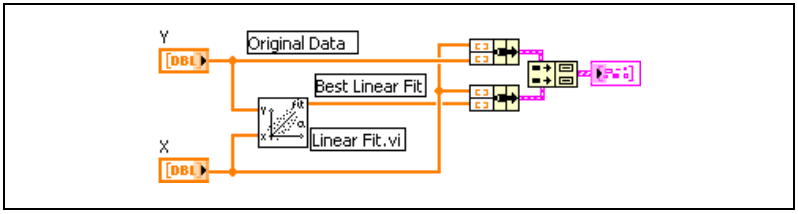


Figure 9-1. Fitting a Line to Data

You can modify the block diagram to fit exponential and polynomial curves by replacing the Linear Fit VI with the Exponential Fit VI or the General Polynomial Fit VI.

Figure 9-2 shows a multiplot graph of the result of fitting a line to a noisy data set.

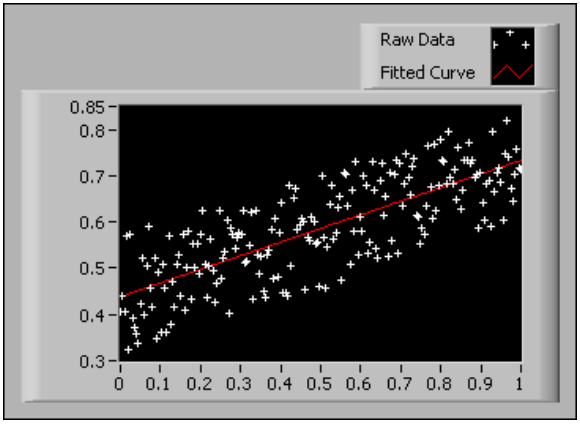


Figure 9-2. Fitting a Line to a Noisy Data Set

The practical applications of curve fitting include the following applications:

- Removing measurement noise
- Filling in missing data points, such as when one or more measurements are missing or improperly recorded
- Interpolating, which is estimating data between data points, such as if the time between measurements is not small enough
- Extrapolating, which is estimating data beyond data points, such as looking for data values before or after a measurement
- Differentiating digital data, such as finding the derivative of the data points by modeling the discrete data with a polynomial and differentiating the resulting polynomial equation
- Integrating digital data, such as finding the area under a curve when you have only the discrete points of the curve
- Obtaining the trajectory of an object based on discrete measurements of its velocity, which is the first derivative, or acceleration, which is the second derivative

General LS Linear Fit Theory

For a given set of observation data, the general least-squares (LS) linear fit problem is to find a set of coefficients that fits the linear model, as shown in Equation 9-3.

$$y_i = b_o x_{i0} + \ldots + b_{k-1} x_{ik-1} = \sum_{j=0}^{k-1} b_j x_{ij} \qquad i = 0, 1, \ldots, n-1 \quad (9\text{-}3)$$

where x_{ij} is the observed data contained in the observation matrix H, n is the number of elements in the set of observed data and the number of rows of in H, b is the set of coefficients that fit the linear model, and k is the number of coefficients.

The following equation defines the observation matrix *H*.

$$H = \begin{bmatrix} x_{00} & x_{01} \cdots & x_{0k-1} \\ x_{10} & x_{11} \cdots & x_{1k-1} \\ \vdots & & \\ x_{n-10} & x_{n-12} \cdots & x_{n-1k-1} \end{bmatrix}$$

You can rewrite Equation 9-3 as the following equation.

$$Y = HB.$$

The general LS linear fit model is a multiple linear regression model. A multiple linear regression model uses several variables, x_{i0}, x_{i1}, …, x_{ik-1}, to predict one variable, y_i.

In most analysis situations, you acquire more observation data than coefficients. Equation 9-3 might not yield all the coefficients in set *B*. The fit problem becomes to find the coefficient set *B* that minimizes the difference between the observed data y_i and the predicted value z_i. Equation 9-4 defines z_i.

$$z_i = \sum_{j=0}^{k-1} b_j x_{ij} \tag{9-4}$$

You can use the least chi-square plane method to find the solution set *B* that minimizes the quantity given by Equation 9-5.

$$\chi^2 = \sum_{i=0}^{n-1} \left(\frac{y_i - z_i}{\sigma_i} \right)^2 = \sum_{i=0}^{n-1} \left(\frac{y_i - \sum_{j=0}^{k-1} b_j x_{ij}}{\sigma_i} \right)^2 = |H_0 B - Y_0|^2 \tag{9-5}$$

where $h_{oij} = \frac{x_{ij}}{\sigma_i}$, $y_{oi} = \frac{y_i}{\sigma_i}$, $i = 0, 1, \ldots, n-1$, and $j = 0, 1, \ldots, k-1$.

In Equation 9-5, σ_i is the standard deviation. If the measurement errors are independent and normally distributed with constant standard deviation, $\sigma_i = \sigma$, Equation 9-5 also is the least-square estimation.

You can use the following methods to minimize χ^2 from Equation 9-5:

- Solve normal equations of the least-square problems using LU or Cholesky factorization.
- Minimize χ^2 to find the least-square solution of equations.

Solving normal equations involves completing the following steps.

1. Set the partial derivatives of χ^2 to zero with respect to b_0, b_1, …, b_{k-1}, as shown by the following equations.

$$\begin{cases} \dfrac{\partial \chi^2}{\partial b_0} = 0 \\ \dfrac{\partial \chi^2}{\partial b_1} = 0 \\ \vdots \\ \dfrac{\partial \chi^2}{\partial b_{k-1}} = 0 \end{cases} \tag{9-6}$$

2. Derive the equations in Equation 9-6 to the following equation form.

$$H_0^T H_0 B = H_0^T Y \tag{9-7}$$

where H_0^T is the transpose of H_0.

Equations of the form given by Equation 9-7 are called normal equations of the least-square problems. You can solve them using LU or Cholesky factorization algorithms. However, the solution from the normal equations is susceptible to roundoff error.

The preferred method of minimizing χ^2 is to find the least-square solution of equations. Equation 9-8 defines the form of the least-square solution of equations.

$$H_0 B = Y_0 \tag{9-8}$$

You can use QR or SVD factorization to find the solution set *B* for Equation 9-8. For QR factorization, you can use the Householder algorithm, the Givens algorithm, or the Givens 2 algorithm, which also is known as the fast Givens algorithm. Different algorithms can give you different precision. In some cases, if one algorithm cannot solve the equation, another algorithm might solve it. You can try different algorithms to find the one best suited for the observation data.

Polynomial Fit with a Single Predictor Variable

Polynomial fit with a single predictor variable uses one variable to predict another variable. Polynomial fit with a single predictor variable is a special case of multiple regression. If the observation data sets are $\{x_i, y_i\}$, where $i = 0, 1, \ldots, n-1$, Equation 9-9 defines the model for polynomial fit.

$$y_i = \sum_{j=0}^{k-1} b_j x_i^j = b_0 + b_1 x_i + b_2 x_i^2 + \ldots + b_{k-1} x_i^{k-1} \quad i = 0, 1, 2, \ldots, n-1 \quad (9\text{-}9)$$

Comparing Equations 9-3 and 9-9 shows that $x_{ij} = x_i^j$, as shown by the following equations.

$$\begin{aligned} x_{i0} &= x_i^0 \\ &= 1, x_{i1} = x_i, x_{i2} = x_i^2, \ldots, x_{ik-1} = x_i^{k-1} \end{aligned}$$

Because $x_{ij} = x_i^j$, you can build the observation matrix *H* as shown by the following equation.

$$H = \begin{Bmatrix} 1 & x_0 & x_0^2 & \ldots & x_0^{k-1} \\ 1 & x_1 & x_1^2 & \ldots & x_1^{k-1} \\ & & \vdots & & \\ 1 & x_{n-1} & x_{n-1}^2 & \ldots & x_{n-1}^{k-1} \end{Bmatrix}$$

Instead of using $x_{ij} = x_j^i$, you also can choose another function formula to fit the data sets $\{x_i, y_i\}$. In general, you can select $x_{ij} = f_j(x_i)$. Here, $f_j(x_i)$

is the function model that you choose to fit your observation data. In polynomial fit, $f_j(x_i) = x_i^j$.

In general, you can build H as shown in the following equation.

$$H = \begin{Bmatrix} f_0(x_0) & f_1(x_0) & f_2(x_0) & \dots & f_{k-1}(x_0) \\ f_0(x_1) & f_1(x_1) & f_2(x_1) & \dots & f_{k-1}(x_1) \\ & & \vdots & & \\ f_0(x_{n-1}) & f_1(x_{n-1}) & f_2(x_{n-1}) & \dots & f_{k-1}(x_{n-1}) \end{Bmatrix}$$

The following equation defines the fit model.

$$y_i = b_0 f_0(x) + b_1 f_1(x) + \dots + b_{k-1} f_{k-1}(x)$$

Curve Fitting in LabVIEW

For the Curve Fitting VIs, the input sequences **Y** and **X** represent the data set $y(x)$. A sample or point in the data set is (x_i, y_i). x_i is the i^{th} element of the sequence **X**. y_i is the i^{th} element of the sequence **Y**.

Some Curve Fitting VIs return only the coefficients for the curve that best describe the input data while other Curve Fitting VIs return the fitted curve. Using the VIs that return only coefficients allows you to further manipulate the data. The VIs that return the fitted curve also return the coefficients and the mean squared error (MSE). MSE is a relative measure of the residuals between the expected curve values and the actual observed values. Because the input data represents a discrete system, the VIs use the following equation to calculate MSE.

$$\text{MSE} = \frac{1}{n}\sum_{i=0}^{n-1}(f_i - y_i)^2$$

where f is the sequence representing the fitted values, y is the sequence representing the observed values, and n is the number of observed sample points.

Linear Fit

The Linear Fit VI fits experimental data to a straight line of the general form described by the following equation.

$$y = mx + b$$

The Linear Fit VI calculates the coefficients a_0 and a_1 that best fit the experimental data ($x[i]$ and $y[i]$) to a straight line model described by the following equation.

$$y[i]=a_0 + a_1x[i]$$

where $y[i]$ is a linear combination of the coefficients a_0 and a_1.

Exponential Fit

The Exponential Fit VI fits data to an exponential curve of the general form described by the following equation.

$$y = ae^{bx}$$

The following equation specifically describes the exponential curve resulting from the exponential fit algorithm.

$$y[i] = a_0e^{a_1x[i]}$$

General Polynomial Fit

The General Polynomial Fit VI fits data to a polynomial function of the general form described by the following equation.

$$y = a + bx + cx^2 + \ldots$$

The following equation specifically describes the polynomial function resulting from the general polynomial fit algorithm.

$$y[i] = a_0 + a_1x[i]+a_2x[i]^2\ldots$$

General LS Linear Fit

The General LS Linear Fit VI fits data to a line described by the following equation.

$$y[i] = a_0 + a_1 f_1(x[i]) + a_2 f_2(x[i]) + \ldots$$

where $y[i]$ is a linear combination of the parameters a_0, a_1, a_2, ….

You can extend the concept of a linear combination of coefficients further so that the multiplier for a_1 is some function of x, as shown in the following equations.

$$y[i] = a_0 + a_1 \sin(\omega x[i])$$

$$y[i] = a_0 + a_1 (x[i])^2$$

$$y[i] = a_0 + a_1 \cos(\omega x[i]^2)$$

where ω is the angular frequency.

In each of the preceding equations, $y[i]$ is a linear combination of the coefficients a_0 and a_1. In the case of the General LS Linear Fit VI, you can have $y[i]$ that is a linear combination of several coefficients. Each coefficient can have a multiplier of some function of $x[i]$. Therefore, you can use the General LS Linear Fit VI to calculate coefficients of the functional models and represent the coefficients of the functional models as linear combinations of the coefficients, as shown in the following equations.

$$y = a_0 + a_1 \sin(\omega x)$$

$$y = a_0 + a_1 x^2 + a_2 \cos(\omega x^2)$$

$$y = a_0 + a_1 (3\sin(\omega x)) + a_2 x^3 + \frac{a_3}{x} + \ldots$$

In each of the preceding equations, y is a linear function of the coefficients, although it might be a nonlinear function of x.

Computing Covariance

The General LS Linear Fit VI returns a $k \times k$ matrix of covariances between the coefficients a_k. The General LS Linear Fit VI uses the following equation to compute the covariance matrix C.

$$C = (H_0^T H_0)^{-1}$$

Building the Observation Matrix

When you use the General LS Linear Fit VI, you must build the observation matrix H. For example, Equation 9-10 defines a model for data from a transducer.

$$y = a_0 + a_1 \sin(\omega x) + a_2 \cos(\omega x) + a_3 x^2 \quad (9\text{-}10)$$

In Equation 9-10, each a_j has the following different functions as a multiplier:

- One multiplies a_0.
- $\sin(\omega x)$ multiplies a_1.
- $\cos(\omega x)$ multiplies a_2.
- x^2 multiplies a_3.

To build H, set each column of H to the independent functions evaluated at each x value $x[i]$. If the data set contains 100 x values, the following equation defines H.

$$H = \begin{bmatrix} 1 & \sin(\omega x_0) & \cos(\omega x_0) & x_0^2 \\ 1 & \sin(\omega x_1) & \cos(\omega x_1) & x_1^2 \\ 1 & \sin(\omega x_2) & \cos(\omega x_2) & x_2^2 \\ \cdots & \cdots & \cdots & \cdots \\ 1 & \sin(\omega x_{99}) & \cos(\omega x_{99}) & x_{99}^2 \end{bmatrix}$$

If the data set contains N data points and if k coefficients (a_0, a_1, … a_{k-1}) exist for which to solve, H is an $N \times k$ matrix with N rows and k columns. Therefore, the number of rows in H equals the number of data points N. The number of columns in H equals the number of coefficients k.

Nonlinear Levenberg-Marquardt Fit

The nonlinear Levenberg-Marquardt Fit method fits data to the curve described by the following equation.

$$y[i] = f(x[i]), a_0, a_1, a_2,...$$

where a_0, a_1, a_2,... are the parameters.

The nonlinear Levenberg-Marquardt method is the most general curve fitting method and does not require y to have a linear relationship with a_0, a_1, a_2, ... You can use the nonlinear Levenberg-Marquardt method to fit linear or nonlinear curves. However, the most common application of the method is to fit a nonlinear curve because the general linear fit method is better for linear curve fitting. You must verify the results you obtain with the Levenberg-Marquardt method because the method does not always guarantee a correct result.

10

Probability and Statistics

This chapter describes fundamental concepts of probability and statistics and how to use these concepts to solve real-world problems. Use the NI Example Finder to find examples of using the Probability and Statistics VIs.

Statistics

Statistics allow you to summarize data and draw conclusions for the present by condensing large amounts of data into a form that brings out all the essential information and is yet easy to remember. To condense data, single numbers must make the data more intelligible and help draw useful inferences. For example, in a season, a sports player participates in 51 games and scores a total of 1,568 points. The total of 1,568 points includes 45 points in Game A, 36 points in Game B, 51 points in Game C, 45 points in Game D, and 40 points in Game E. As the number of games increases, remembering how many points the player scored in each individual game becomes increasingly difficult. If you divide the total number of points that the player scored by the number of games played, you obtain a single number that tells you the average number of points the player scored per game. Equation 10-1 yields the points per game average for the player.

$$\frac{1{,}568 \text{ points}}{51 \text{ games}} = 30.7 \text{ points per game average} \qquad (10\text{-}1)$$

Computing percentage provides a method for making comparisons. For example, the officials of an American city are considering installing a traffic signal at a major intersection. The purpose of the traffic signal is to protect motorists turning left from oncoming traffic. However, the city has only enough money to fund one traffic signal but has three intersections that potentially need the signal. Traffic engineers study each of the three intersections for a week. The engineers record the total number of cars using the intersection, the number of cars travelling straight through the

intersection, the number of cars making left-hand turns, and the number of cars making right-hand turns. Table 10-1 shows the data for one of the intersections.

Table 10-1. Data for One Major Intersection

Day	Total Number of Cars Using the Intersection	Number of Cars Turning Left	Number of Cars Turning Right	Number of Cars Continuing Straight
1	1,258	528	330	400
2	1,306	549	340	417
3	1,355	569	352	434
4	1,227	515	319	393
5	1,334	560	346	428
6	694	291	180	223
7	416	174	108	134
Totals	7,590	3,186	1,975	2,429

Looking only at the raw data from each intersection might make determining which intersection needs the traffic signal difficult because the raw numbers can vary widely. However, computing the percentage of cars turning at each intersection provides a common basis for comparison. To obtain the percentage of cars turning left, divide the number of cars turning left by the total number of cars using the intersection and multiply that result by 100. For the intersection whose data is shown in Table 10-1, the following equation gives the percentage of cars turning left.

$$\frac{3{,}186}{7{,}590} \times 100 = 42\%$$

Given the data for the other two intersections, the city officials can obtain the percentage of cars turning left at those two intersections. Converting the raw data to a percentage condenses the information for the three intersections into single numbers representing the percentage of cars that turn left at each intersection. The city officials can compare the percentage of cars turning left at each intersection and rank the intersections in order of highest percentage of cars turning left to the lowest percentage of cars

turning left. Ranking the intersections can help determine where the traffic signal is needed most. Thus, in a broad sense, the term statistics implies different ways to summarize data to derive useful and important information from it.

Mean

The mean value is the average value for a set of data samples. The following equation defines an input sequence *X* consisting of *n* samples.

$$X = \{x_0, x_1, x_2, x_3, \ldots, x_{n-1}\}$$

The following equation yields the mean value for input sequence *X*.

$$\bar{x} = \frac{1}{n}(x_0 + x_1 + x_2 + x_3 + \ldots + x_{n-1})$$

The mean equals the sum of all the sample values divided by the number of samples, as shown in Equation 10-1.

Median

The median of a data sequence is the midpoint value in the sorted version of the sequence. The median is useful for making qualitative statements, such as whether a particular data point lies in the upper or lower portion of an input sequence.

The following equation represents the sorted sequence of an input sequence *X*.

$$S = \{s_0, s_1, s_2, \ldots, s_{n-1}\}$$

You can sort the sequence either in ascending order or in descending order. The following equation yields the median value of *S*.

$$x_{median} = \begin{cases} s_i & n \text{ is odd} \\ 0.5(s_{k-1} + s_k) & n \text{ is even} \end{cases} \quad (10\text{-}2)$$

where $i = \frac{n-1}{2}$ and $k = \frac{n}{2}$.

Equation 10-3 defines a sorted sequence consisting of an odd number of samples sorted in descending order.

$$S = \{5, 4, 3, 2, 1\} \quad (10\text{-}3)$$

In Equation 10-3, the median is the midpoint value 3.

Equation 10-4 defines a sorted sequence consisting of an even number of samples sorted in ascending order.

$$S = \{1, 2, 3, 4\} \tag{10-4}$$

The sorted sequence in Equation 10-4 has two midpoint values, 2 and 3. Using Equation 10-2 for n is even, the following equation yields the median value for the sorted sequence in Equation 10-4.

$$x_{median} = 0.5(s_{k-1} + s_k) = 0.5(2 + 3) = 2.5$$

Sample Variance and Population Variance

The Standard Deviation and Variance VI can calculate either the sample variance or the population variance. Statisticians and mathematicians prefer to use the sample variance. Engineers prefer to use the population variance. For values of $n \geq 30$, both methods produce similar results.

Sample Variance

Sample variance measures the spread or dispersion of the sample values. You can use the sample variance as a measure of the consistency. The sample variance is always positive, except when all the sample values are equal to each other and in turn, equal to the mean.

The sample variance s^2 for an input sequence X equals the sum of the squares of the deviations of the sample values from the mean divided by $n - 1$, as shown in the following equation.

$$s^2 = \frac{1}{n-1}[(x_1 - \bar{x})^2 + (x_2 - \bar{x})^2 + \ldots + (x_n - \bar{x})^2]$$

where $n > 1$ and is the number of samples in X and $\bar{x}$ is the mean of X.

Population Variance

The population variance σ^2 for an input sequence X equals the sum of the squares of the deviations of the sample values from the mean divided by n, as shown in the following equation.

$$\sigma^2 = \frac{1}{n}[(x_1 - \bar{x})^2 + (x_2 - \bar{x})^2 + \ldots + (x_n - \bar{x})^2]$$

where $n > 1$ and is the number of samples in X, and $\bar{x}$ is the mean of X.

Standard Deviation

The standard deviation s of an input sequence equals the positive square root of the sample variance s^2, as shown in the following equation.

$$s = \sqrt{s^2}$$

Mode

The mode of an input sequence is the value that occurs most often in the input sequence. The following equation defines an input sequence X.

$$X = \{0, 1, 3, 3, 4, 4, 4, 5, 5, 7\}$$

The mode of X is 4 because 4 is the value that occurs most often in X.

Moment about the Mean

The moment about the mean is a measure of the deviation of the elements in an input sequence from the mean. The following equation yields the m^{th} order moment σ_n^m for an input sequence X.

$$\sigma_x^m = \frac{1}{n}\sum_{i=0}^{n-1}(x_i - \bar{x})^m$$

where n is the number of elements in X and $\bar{x}$ is the mean of X.

For $m = 2$, the moment about the mean equals the population variance σ^2.

Skewness

Skewness is a measure of symmetry and corresponds to the third-order moment.

Kurtosis

Kurtosis is a measure of peakedness and corresponds to the fourth-order moment.

Histogram

A histogram is a bar graph that displays frequency data and is an indication of the data distribution. A histogram provides a method for graphically displaying data and summarizing key information.

Equation 10-5 defines a data sequence.

$$X = \{0, 1, 3, 3, 4, 4, 4, 5, 5, 8\} \quad (10\text{-}5)$$

To compute a histogram for X, divide the total range of values into the following eight intervals, or bins:

- 0–1
- 1–2
- 2–3
- 3–4
- 4–5
- 5–6
- 6–7
- 7–8

The histogram display for X indicates the number of data samples that lie in each interval, excluding the upper boundary. Figure 10-1 shows the histogram for the sequence in Equation 10-5.

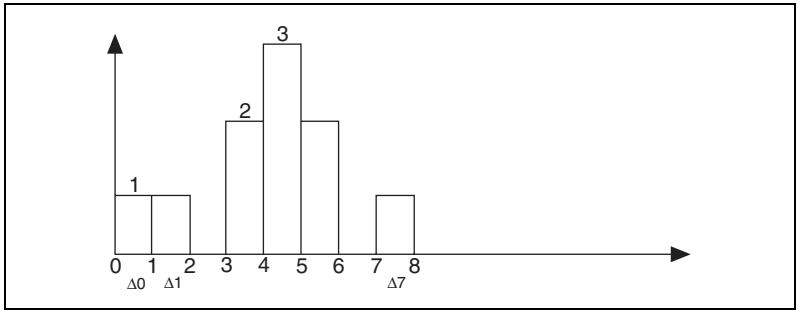


Figure 10-1. Histogram

Figure 10-1 shows that no data samples are in the 2–3 and 6–7 intervals. One data sample lies in each of the intervals 0–1, 1–2, and 7–8. Two data samples lie in each of the intervals 3–4 and 5–6. Three data samples lie in the 4–5 interval.

The number of intervals in the histogram affects the resolution of the histogram. A common method of determining the number of intervals to use in a histogram is Sturges' Rule, which is given by the following equation.

$$\text{Number of Intervals} = 1 + 3.3\log(\text{size of } (X))$$

Mean Square Error (mse)

The mean square error (mse) is the average of the sum of the square of the difference between the corresponding elements of two input sequences. The following equation yields the mse for two input sequences *X* and *Y*.

$$\text{mse} = \frac{1}{n}\sum_{i=0}^{n-1}(x_i - y_i)^2$$

where *n* is the number of data points.

You can use the mse to compare two sequences. For example, system S_1 receives a digital signal *x* and produces an output signal y_1. System S_2 produces y_2 when it receives *x*. Theoretically, $y_1 = y_2$. To verify that $y_1 = y_2$, you want to compare y_1 and y_2. Both y_1 and y_2 contain a large number of data points. Because y_1 and y_2 are large, an element-by-element comparison is difficult. You can calculate the mse of y_1 and y_2. If the mse is smaller than an acceptable tolerance, y_1 and y_2 are equivalent.

Root Mean Square (rms)

The root mean square (rms) of an input sequence equals the positive square root of the mean of the square of the input sequence. In other words, you can square the input sequence, take the mean of the new squared sequence, and take the square root of the mean of the new squared sequence. The following equation yields the rms Ψ_x for an input sequence X.

$$\Psi_x = \sqrt{\frac{1}{n}\sum_{i=0}^{n-1} x_i^2}$$

where n is the number of elements in X.

Root mean square is a widely used quantity for analog signals. The following equation yields the root mean square voltage V_{rms} for a sine voltage waveform.

$$V_{rms} = \frac{V_p}{\sqrt{2}}$$

where V_p is the peak amplitude of the signal.

Probability

In any random experiment, a chance, or probability, always exists that a particular event will or will not occur. The probability that event A will occur is the ratio of the number of outcomes favorable to A to the total number of equally likely outcomes.

You can assign a number between zero and one to an event as an indication of the probability that the event will occur. If you are absolutely sure that the event will occur, its probability is 100% or one. If you are sure that the event will not occur, its probability is zero.

Random Variables

Many experiments generate outcomes that you can interpret in terms of real numbers. Some examples are the number of cars passing a stop sign during a day, the number of voters favoring candidate A, and the number of accidents at a particular intersection. Random variables are the numerical outcomes of an experiment whose values can change from experiment to experiment.

Discrete Random Variables

Discrete random variables can take on only a finite number of possible values. For example, if you roll a single unbiased die, six possible events can occur. The roll can result in a 1, 2, 3, 4, 5, or 6. The probability that a 2 will result is one in six, or 0.16666.

Continuous Random Variables

Continuous random variables can take on any value in an interval of real numbers. For example, an experiment measures the life expectancy x of 50 batteries of a certain type. The batteries selected for the experiment come from a larger population of the same type of battery. Figure 10-2 shows the histogram for the observed data.

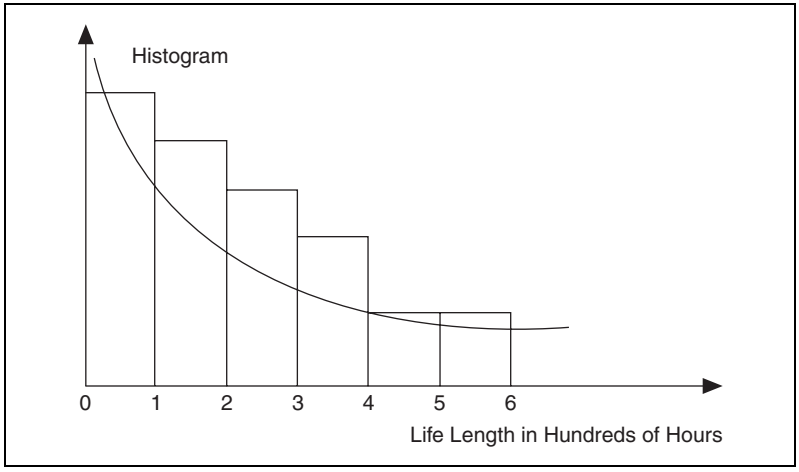


Figure 10-2. Life Lengths Histogram

Figure 10-2 shows that most of the values for x are between zero and 100 hours. The histogram values drop off smoothly for larger values of x. The value of x can equal any value between zero and the largest observed value, making x a continuous random variable.

You can approximate the histogram in Figure 10-2 by an exponentially decaying curve. The exponentially decaying curve is a mathematical model for the behavior of the data sample. If you want to know the probability that a randomly selected battery will last longer than 400 hours, you can approximate the probability value by the area under the curve to the right of the value 4. The function that models the histogram of the random variable is the probability density function. Refer to the *Probability*

Distribution and Density Functions section of this chapter for more information about the probability density function.

A random variable X is continuous if it can take on an infinite number of possible values associated with intervals of real numbers and a probability density function $f(x)$ exists such that the following relationships and equations are true.

$$f(x) \geq 0 \text{ for all } x$$

$$\int_{-\infty}^{\infty} f(x)dx = 1$$

$$P(a \leq X \leq b) = \int_{a}^{b} f(x)dx \tag{10-6}$$

The chance that X will assume a specific value of $X = a$ is extremely small. The following equation shows solving Equation 10-6 for a specific value of X.

$$X = a, P(X = a) = \int_{a}^{a} f(x)dx = 0$$

Because X can assume an infinite number of possible values, the probability of it assuming a specific value is zero.

Normal Distribution

The normal distribution is a continuous probability distribution. The functional form of the normal distribution is the normal density function. The following equation defines the normal density function $f(x)$.

$$f(x) = \frac{1}{\sqrt{2\pi}s} e^{-(x-\bar{x})^2/(2s^2)}$$

The normal density function has a symmetric bell shape. The following parameters completely determine the shape and location of the normal density function:

- The center of the curve is the mean value $\bar{x} = 0$.
- The spread of the curve is the variance $s^2 = 1$.

If a random variable has a normal distribution with a mean equal to zero and a variance equal to one, the random variable has a standard normal distribution.

Computing the One-Sided Probability of a Normally Distributed Random Variable

The following equation defines the one-sided probability of a normally distributed random variable.

$$p = Prob(X \leq x)$$

where p is the one-sided probability, X is a standard normal distribution with the mean value equal to zero and the variance equal to one, and x is the value.

You can use the Normal Distribution VI to compute p for x. Suppose you measure the heights of 1,000 randomly selected adult males and obtain a data set S. The histogram distribution of S shows many measurements grouped closely about a mean height, with relatively few very short and very tall males in the population. Therefore, you can closely approximate the histogram with the normal distribution.

Next, you want to find the probability that the height of a male in a different set of 1,000 randomly chosen males is greater than or equal to 170 cm. After normalizing 170 cm, you can use the Normal Distribution VI to compute the one-sided probability p. Complete the following steps to normalize 170 cm and calculate p using the Normal Distribution VI.

1. Subtract the mean from 170 cm.
2. Scale the difference from step 1 by the standard deviation to obtain the normalized x value.
3. Wire the normalized x value to the **x** input of the Normal Distribution VI and run the VI.

The choice of the probability density function is fundamental to obtaining a correct probability value.

In addition to the normal distribution method, you can use the following methods to compute *p*:

- Chi-Square distribution
- F distribution
- T distribution

Finding *x* with a Known *p*

The Inv Normal Distribution VI computes the values *x* that have the chance of lying in a normally distributed sample for a given *p*. For example, you might want to find the heights of males that have a 60% chance of lying in a randomly chosen data set.

In addition to the inverse normal distribution method, you can use the following methods to compute *x* with a known *p*:

- Inverse Chi-Square distribution
- Inverse F distribution
- Inverse T distribution

Probability Distribution and Density Functions

Equation 10-7 defines the probability distribution function $F(x)$.

$$F(x) = \int_{-\infty}^{x} f(\mu)d\mu \tag{10-7}$$

where $f(x)$ is the probability density function, $f(x) \geq 0 \quad \forall x \in$ domain of f, and $\int_{-\infty}^{\infty} f(x)dx = 1$.

By performing differentiation, you can derive the following equation from Equation 10-7.

$$f(x) = \frac{dF(x)}{dx}$$

You can use a histogram to obtain a denormalized discrete representation of $f(x)$. The following equation defines the discrete representation of $f(x)$.

$$\sum_{i=0}^{n-1} x_i \Delta x = 1$$

The following equation yields the sum of the elements of the histogram.

$$\sum_{l=0}^{m-1} h_l = n$$

where m is the number of samples in the histogram and n is the number of samples in the input sequence representing the function.

Therefore, to obtain an estimate of $F(x)$ and $f(x)$, normalize the histogram by a factor of $\Delta x = 1/n$ and let $h_j = x_j$.

Figure 10-3 shows the block diagram of a VI that generates $F(x)$ and $f(x)$ for Gaussian white noise.

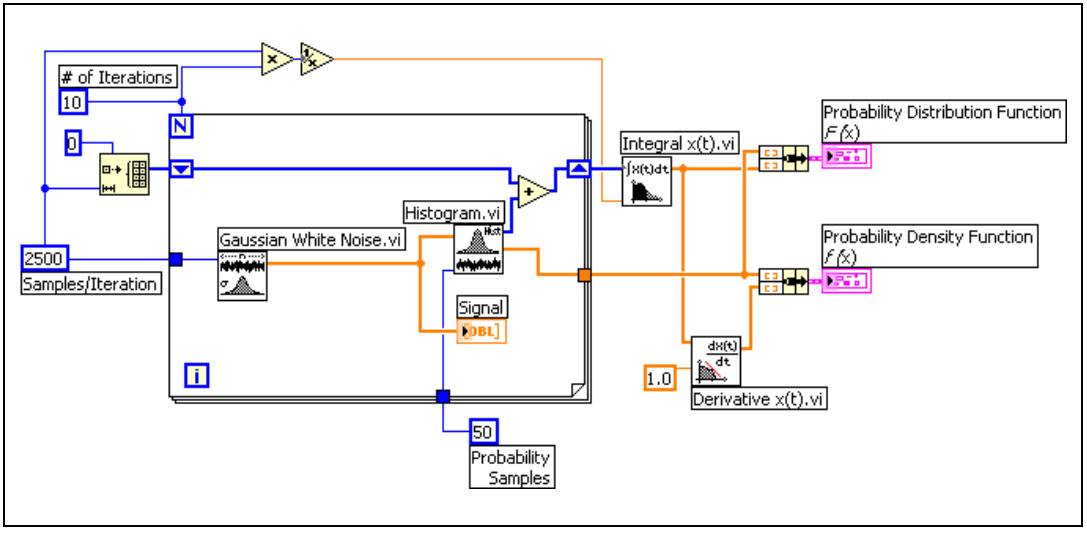


Figure 10-3. Generating Probability Distribution Function and Probability Density Function

The VI in Figure 10-3 uses 25,000 samples, 2,500 in each of the 10 loop iterations, to compute the probability distribution function for Gaussian white noise. The Integral x(t) VI computes the probability distribution function. The Derivative x(t) VI performs differentiation on the probability distribution function to compute the probability density function.

Figure 10-4 shows the results the VI in Figure 10-3 returns.

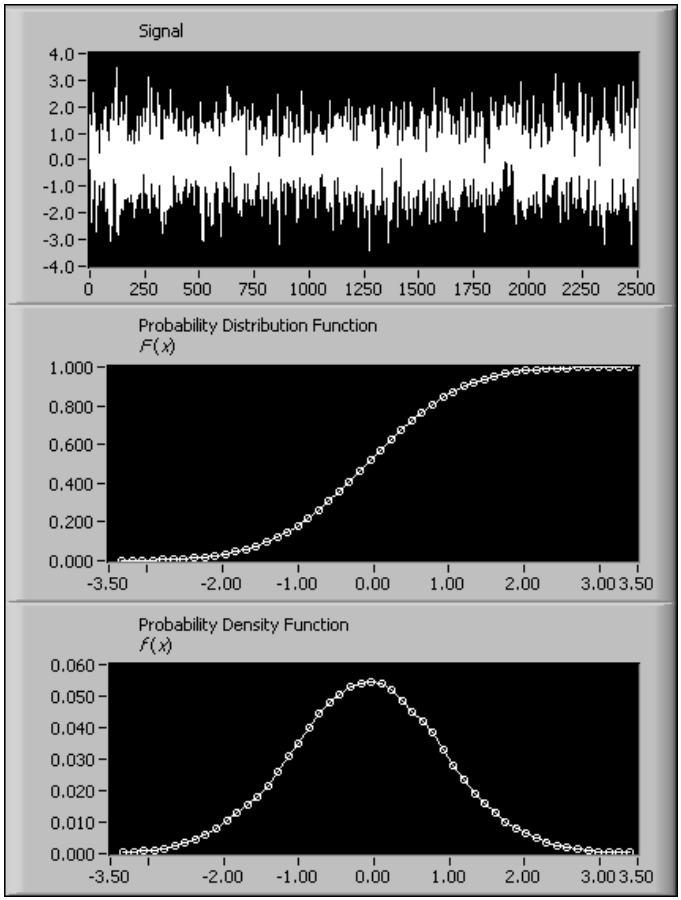


Figure 10-4. Input Signal, Probability Distribution Function, and Probability Density Function

Figure 10-4 shows the last block of Gaussian-distributed noise samples, the plot of the probability distribution function $F(x)$, and the plot of the probability density function $f(x)$. The plot of $F(x)$ monotonically increases and is limited to the maximum value of 1.00 as the value of the x-axis increases. The plot of $f(x)$ shows a Gaussian distribution that conforms to the specific pattern of the noise signal.

11

Linear Algebra

This chapter describes how to use the Linear Algebra VIs to perform matrix computation and analysis. Use the NI Example Finder to find examples of using the Linear Algebra VIs.

Linear Systems and Matrix Analysis

Systems of linear algebraic equations arise in many applications that involve scientific computations, such as signal processing, computational fluid dynamics, and others. Such systems occur naturally or are the result of approximating differential equations by algebraic equations.

Types of Matrices

Whatever the application, it is always necessary to find an accurate solution for the system of equations in a very efficient way. In matrix-vector notation, such a system of linear algebraic equations has the following form.

$$Ax = b$$

where A is an $n \times n$ matrix, b is a given vector consisting of n elements, and x is the unknown solution vector to be determined.

A matrix is a 2D array of elements with m rows and n columns. The elements in the 2D array might be real numbers, complex numbers, functions, or operators. The matrix A shown below is an array of m rows and n columns with $m \times n$ elements.

$$A = \begin{bmatrix} a_{0,0} & a_{0,1} & \cdots & a_{0,n-1} \\ a_{1,0} & a_{1,1} & \cdots & a_{1,n-1} \\ \cdots & \cdots & \cdots & \cdots \\ a_{m-1,0} & a_{m-1,1} & \cdots & a_{m-1,n-1} \end{bmatrix}$$

Here, $a_{i,j}$ denotes the $(i,j)^{\text{th}}$ element located in the i^{th} row and the j^{th} column. In general, such a matrix is a rectangular matrix. When $m = n$ so that the

number of rows is equal to the number of columns, the matrix is a square matrix. An $m \times 1$ matrix—m rows and one column—is a column vector. A row vector is a $1 \times n$ matrix—one row and n columns. If all the elements other than the diagonal elements are zero—that is, $a_{i,j} = 0$, $i \neq j$—such a matrix is a diagonal matrix. For example,

$$A = \begin{bmatrix} 4 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 9 \end{bmatrix}$$

is a diagonal matrix. A diagonal matrix with all the diagonal elements equal to one is an identity matrix, also known as unit matrix. If all the elements below the main diagonal are zero, the matrix is an upper triangular matrix. On the other hand, if all the elements above the main diagonal are zero, the matrix is a lower triangular matrix. When all the elements are real numbers, the matrix is a real matrix. On the other hand, when at least one of the elements of the matrix is a complex number, the matrix is a complex matrix.

Determinant of a Matrix

One of the most important attributes of a matrix is its determinant. In the simplest case, the determinant of a 2×2 matrix

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

is given by $ad - bc$. The determinant of a square matrix is formed by taking the determinant of its elements. For example, if

$$A = \begin{bmatrix} 2 & 5 & 3 \\ 6 & 1 & 7 \\ 1 & 6 & 9 \end{bmatrix}$$

then the determinant of A, denoted by $|A|$, is

$$|A| = \begin{vmatrix} 2 & 5 & 3 \\ 6 & 1 & 7 \\ 1 & 6 & 9 \end{vmatrix} = \left(2 \begin{vmatrix} 1 & 7 \\ 6 & 9 \end{vmatrix} - 5 \begin{vmatrix} 6 & 7 \\ 1 & 9 \end{vmatrix} + 3 \begin{vmatrix} 6 & 1 \\ 1 & 6 \end{vmatrix} \right) =$$

$$2(-33) - 5(47) + 3(35) = -196$$

The determinant of a diagonal matrix, an upper triangular matrix, or a lower triangular matrix is the product of its diagonal elements.

The determinant tells many important properties of the matrix. For example, if the determinant of the matrix is zero, the matrix is singular. In other words, the above matrix with nonzero determinant is nonsingular. Refer to the *Matrix Inverse and Solving Systems of Linear Equations* section of this chapter for more information about singularity and the solution of linear equations and matrix inverses.

Transpose of a Matrix

The transpose of a real matrix is formed by interchanging its rows and columns. If the matrix B represents the transpose of A, denoted by A^T, then $b_{j,i} = a_{i,j}$. For the matrix A defined above,

$$B = A^T = \begin{bmatrix} 2 & 6 & 1 \\ 5 & 1 & 6 \\ 3 & 7 & 9 \end{bmatrix}$$

In the case of complex matrices, we define complex conjugate transposition. If the matrix D represents the complex conjugate transpose (if $a = x + iy$, then complex conjugate $a^* = x - iy$) of a complex matrix C, then

$$D = C^H \Rightarrow d_{i,j} = c^*_{j,i}$$

That is, the matrix D is obtained by replacing every element in C by its complex conjugate and then interchanging the rows and columns of the resulting matrix.

A real matrix is a symmetric matrix if the transpose of the matrix is equal to the matrix itself. The example matrix A is not a symmetric matrix. If a complex matrix C satisfies the relation $C = C^H$, C is a Hermitian matrix.

Linear Independence

A set of vectors $x_1, x_2, \ldots, x_n$ is linearly dependent only if there exist scalars $\alpha_1, \alpha_2, \ldots, \alpha_n$, not all zero, such that

$$\alpha_1 x_1 + \alpha_2 x_2 + \ldots + \alpha_n x_n = 0 \qquad (11\text{-}1)$$

In simpler terms, if one of the vectors can be written in terms of a linear combination of the others, the vectors are linearly dependent.

If the only set of α_i for which Equation 11-1 holds is $\alpha_1 = 0, \alpha_2 = 0, \ldots, \alpha_n = 0$, the set of vectors $x_1, x_2, \ldots, x_n$ is linearly independent. So in this case, none of the vectors can be written in terms of a linear combination of the others. Given any set of vectors, Equation 11-1 always holds for $\alpha_1 = 0, \alpha_2 = 0, \ldots, \alpha_n = 0$. Therefore, to show the linear independence of the set, you must show that $\alpha_1 = 0, \alpha_2 = 0, \ldots, \alpha_n = 0$ is the only set of α_i for which Equation 11-1 holds.

For example, first consider the vectors

$$x = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \qquad y = \begin{bmatrix} 3 \\ 4 \end{bmatrix}$$

$\alpha_1 = 0$ and $\alpha_2 = 0$ are the only values for which the relation $\alpha_1 x + \alpha_2 y = 0$ holds true. Therefore, these two vectors are linearly independent of each other. Now consider the vectors

$$x = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \qquad y = \begin{bmatrix} 2 \\ 4 \end{bmatrix}$$

If $\alpha_1 = -2$ and $\alpha_2 = 1$, $\alpha_1 x + \alpha_2 y = 0$. Therefore, these two vectors are linearly dependent on each other. You must understand this definition of linear independence of vectors to fully appreciate the concept of the rank of the matrix.

Matrix Rank

The rank of a matrix A, denoted by $\rho(A)$, is the maximum number of linearly independent columns in A. If you look at the example matrix A, you find that all the columns of A are linearly independent of each other. That is, none of the columns can be obtained by forming a linear combination of the other columns. Hence, the rank of the matrix is 3. Consider one more example matrix, B, where

$$B = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 2 & 3 \\ 2 & 0 & 2 \end{bmatrix}$$

This matrix has only two linearly independent columns because the third column of *B* is linearly dependent on the first two columns. Hence, the rank of this matrix is 2. It can be shown that the number of linearly independent columns of a matrix is equal to the number of independent rows. So the rank can never be greater than the smaller dimension of the matrix. Consequently, if *A* is an $n \times m$ matrix, then

$$\rho(A) \leq min(n, m)$$

where *min* denotes the minimum of the two numbers. In matrix theory, the rank of a square matrix pertains to the highest order nonsingular matrix that can be formed from it. A matrix is singular if its determinant is zero. So the rank pertains to the highest order matrix that you can obtain whose determinant is not zero. For example, consider a 4×4 matrix

$$B = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 0 & 1 & -1 & 0 \\ 1 & 0 & 1 & 2 \\ 1 & 1 & 0 & 2 \end{bmatrix}$$

For this matrix, $det(B) = 0$, but

$$\left| \begin{bmatrix} 1 & 2 & 3 \\ 0 & 1 & -1 \\ 1 & 0 & 1 \end{bmatrix} \right| = -1$$

Hence, the rank of *B* is 3. A square matrix has full rank only if its determinant is different from zero. Matrix *B* is not a full-rank matrix.

Magnitude (Norms) of Matrices

You must develop a notion of the magnitude of vectors and matrices to measure errors and sensitivity in solving a linear system of equations. As an example, these linear systems can be obtained from applications in control systems and computational fluid dynamics. In two dimensions, for example, you cannot compare two vectors $x = [x1\ x2]$ and $y = [y1\ y2]$ because you might have $x1 > y1$ but $x2 < y2$. A vector norm is a way to assign a scalar quantity to these vectors so that they can be compared with each other. It is similar to the concept of magnitude, modulus, or absolute value for scalar numbers.

There are ways to compute the norm of a matrix. These include the 2-norm (Euclidean norm), the 1-norm, the Frobenius norm (F-norm), and the Infinity norm (inf-norm). Each norm has its own physical interpretation. Consider a unit ball containing the origin. The Euclidean norm of a vector is simply the factor by which the ball must be expanded or shrunk in order to encompass the given vector exactly, as shown in Figure 11-1.

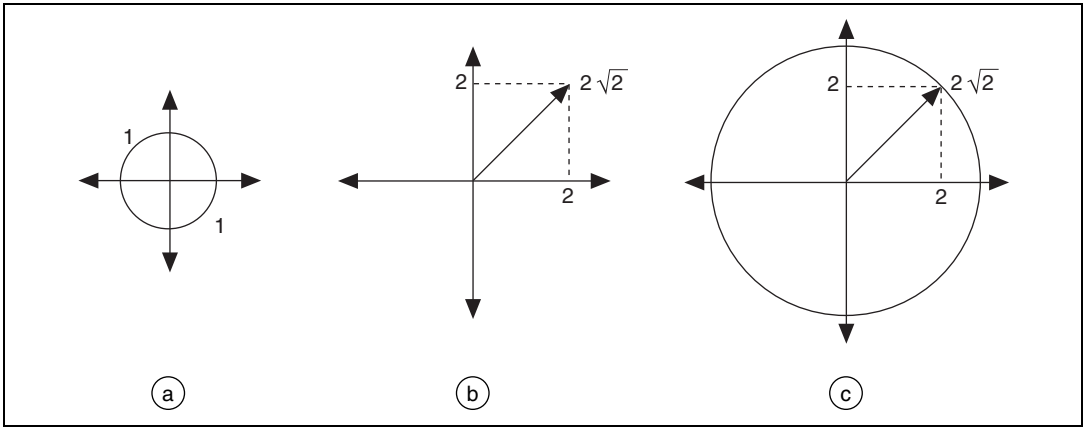


Figure 11-1. Euclidean Norm of a Vector

Figure 11-1a shows a unit ball of radius = 1 unit. Figure 11-1b shows a vector of length $\sqrt{2^2 + 2^2} = \sqrt{8} = 2\sqrt{2}$. As shown in Figure 11-1c, the unit ball must be expanded by a factor of $2\sqrt{2}$ before it can exactly encompass the given vector. Hence, the Euclidean norm of the vector is $2\sqrt{2}$.

The norm of a matrix is defined in terms of an underlying vector norm. It is the maximum relative stretching that the matrix does to any vector. With the vector 2-norm, the unit ball expands by a factor equal to the norm. On the other hand, with the matrix 2-norm, the unit ball might become an ellipsoidal (ellipse in 3D), with some axes longer than others. The longest axis determines the norm of the matrix.

Some matrix norms are much easier to compute than others. The 1-norm is obtained by finding the sum of the absolute value of all the elements in each column of the matrix. The largest of these sums is the 1-norm. In mathematical terms, the 1-norm is simply the maximum absolute column sum of the matrix.

$$\|A\|_1 = max_j \sum_{i=0}^{n-1} |a_{i,j}|$$

For example,

$$A = \begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix}$$

then

$$\|A\|_1 = max(3, 7) = 7$$

The inf-norm of a matrix is the maximum absolute row sum of the matrix.

$$\|A\|_\infty = max_i \sum_{j=0}^{n-1} |a_{i,j}| \qquad (11\text{-}2)$$

In this case, you add the magnitudes of all elements in each row of the matrix. The maximum value that you get is the inf-norm. For the Equation 11-2 example matrix,

$$\|A\|_\infty = max(4, 6) = 6$$

The 2-norm is the most difficult to compute because it is given by the largest singular value of the matrix. Refer to the *Matrix Factorization* section of this chapter for more information about singular values.

Determining Singularity (Condition Number)

Whereas the norm of the matrix provides a way to measure the magnitude of the matrix, the condition number of a matrix is a measure of how close the matrix is to being singular. The condition number of a square nonsingular matrix is defined as

$$cond(A) = \|A\|_p \cdot \|A^{-1}\|_p$$

where p can be one of the four norm types described in the *Magnitude (Norms) of Matrices* section of this chapter. For example, to find the condition number of a matrix A, you can find the 2-norm of A, the 2-norm of the inverse of the matrix A, denoted by A^{-1}, and then multiply them together. The inverse of a square matrix A is a square matrix B such that $AB = I$, where I is the identity matrix. As described earlier in this chapter,

the 2-norm is difficult to calculate on paper. You can use the Matrix Norm VI to compute the 2-norm. For example,

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}, A^{-1} = \begin{bmatrix} -2 & 1 \\ 1.5 & -0.5 \end{bmatrix}, \|A\|_2 = 5.4650, \|A^{-1}\|_2$$

$$= 2.7325, cond(A) = 14.9331$$

The condition number can vary between 1 and infinity. A matrix with a large condition number is nearly singular, while a matrix with a condition number close to 1 is far from being singular. The matrix *A* above is nonsingular. However, consider the matrix

$$B = \begin{bmatrix} 1 & 0.99 \\ 1.99 & 2 \end{bmatrix}$$

The condition number of this matrix is 47,168, and hence the matrix is close to being singular. A matrix is singular if its determinant is equal to zero. However, the determinant is not a good indicator for assessing how close a matrix is to being singular. For the matrix *B* above, the determinant (0.0299) is nonzero. However, the large condition number indicates that the matrix is close to being singular. Remember that the condition number of a matrix is always greater than or equal to one; the latter being true for identity and permutation matrices. A *permutation matrix* is an identity matrix with some rows and columns exchanged. The condition number is a very useful quantity in assessing the accuracy of solutions to linear systems.

Basic Matrix Operations and Eigenvalues-Eigenvector Problems

In this section, consider some very basic matrix operations. Two matrices, *A* and *B*, are equal if they have the same number of rows and columns and their corresponding elements all are equal. Multiplication of a matrix *A* by a scalar α is equal to multiplication of all its elements by the scalar. That is,

$$\boldsymbol{C} = \alpha \boldsymbol{A} \Rightarrow c_{i,j} = \alpha a_{i,j}$$

For example,

$$2\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix} = \begin{bmatrix}2 & 4\\ 6 & 8\end{bmatrix}$$

Two (or more) matrices can be added or subtracted only if they have the same number of rows and columns. If both matrices *A* and *B* have *m* rows and *n* columns, their sum *C* is an $m \times n$ matrix defined as $C = A \pm B$, where $c_{i,j} = a_{i,j} \pm b_{i,j}$. For example,

$$\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix} + \begin{bmatrix}2 & 4\\ 5 & 1\end{bmatrix} = \begin{bmatrix}3 & 6\\ 8 & 5\end{bmatrix}$$

For multiplication of two matrices, the number of columns of the first matrix must be equal to the number of rows of the second matrix. If matrix *A* has *m* rows and *n* columns and matrix *B* has *n* rows and *p* columns, their product *C* is an $m \times p$ matrix defined as $C = AB$, where

$$c_{i,j} = \sum_{k=0}^{n-1} a_{i,k} b_{k,j}$$

For example,

$$\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix} \times \begin{bmatrix}2 & 4\\ 5 & 1\end{bmatrix} = \begin{bmatrix}12 & 6\\ 26 & 16\end{bmatrix}$$

So you multiply the elements of the first row of *A* by the corresponding elements of the first column of *B* and add all the results to get the elements in the first row and first column of *C*. Similarly, to calculate the element in the i^{th} row and the j^{th} column of *C*, multiply the elements in the i^{th} row of *A* by the corresponding elements in the j^{th} column of *C*, and then add them all. This is shown pictorially in Figure 11-2.

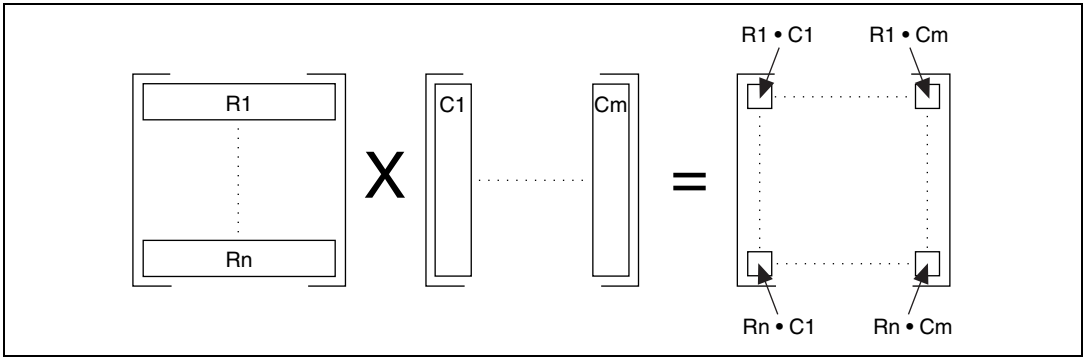


Figure 11-2. Matrix Multiplication

Matrix multiplication, in general, is not commutative, that is, $AB \neq BA$. Also, multiplication of a matrix by an identity matrix results in the original matrix.

Dot Product and Outer Product

If *X* represents a vector and *Y* represents another vector, the dot product of these two vectors is obtained by multiplying the corresponding elements of each vector and adding the results. This is denoted by

$$X \bullet Y = \sum_{i=0}^{n-1} x_i y_i$$

where *n* is the number of elements in *X* and *Y*. Both vectors must have the same number of elements. The dot product is a scalar quantity and has many practical applications.

For example, consider the vectors $a = 2i + 4j$ and $b = 2i + j$ in a two-dimensional rectangular coordinate system, as shown in Figure 11-3.

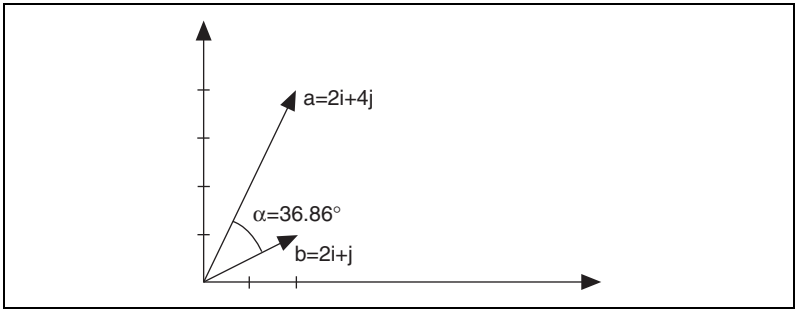


Figure 11-3. Vectors *a* and *b*

Then the dot product of these two vectors is given by

$$d = \begin{bmatrix} 2 \\ 4 \end{bmatrix} \bullet \begin{bmatrix} 2 \\ 1 \end{bmatrix} = (2 \times 2) + (4 \times 1) = 8$$

The angle α between these two vectors is given by

$$\alpha = inv\cos\left(\frac{a \bullet b}{|a||b|}\right) = inv\cos\left(\frac{8}{10}\right) = 36.86^{o}$$

where $|a|$ denotes the magnitude of a.

As a second application, consider a body on which a constant force a acts, as shown in Figure 11-4. The work W done by a in displacing the body is defined as the product of $|d|$ and the component of a in the direction of displacement d. That is,

$$W = |a||d|\cos\alpha = a \bullet d$$

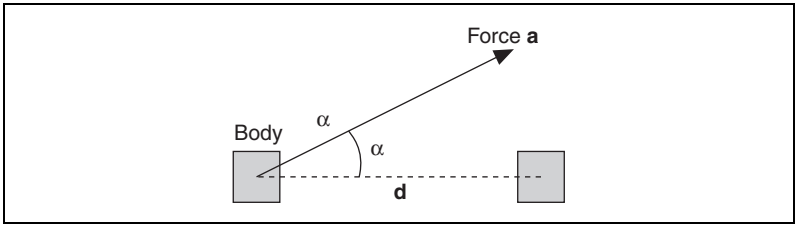


Figure 11-4. Force Vector

On the other hand, the outer product of these two vectors is a matrix. The $(i,j)^{th}$ element of this matrix is obtained using the formula

$$a_{(i,j)} = x_i \times y_j$$

For example,

$$\begin{bmatrix} 1 \\ 2 \end{bmatrix} \times \begin{bmatrix} 3 \\ 4 \end{bmatrix} = \begin{bmatrix} 3 & 4 \\ 6 & 8 \end{bmatrix}$$

Eigenvalues and Eigenvectors

To understand eigenvalues and eigenvectors, start with the classical definition. Given an $n \times n$ matrix A, the problem is to find a scalar λ and a nonzero vector x such that

$$Ax = \lambda x \tag{11-3}$$

In Equation 11-3, λ is an eigenvalue. Similar matrices have the same eigenvalues. In Equation 11-3, x is the eigenvector that corresponds to the eigenvalue. An eigenvector of a matrix is a nonzero vector that does not rotate when the matrix is applied to it.

Calculating the eigenvalues and eigenvectors are fundamental principles of linear algebra and allow you to solve many problems such as systems of differential equations when you understand what they represent. Consider an eigenvector x of a matrix A as a nonzero vector that does not rotate when x is multiplied by A, except perhaps to point in precisely the opposite direction. x may change length or reverse its direction, but it will not turn sideways. In other words, there is some scalar constant λ such that Equation 11-3 holds true. The value λ is an eigenvalue of A.

Consider the following example. One of the eigenvectors of the matrix A, where

$$A = \begin{bmatrix} 2 & 3 \\ 3 & 5 \end{bmatrix}$$

is

$$x = \begin{bmatrix} 0.62 \\ 1.00 \end{bmatrix}$$

Multiplying the matrix A and the vector x simply causes the vector x to be expanded by a factor of 6.85. Hence, the value 6.85 is one of the eigenvalues of the vector x. For any constant α, the vector αx also is an eigenvector with eigenvalue λ because

$$A(\alpha x) = \alpha Ax = \lambda \alpha x$$

In other words, an eigenvector of a matrix determines a direction in which the matrix expands or shrinks any vector lying in that direction by a scalar multiple, and the expansion or contraction factor is given by the corresponding eigenvalue. A generalized eigenvalue problem is to find a scalar λ and a nonzero vector x such that

$$Ax = \lambda Bx$$

where B is another $n \times n$ matrix.

The following are some important properties of eigenvalues and eigenvectors:

- The eigenvalues of a matrix are not necessarily all distinct. In other words, a matrix can have multiple eigenvalues.
- All the eigenvalues of a real matrix need not be real. However, complex eigenvalues of a real matrix must occur in complex conjugate pairs.
- The eigenvalues of a diagonal matrix are its diagonal entries, and the eigenvectors are the corresponding columns of an identity matrix of the same dimension.
- A real symmetric matrix always has real eigenvalues and eigenvectors.
- Eigenvectors can be scaled arbitrarily.

There are many practical applications in the field of science and engineering for an eigenvalue problem. For example, the stability of a structure and its natural modes and frequencies of vibration are determined by the eigenvalues and eigenvectors of an appropriate matrix. Eigenvalues also are very useful in analyzing numerical methods, such as convergence analysis of iterative methods for solving systems of algebraic equations and the stability analysis of methods for solving systems of differential equations.

The EigenValues and Vectors VI has an **Input Matrix** input, which is an $N \times N$ real square matrix. The **matrix type** input specifies the type of the input matrix. The **matrix type** input could be `0`, indicating a general matrix, or `1`, indicating a symmetric matrix. A symmetric matrix always has real

eigenvalues and eigenvectors. A general matrix has no special property such as symmetry or triangular structure.

The **output option** input specifies what needs to be computed. A value of `0` indicates that only the eigenvalues need to be computed. A value of `1` indicates that both the eigenvalues and the eigenvectors should be computed. It is computationally expensive to compute both the eigenvalues and the eigenvectors. So it is important that you use the **output option** input of the EigenValues and Vectors VI carefully. Depending on your particular application, you might just want to compute the eigenvalues or both the eigenvalues and the eigenvectors. Also, a symmetric matrix needs less computation than a nonsymmetric matrix. Choose the matrix type carefully.

Matrix Inverse and Solving Systems of Linear Equations

The inverse, denoted by A^{-1}, of a square matrix A is a square matrix such that

$$A^{-1}A = AA^{-1} = I$$

where I is the identity matrix. The inverse of a matrix exists only if the determinant of the matrix is not zero—that is, it is nonsingular. In general, you can find the inverse of only a square matrix. However, you can compute the pseudoinverse of a rectangular matrix. Refer to the *Matrix Factorization* section of this chapter for more information about the pseudoinverse of a rectangular matrix.

Solutions of Systems of Linear Equations

In matrix-vector notation, a system of linear equations has the form $Ax = b$, where A is an $n \times n$ matrix and b is a given n-vector. The aim is to determine x, the unknown solution n-vector. Whether such a solution exists and whether it is unique lies in determining the singularity or nonsingularity of the matrix A.

A matrix is singular if it has any one of the following equivalent properties:

- The inverse of the matrix does not exist.
- The determinant of the matrix is zero.
- The rows or columns of A are linearly dependent.
- $Az = 0$ for some vector $z \neq 0$.

Otherwise, the matrix is nonsingular. If the matrix is nonsingular, its inverse A^{-1} exists, and the system $Ax = b$ has a unique solution, $x = A^{-1}b$, regardless of the value for b.

On the other hand, if the matrix is singular, the number of solutions is determined by the right-hand-side vector b. If A is singular and $Ax = b$, $A(x + \Upsilon z) = b$ for any scalar Υ, where the vector z is as in the previous definition. Thus, if a singular system has a solution, the solution cannot be unique.

Explicitly computing the inverse of a matrix is prone to numerical inaccuracies. Therefore, you should not solve a linear system of equations by multiplying the inverse of the matrix A by the known right-hand-side vector. The general strategy to solve such a system of equations is to transform the original system into one whose solution is the same as that of the original system but is easier to compute. One way to do so is to use the Gaussian Elimination technique. The Gaussian Elimination technique has three basic steps. First, express the matrix A as a product

$$A = LU$$

where L is a unit lower triangular matrix and U is an upper triangular matrix. Such a factorization is LU factorization. Given this, the linear system $Ax = b$ can be expressed as $LUx = b$. Such a system then can be solved by first solving the lower triangular system $Ly = b$ for y by forward-substitution. This is the second step in the Gaussian Elimination technique. For example, if

$$l = \begin{bmatrix} a & 0 \\ b & c \end{bmatrix} \qquad y = \begin{bmatrix} p \\ q \end{bmatrix} \qquad b = \begin{bmatrix} r \\ s \end{bmatrix}$$

then

$$p = \frac{r}{a}, q = \frac{(s - bp)}{c}$$

The first element of y can be determined easily due to the lower triangular nature of the matrix L. Then you can use this value to compute the remaining elements of the unknown vector sequentially—hence the name forward-substitution. The final step involves solving the upper triangular system $Ux = y$ by back-substitution. For example, if

$$U = \begin{bmatrix} a & b \\ 0 & c \end{bmatrix} \qquad x = \begin{bmatrix} m \\ n \end{bmatrix} \qquad y = \begin{bmatrix} p \\ q \end{bmatrix}$$

then

$$n = \frac{q}{c}, m = \frac{(p - bn)}{a}$$

In this case, this last element of x can be determined easily and then used to determine the other elements sequentially—hence the name back-substitution. So far, this chapter has described the case of square matrices. Because a nonsquare matrix is necessarily singular, the system of equations must have either no solution or a nonunique solution. In such a situation, you usually find a unique solution x that satisfies the linear system in an approximate sense.

You can use the Linear Algebra VIs to compute the inverse of a matrix, compute LU decomposition of a matrix, and solve a system of linear equations. It is important to identify the input matrix properly, as it helps avoid unnecessary computations, which in turn helps to minimize numerical inaccuracies. The four possible matrix types are general matrices, positive definite matrices, and lower and upper triangular matrices. A real matrix is positive definite only if it is symmetric, and if the quadratic form for all nonzero vectors is X. If the input matrix is square but does not have a full rank (a rank-deficient matrix), the VI finds the least square solution x. The least square solution is the one that minimizes the norm of $Ax - b$. The same also holds true for nonsquare matrices.

Matrix Factorization

The *Matrix Inverse and Solving Systems of Linear Equations* section of this chapter describes how a linear system of equations can be transformed into a system whose solution is simpler to compute. The basic idea was to factorize the input matrix into the multiplication of several, simpler matrices. The LU decomposition technique factors the input matrix as a product of upper and lower triangular matrices. Other commonly used factorization methods are Cholesky, QR, and the Singular Value

Decomposition (SVD). You can use these factorization methods to solve many matrix problems, such as solving linear system of equations, inverting a matrix, and finding the determinant of a matrix.

If the input matrix A is symmetric and positive definite, an LU factorization can be computed such that $A = U^T U$, where U is an upper triangular matrix. This is Cholesky factorization. This method requires only about half the work and half the storage compared to LU factorization of a general matrix by Gaussian Elimination. You can determine if a matrix is positive definite by using the Test Positive Definite VI.

A matrix Q is orthogonal if its columns are orthonormal—that is, if $Q^T Q = I$, the identity matrix. QR factorization technique factors a matrix as the product of an orthogonal matrix Q and an upper triangular matrix R—that is, $A = QR$. QR factorization is useful for both square and rectangular matrices. A number of algorithms are possible for QR factorization, such as the Householder transformation, the Givens transformation, and the Fast Givens transformation.

The Singular Value Decomposition (SVD) method decomposes a matrix into the product of three matrices—$A = USV^T$. U and V are orthogonal matrices. S is a diagonal matrix whose diagonal values are called the singular values of A. The singular values of A are the nonnegative square roots of the eigenvalues of $A^T A$, and the columns of U and V, which are called left and right singular vectors, are orthonormal eigenvectors of AA^T and $A^T A$, respectively. SVD is useful for solving analysis problems such as computing the rank, norm, condition number, and pseudoinverse of matrices.

Pseudoinverse

The pseudoinverse of a scalar σ is defined as $1/\sigma$ if $\sigma \neq 0$, and zero otherwise. In case of scalars, pseudoinverse is the same as the inverse. You now can define the pseudoinverse of a diagonal matrix by transposing the matrix and then taking the scalar pseudoinverse of each entry. Then the pseudoinverse of a general real $m \times n$ matrix A, denoted by $A^\dagger$, is given by

$$A^\dagger = VS^\dagger U^T$$

The pseudoinverse exists regardless of whether the matrix is square or rectangular. If A is square and nonsingular, the pseudoinverse is the same as the usual matrix inverse. You can use the PseudoInverse Matrix VI to compute the pseudoinverse of real and complex matrices.

12

Optimization

This chapter describes basic concepts and methods used to solve optimization problems. Refer to Appendix A, *References*, for a list of references to more information about optimization.

Introduction to Optimization

Optimization is the search for a set of parameters that minimize a function. For example, you can use optimization to define an optimal set of parameters for the design of a specific application, such as the optimal parameters for designing a control mechanism for a system or the conditions that minimize the cost of a manufacturing process. Generally, optimization problems involve a set of possible solutions X and the objective function $f(x)$, also known as the cost function. $f(x)$ is the function of the variable or variables you want to minimize or maximize.

The optimization process either minimizes or maximizes $f(x)$ until reaching the optimal value for $f(x)$. When minimizing $f(x)$, the optimal solution $x^* \in X$ satisfies the following condition.

$$f(x^*) \leq f(x) \qquad \forall x \in X \tag{12-1}$$

The optimization process searches for the value of x^* that minimizes $f(x)$, subject to the constraint $x^* \in X$, where X is the constraint set. A value that satisfies the conditions defined in Equation 12-1 is a global minimum. Refer to the *Local and Global Minima* section of this chapter for more information about global minima.

In the case of maximization, x^* satisfies the following condition.

$$f(x^*) \geq f(x) \ \ \forall x \in X$$

A value satisfying the preceding condition is a global maximum. This chapter describes optimization in terms of minimizing $f(x)$.

Constraints on the Objective Function

The presence and structure of any constraints on the value of $f(x)$ influence the selection of the algorithm you use to solve an optimization problem. Certain algorithms solve only unconstrained optimization problems. If the value of $f(x)$ has any of the following constraints, the optimal value of $f(x)$ must satisfy the condition the constraint defines:

- Equality constraints, such as $G_i(x) = 4$ $(i = 1, \ldots, m_e)$
- Inequality constraints, such as $G_i(x) \leq 4$ $(i = m_e + 1, \ldots, m)$
- Lower and upper level boundaries, such as x_l, x_u

Note Currently, LabVIEW does not include VIs you can use to solve optimization problems in which the value of the objective function has constraints.

Linear and Nonlinear Programming Problems

The most common method of categorizing optimization problems is as either a linear programming problem or a nonlinear programming problem. In addition to constraints on the value of $f(x)$, whether an optimization problem is linear or nonlinear influences the selection of the algorithm you use to solve the problem.

Note In the context of optimization, the term *programming* does not refer to computer programming. Programming also refers to scheduling or planning. Linear and nonlinear programming are subsets of mathematical programming. The objective of mathematical programming is the same as optimization—maximizing or minimizing $f(x)$.

Discrete Optimization Problems

Linear programming problems are discrete optimization problems. A finite solution set X and a combinatorial nature characterize discrete optimization problems. A combinatorial nature refers to the fact that several solutions to the problem exist. Each solution to the problem consists of a different combination of parameters. However, at least one optimal solution exists. Planning a route to several destinations so you travel the minimum distance typifies a combinatorial optimization problem.

Continuous Optimization Problems

Nonlinear programming problems are continuous optimization problems. An infinite and continuous set X characterizes continuous optimization problems.

Solving Problems Iteratively

Algorithms for solving optimization problems use an iterative process. Beginning at a user-specified starting point, the algorithms establish a search direction. Each iteration of the algorithm proceeds along the search direction to the optimal solution by solving subproblems. Finding the optimal solution terminates the iterative process of the algorithm.

As the number of design variables increases, the complexity of the optimization problem increases. As problem complexity increases, computational overhead increases due to the size and number of subproblems the optimization algorithm must solve to find the optimal solution. Because of the computational overhead associated with highly complex problems, consider limiting the number of iterations allocated to find the optimal solution. Use the **accuracy** input of the Optimization VIs to specify the accuracy of the optimal solution.

Linear Programming

Linear programming problems have the following characteristics:

- Linear objective function
- Solution set X with a polyhedron shape defined by linear inequality constraints
- Continuous $f(x)$
- Partially combinatorial structure

Solving linear programming problems involves finding the optimal value of $f(x)$ where $f(x)$ is a linear combination of variables, as shown in Equation 12-2.

$$f(x) = a_1x_1 + \ldots + a_nx_n \tag{12-2}$$

The value of $f(x)$ in Equation 12-2 can have the following constraints:

- Primary constraints of $x_1 \geq 0, \ldots, x_n \geq 0$
- Additional constraints of $M = m_1 + m_2 + m_3$
- m_1 of the following form

$$a_{i1}x_1 + \ldots + a_{in}x_n \leq b_i, (b_i \geq 0), i = 1, \ldots, m_1$$

- m_2 of the following form

$$a_{j1}x_1 + \ldots + a_{jn}x_n \geq b_j, (b_j \geq 0), j = m_1 + 1, \ldots, m_1 + m_2$$

- m_3 of the following form

$$a_{k1}x_1 + \ldots + a_{kn}x_n = b_k, (b_k \geq 0), k = m_1 + m_2 + 1, \ldots, M$$

Any vector x that satisfies all the constraints on the value of $f(x)$ constitutes a feasible answer to the linear programming problem. The vector yielding the best result for $f(x)$ is the optimal solution.

The following relationship represents the standard form of the linear programming problem.

$$\min\{c^T x : Ax = b, x \geq 0\}$$

where $x \in \mathbb{R}^n$ is the vector of unknowns, $c \in \mathbb{R}^n$ is the cost vector, and $A \in \mathbb{R}^{m \times n}$ is the constraint matrix. At least one member of solution set X is at a vertex of the polyhedron that describes X.

Linear Programming Simplex Method

A simplex describes the solution set X for a linear programming problem. The constraints on the value of $f(x)$ define the polygonal surface hyperplanes of the simplex. The hyperplanes intersect at vertices along the surface of the simplex. The linear nature of $f(x)$ means the optimal solution is at one of the vertices of the simplex. The linear programming simplex method iteratively moves from one vertex to the adjoining vertex until moving to an adjoining vertex no longer yields a more optimal solution.

Note Although both the linear programming simplex method and the nonlinear downhill simplex method use the concept of a simplex, the methods have nothing else in common. Refer to the *Downhill Simplex Method* section of this chapter for information about the downhill simplex method.

Nonlinear Programming

Nonlinear programming problems have either a nonlinear $f(x)$ or a solution set X defined by nonlinear equations and inequalities. Nonlinear programming is a broad category of optimization problems and includes the following subcategories:

- Quadratic programming problems
- Least-squares problems
- Convex problems

Impact of Derivative Use on Search Method Selection

When you select a search method, consider whether the method uses derivatives, which can help you determine the suitability of the method for a particular optimization problem. For example, the downhill simplex method, also known as the Nelder-Mead method, uses only evaluations of $f(x)$ to find the optimal solution. Because it uses only evaluations of $f(x)$, the downhill simplex method is a good choice for problems with pronounced nonlinearity or with problems containing a significant number of discontinuities.

The search methods that use derivatives, such as the gradient search methods, work best with problems in which the objective function is continuous in its first derivative.

Line Minimization

The process of iteratively searching along a vector for the minimum value on the vector is line minimization or line searching. Line minimization can help establish a search direction or verify that the chosen search direction is likely to produce an optimal solution.

Nonlinear programming search algorithms use line minimization to solve the subproblems leading to an optimal value for $f(x)$. The search algorithm searches along a vector until it reaches the minimum value on the vector. After the search algorithm reaches the minimum on one vector, the search continues along another vector, usually orthogonal to the first vector. The line search continues along the new vector until reaching its minimum value. The line minimization process continues until the search algorithm finds the optimal solution.

Local and Global Minima

The goal of any optimization problem is to find a global optimal solution. However, nonlinear programming problems are continuous optimization problems so the solution set X for a nonlinear programming problem might be infinitely large. Therefore, you might not be able to find a global optimum for $f(x)$. In practice, you solve most nonlinear programming problems by finding a local optimum for $f(x)$.

Global Minimum

In terms of solution set X, x^* is a global minimum of f over X if it satisfies the following relationship.

$$f(x^*) \leq f(x) \qquad \forall x \in X$$

Local Minimum

A local minimum is a minimum of the function over a subset of the domain. In terms of solution set X, x^* is a local minimum of f over X if $x^* \in X$, and an $\varepsilon > 0$ exists so that the following relationship is true.

$$f(x^*) \leq f(x) \qquad \forall x \in X \text{ with } \|x - x^*\| < \varepsilon$$

where $\|x\| = \sqrt{x'x}$.

Figure 12-1 illustrates a function of x where the domain is any value between 32 and 65; $x \in [32, 65]$.

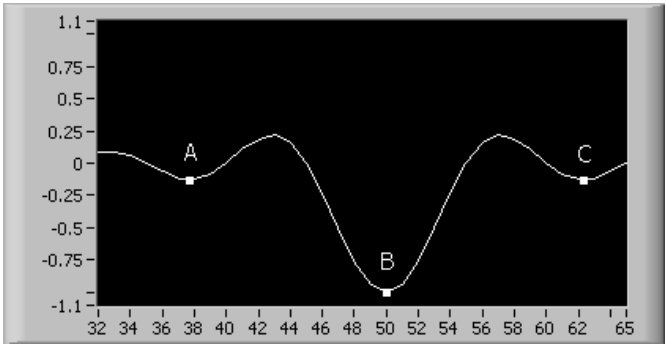


Figure 12-1. Domain of X (32, 65)

In Figure 12-1, A is a local minimum because you can find $\varepsilon > 0$, such that $f(x^*) \leq f(x)$. $\varepsilon = 1$ would suffice. Similarly, C is a local minimum. B is the global minimum because $f(x^*) \leq x$ for $\forall x \in [32, 65]$.

Downhill Simplex Method

The downhill simplex method developed by Nelder and Mead uses a simplex and performs function evaluations without derivatives.

Note Although the downhill simplex method and the linear programming simplex method use the concept of a simplex, the methods have nothing else in common. Refer to the *Linear*

Programming Simplex Method section of this chapter for information about the linear programming simplex method and the geometry of the simplex.

Most practical applications involve solution sets that are nondegenrate simplexes. A nondegenrate simplex encloses a finite volume of N dimensions. If you take any point of the nondegenrate simplex as the origin of the simplex, the remaining N points of the simplex define vector directions spanning the N-dimensional space.

The downhill simplex method requires that you define an initial simplex by specifying $N + 1$ starting points. No effective means of determining the initial starting point exists. You must use your judgement about the best location from which to start. After deciding upon an initial starting point P_0, you can use Equation 12-3 to determine the other points needed to define the initial simplex.

$$P_i = P_0 + \lambda e_i \tag{12-3}$$

where e_i is a unit vector and λ is an estimate of the characteristic length scale of the problem.

Starting with the initial simplex defined by the points from Equation 12-3, the downhill simplex method performs a series of reflections. A reflection moves from a point on the simplex through the opposite face of the simplex to a point where the function f is smaller. The configuration of the reflections conserves the volume of the simplex, which maintains the nondegeneracy of the simplex. The method continues to perform reflections until the function value reaches a predetermined tolerance.

Because of the multidimensional nature of the downhill simplex method, the value it finds for $f(x)$ might not be the optimal solution. You can verify that the value for $f(x)$ is the optimal solution by repeating the process. When you repeat the process, use the optimal solution from when you first ran the method as P_0. Reinitialize the method to $N + 1$ starting points using Equation 12-3.

Golden Section Search Method

The golden section search method finds a local minimum of a 1D function by bracketing the minimum. Bracketing a minimum requires a triplet of points, as shown in the following relationship.

$$a < b < c \text{ such that } f(b) < f(a) \text{ and } f(b) < f(c) \tag{12-4}$$

Because the relationship in Equation 12-4 is true, the minimum of the function is within the interval (a, c). The search method starts by choosing a new point x between either a and b or between b and c. For example, choose a point x between b and c and evaluate $f(x)$. If $f(b) < f(x)$, the new bracketing triplet is $a < b < x$. If $f(b) > f(x)$, the new bracketing triplet is $b < x < c$. In each instance, the middle point, b or x, is the current optimal minimum found during the current iteration of the search.

Choosing a New Point *x* in the Golden Section

Given that $a < b < c$, point b is a fractional distance W between a and c, as shown in the following equations.

$$\frac{b-a}{c-a} = W \qquad \frac{c-b}{c-a} = 1 - W$$

A new point x is an additional fractional distance Z beyond b, as shown in Equation 12-5.

$$\frac{x-b}{c-a} = Z \tag{12-5}$$

Given Equation 12-5, the next bracketing triplet can have either a length of $W + Z$ relative to the current bracketing triplet or a length of $1 - W$. To minimize the possible worst case, choose Z such that the following equations are true.

$$W + Z = 1 - W$$

$$Z = 1 - 2W \tag{12-6}$$

Given Equation 12-6, the new x is the point in the interval symmetric to b. Therefore, Equation 12-7 is true.

$$|b - a| = |x - c| \tag{12-7}$$

You can imply from Equation 12-7 that x is within the larger segment because Z is positive only if $W < 1/2$.

If Z is the current optimal value of $f(x)$, W is the previous optimal value of $f(x)$. Therefore, the fractional distance of x from b to c equals the fractional distance of b from a to c, as shown in Equation 12-8.

$$\frac{Z}{1-W} = W \tag{12-8}$$

Equations 12-6 and 12-8 yield the following quadratic equation.

$$W^2 - 3W + 1 = 0$$

$$W = \frac{3 - \sqrt{5}}{2} \approx 0.38197 \qquad (12\text{-}9)$$

Therefore, the middle point b of the optimal bracketing interval $a < b < c$ is the fractional distance of 0.38197 from one of the end points and the fractional distance of 0.61803 from the other end point. 0.38197 and 0.61803 comprise the golden mean, or golden section, of the Pythagoreans.

The golden section search method uses a bracketing triplet and measures from point b to find a new point x a fractional distance of 0.38197 into the larger interval, either (a, b) or (b, c), on each iteration of the search method. Even when starting with an initial bracketing triplet whose segments are not within the golden section, the process of successively choosing a new point x at the golden mean quickly causes the method to converge linearly to the correct, self-replicating golden section. After the search method converges to the self-replicating golden section, each new function evaluation brackets the minimum to an interval only 0.61803 times the size of the preceding interval.

Gradient Search Methods

Gradient search methods determine a search direction by using information about the slope of $f(x)$. The search direction points toward the most probable location of the minimum. After the gradient search method establishes the search direction, it uses iterative descent to move toward the minimum.

The iterative descent process starts at a point x_0, which is an estimate of the best starting point, and successively produces vectors $x_1, x_2, \ldots$, so f decreases with each iteration, as shown in the following relationship.

$$f(x_{k+1}) < f(x_k), \qquad k = 0, 1, \ldots$$

where k is the iteration number, $f(x_{k+1})$ is the objective function value at iteration $k + 1$, and $f(x_k)$ is the objective function value at iteration k.

Successively decreasing f improves the current estimate of the solution. The iterative descent process attempts to decrease f to its minimum.

The following equations and relationships provide a general definition of the gradient search method of solving nonlinear programming problems.

$$x_{k+1} = x_k + \alpha_k d_k \qquad k = 0, 1, \ldots \tag{12-10}$$

where d_k is the search direction and α_k is the step size.

In Equation 12-10, if the gradient of the objective function $\nabla f(x_k) \neq 0$, the gradient search method needs a positive value for α_k and a value for d_k that fulfills the following relationship.

$$\nabla f(x_k)' d_k < 0$$

Iterations of gradient search methods continue until $x_{k+1} = x_k$.

Caveats about Converging to an Optimal Solution

A global minimum is a value for $f(x)$ that satisfies the relationship described in Equation 12-1.

Ideally, iteratively decreasing f converges to a global minimum for $f(x)$. In practice, convergence rarely proceeds to a global minimum for $f(x)$ because of the presence of local minima that are not global. Local minima attract gradient search methods because the form of f near the current iterate and not the global structure of f determines the downhill course the method takes.

When a gradient search method begins on or encounters a stationary point, the method stops at the stationary point. Therefore, a gradient search method converges to a stationary point. If f is convex, the stationary point is a global minimum. However, if f is not convex, the stationary point might not be a global minimum. Therefore, if you have little information about the locations of local minima, you might have to start the gradient search method from several starting points.

Terminating Gradient Search Methods

Because a gradient search method does not produce convergence at a global minimum, you must decide upon an error tolerance ε that assures that the point at which the gradient search method stops is at least close to a local minimum. Unfortunately, no explicit rules exist for determining an absolutely accurate ε. The selection of a value for ε is somewhat arbitrary and based on an estimate about the value of the optimal solution.

Use the **accuracy** input of the Optimization VIs to specify a value for ε. The nonlinear programming optimization VIs iteratively compare the difference between the highest and lowest input values to the value of **accuracy** until two consecutive approximations do not differ by more than the value of **accuracy**. When two consecutive approximations do not differ by more than the value of **accuracy**, the VI stops.

Conjugate Direction Search Methods

Conjugate direction methods attempt to find $f(x)$ by defining a direction set of vectors such that minimizing along one vector does not interfere with minimizing along another vector, which prevents indefinite cycling through the direction set.

When you minimize a function f along direction u, the gradient of f is perpendicular to u at the line minimum. If P is the origin of a coordinate system with coordinates x, you can approximate f by the Taylor series of f, as shown in Equation 12-11.

$$f(x) = f(P) + \sum_i \frac{\partial f}{\partial x_i} x_i + \frac{1}{2} \sum_{i,\,j} \frac{\partial^2 f}{\partial x_i \partial x_j} x_i x_j + \ldots \approx c - bx + \frac{1}{2} xAx \tag{12-11}$$

where

$$c \equiv f(P)\,,\; b \equiv -\nabla f\big|_P\,,$$

and

$$[A]_{ij} \equiv \left.\frac{\partial^2 f}{\partial x_i \partial x_j}\right|_P$$

The components of matrix A are the second partial derivative matrix of f. Matrix A is the Hessian matrix of f at P.

The following equation gives the gradient of f.

$$\nabla f = Ax - b$$

The following equation shows how the gradient ∇f changes with movement along A.

$$\delta(\nabla f) = A(\delta x)$$

After the search method reaches the minimum by moving in direction u, it moves in a new direction v. To fulfill the condition that minimization along one vector does not interfere with minimization along another vector, the gradient of f must remain perpendicular to u, as shown in Equation 12-12.

$$0 = u\delta(\nabla f) = uAv \quad (12\text{-}12)$$

When Equation 12-12 is true for two vectors u and v, u and v are conjugate vectors. When Equation 12-12 is true pairwise for all members of a set of vectors, the set of vectors is a conjugate set. Performing successive line minimizations of a function along a conjugate set of vectors prevents the search method from having to repeat the minimization along any member of the conjugate set.

If a conjugate set of vectors contains N linearly independent vectors, performing N line minimizations arrives at the minimum for functions having the quadratic form shown in Equation 12-11. If a function does not have exactly the form of Equation 12-11, repeated cycles of N line minimizations eventually converge quadratically to the minimum.

Conjugate Gradient Search Methods

At an N-dimensional point P, the conjugate gradient search methods calculate the function $f(P)$ and the gradient $\nabla f(P)$. $\nabla f(P)$ is the vector of first partial derivatives. The conjugate gradient search method attempts to find $f(x)$ by searching a gradient conjugate to the previous gradient and conjugating to all previous gradients, as much as possible.

The Fletcher-Reeves method and the Polak-Ribiere method are the two most common conjugate gradient search methods. The following theorems serve as the basis for each method.

Theorem A

Theorem A has the following conditions:

- A is a symmetric, positive-definite, $n \times n$ matrix.
- g_0 is an arbitrary vector.
- $h_0 = g_0$.

- The following equations define the two sequences of vectors for $i = 0, 1, 2, \ldots$.

$$g_{i+1} = g_i - \lambda_i A h_i \tag{12-13}$$

$$h_{i+1} = g_{i+1} + \gamma_i h_i \tag{12-14}$$

where the chosen values for λ_i and γ_i make $g_{i+1} g_i = 0$ and $h_{i+1} A h_i = 0$, as shown in the following equations.

$$\gamma_i = \frac{g_{i+1} A h_i}{h_i A h_i} \tag{12-15}$$

$$\lambda_i = \frac{g_i g_i}{g_i A h_i} \tag{12-16}$$

If the denominators equal zero, take $\lambda_i = 0$, $\gamma_i = 0$.

- The following equations are true for all $i \neq j$.

$$g_i g_j = 0 \qquad h_i A h_j = 0 \tag{12-17}$$

The elements in the sequence that Equation 12-13 produces are mutually orthogonal. The elements in the sequence that Equation 12-14 produces are mutually conjugate.

Because Equation 12-17 is true, you can rewrite Equations 12-15 and 12-16 as the following equations.

$$\gamma_i = \frac{g_{i+1} \cdot g_{i+1}}{g_i \cdot g_i} = \frac{(g_{i+1} - g_i) \cdot g_{i+1}}{g_i \cdot g_i} \tag{12-18}$$

$$\lambda_i = \frac{g_i \cdot h_i}{h_i \cdot A \cdot h_i}$$

Theorem B

The following theorem defines a method for constructing the vector from Equation 12-13 when the Hessian matrix A is unknown:

- g_i is the vector sequence defined by Equation 12-13.
- h_i is the vector sequence defined by Equation 12-14.

- Approximate f as the quadratic form given by the following relationship.

$$f(x) \approx c - b \cdot x + \frac{1}{2} x \cdot A \cdot x$$

- $g_i = -\nabla f(P_i)$ for some point P_i.
- Proceed from P_i in the direction h_i to the local minimum of f at point P_{i+1}.
- Set the value for g_{i+1} according to Equation 12-19.

$$g_{i+1} = -\nabla f(P_{i+1}) \tag{12-19}$$

The vector g_{i+1} that Equation 12-19 yields is the same as the vector that Equation 12-13 yields when the Hessian matrix A is known. Therefore, you can optimize f without having knowledge of Hessian matrix A and without the computational resources to calculate and store the Hessian matrix A. You construct the direction sequence h_i with line minimization of the gradient vector and the latest vector in the g sequence.

Difference between Fletcher-Reeves and Polak-Ribiere

Both the Fletcher-Reeves method and the Polak-Ribiere method use Theorem A and Theorem B. However, the Fletcher-Reeves method uses the first term from Equation 12-18 for γ_i, as shown in Equation 12-20.

$$\gamma_i = \frac{g_{i+1} \cdot g_{i+1}}{g_i \cdot g_i} \tag{12-20}$$

The Polak-Ribiere method uses the second term from Equation 12-18 for γ_i, as shown in Equation 12-21.

$$\gamma_i = \frac{(g_{i+1} - g_i) \cdot g_{i+1}}{g_i \cdot g_i} \tag{12-21}$$

Equation 12-20 equals Equation 12-21 for functions with exact quadratic forms. However, most functions in practical applications do not have exact quadratic forms. Therefore, after you find the minimum for the quadratic form, you might need another set of iterations to find the actual minimum.

When the Polak-Ribiere method reaches the minimum for the quadratic form, it resets the direction *h* along the local gradient, essentially starting the conjugate-gradient process again. Therefore, the Polak-Ribiere method can make the transition to additional iterations more efficiently than the Fletcher-Reeves method.

13

Polynomials

Polynomials have many applications in various areas of engineering and science, such as curve fitting, system identification, and control design. This chapter describes polynomials and operations involving polynomials.

General Form of a Polynomial

A univariate polynomial is a mathematical expression involving a sum of powers in one variable multiplied by coefficients. Equation 13-1 shows the general form of an *n*th-order polynomial.

$$P(x) = a_0 + a_1x + a_2x^2 + \ldots + a_nx^n \tag{13-1}$$

where $P(x)$ is the *n*th-order polynomial, the highest power n is the order of the polynomial if $a_n \neq 0$, $a_0, a_1, \ldots, a_n$ are the constant coefficients of the polynomial and can be either real or complex.

You can rewrite Equation 13-1 in its factored form, as shown in Equation 13-2.

$$P(x) = a_n(x - r_1)(x - r_2) \ldots (x - r_n) \tag{13-2}$$

where $r_1, r_2, \ldots, r_n$ are the roots of the polynomial.

The root r_i of $P(x)$ satisfies the following equation.

$$P(x)\big|_{x = r_i} = 0 \qquad i = 1, 2, \ldots, n$$

In general, $P(x)$ might have repeated roots, such that Equation 13-3 is true.

$$P(x) = a_n(x - r_1)^{k_1}(x - r_2)^{k_2}\ldots(x - r_l)^{k_l}(x - r_{l+1})(x - r_{l+2})\ldots(x - r_{l+j}) \tag{13-3}$$

The following conditions are true for Equation 13-3:

- $r_1, r_2, \ldots, r_l$ are the repeated roots of the polynomial
- k_i is the multiplicity of the root r_i, $i = 1, 2, \ldots, l$

- $r_{l+1}, r_{l+2}, \ldots, r_{l+j}$ are the non-repeated roots of the polynomial
- $k_1 + k_2 + \ldots + k_l + j = n$

A polynomial of order n must have n roots. If the polynomial coefficients are all real, the roots of the polynomial are either real or complex conjugate numbers.

Basic Polynomial Operations

The basic polynomial operations include the following operations:

- Finding the order of a polynomial
- Evaluating a polynomial
- Adding, subtracting, multiplying, or dividing polynomials
- Determining the composition of a polynomial
- Determining the greatest common divisor of two polynomials
- Determining the least common multiple of two polynomials
- Calculating the derivative of a polynomial
- Integrating a polynomial
- Finding the number of real roots of a real polynomial

The following equations define two polynomials used in the following sections.

$$P(x) = a_0 + a_1x + a_2x^2 + a_3x^3 = a_3(x - p_1)(x - p_2)(x - p_3) \quad (13\text{-}4)$$

$$Q(x) = b_0 + b_1x + b_2x^2 = b_2(x - q_1)(x - q_2) \quad (13\text{-}5)$$

Order of Polynomial

The largest exponent of the variable determines the order of a polynomial. The order of $P(x)$ in Equation 13-4 is three because of the variable x^3. The order of $Q(x)$ in Equation 13-5 is two because of the variable x^2.

Polynomial Evaluation

Polynomial evaluation determines the value of a polynomial for a particular value of x, as shown by the following equation.

$$P(x)\big|_{x = x_0} = a_0 + a_1x_0 + a_2x_0^2 + a_3x_0^3 = a_0 + x_0(a_1 + x_0(a_2 + x_0a_3))$$

Evaluating an *n*th-order polynomial requires *n* multiplications and *n* additions.

Polynomial Addition

The addition of two polynomials involves adding together coefficients whose variables have the same exponent. The following equation shows the result of adding together the polynomials defined by Equations 13-4 and 13-5.

$$P(x) + Q(x) = (a_o + b_0) + (a_1 + b_1)x + (a_2 + b_2)x^2 + a_3x^3$$

Polynomial Subtraction

Subtracting one polynomial from another involves subtracting coefficients whose variables have the same exponent. The following equation shows the result of subtracting the polynomials defined by Equations 13-4 and 13-5.

$$P(x) - Q(x) = (a_0 - b_0) + (a_1 - b_1)x + (a_2 - b_2)x^2 + a_3x^3$$

Polynomial Multiplication

Multiplying one polynomial by another polynomial involves multiplying each term of one polynomial by each term of the other polynomial. The following equations show the result of multiplying the polynomials defined by Equations 13-4 and 13-5.

$$\begin{aligned} P(x)Q(x) &= (a_0 + a_1x + a_2x^2 + a_3x^3)(b_0 + b_1x + b_2x^2) \\ &= a_0(b_0 + b_1x + b_2x^2) + a_1x(b_0 + b_1x + b_2x^2) \\ &\quad + a_2x^2(b_0 + b_1x + b_2x^2) + a_3x^3(b_0 + b_1x + b_2x^2) \\ &= a_3b_2x^5 + (a_3b_1 + a_2b_2)x^4 + (a_3b_0 + a_2b_1 + a_1b_2)x^3 \\ &\quad + (a_2b_0 + a_1b_1 + a_0b_2)x^2 + (a_1b_0 + a_0b_1)x + a_0b_0 \end{aligned}$$

Polynomial Division

Dividing the two polynomials $P(x)$ and $Q(x)$ results in the quotient $U(x)$ and remainder $V(x)$, such that the following equation is true.

$$P(x) = Q(x)U(x) + V(x)$$

For example, the following equations define polynomials $P(x)$ and $Q(x)$.

$$P(x) = 5 - 3x - x^2 + 2x^3 \qquad (13\text{-}6)$$

$$Q(x) = 1 - 2x + x^2 \tag{13-7}$$

Complete the following steps to divide $P(x)$ by $Q(x)$.

1. Divide the highest order term in Equation 13-6 by the highest order term in Equation 13-7.

$$x^2 - 2x + 1 \overline{)\, 2x^3 - x^2 - 3x + 5} \qquad \begin{array}{r} 2x \end{array} \tag{13-8}$$

2. Multiply the result of Equation 13-8 by $Q(x)$ from Equation 13-7.

$$2xQ(x) = 2x - 4x^2 + 2x^3 \tag{13-9}$$

3. Subtract the product of Equation 13-9 from $P(x)$.

$$\begin{array}{r} 2x \\ x^2 - 2x + 1 \overline{)\, 2x^3 - x^2 - 3x + 5} \\ \underline{-(2x^3 - 4x^2 + 2x)} \\ 3x^2 - 5x + 5 \end{array}$$

The highest order term becomes $3x^2$.

4. Repeat step 1 through step 3 using $3x^2$ as the highest term of $P(x)$.
   a. Divide $3x^2$ by the highest order term in Equation 13-7.

$$\frac{3x^2}{x^2} = 3 \tag{13-10}$$

   b. Multiply the result of Equation 13-10 by $Q(x)$ from Equation 13-7.

$$3Q(x) = 3x^2 - 6x + 3 \tag{13-11}$$

   c. Subtract the result of Equation 13-11 from $3x^2 - 5x + 5$.

$$\begin{array}{r} 2x + 3 \\ x^2 - 2x + 1 \overline{)\, 2x^3 - x^2 - 3x + 5} \\ \underline{-(2x^3 - 4x^2 + 2x)} \\ 3x^2 - 5x + 5 \\ \underline{-(3x^2 - 6x + 3)} \\ x + 2 \end{array}$$

Because the order of the remainder $x + 2$ is lower than the order of $Q(x)$, the polynomial division procedure stops. The following equations give the quotient polynomial $U(x)$ and the remainder polynomial $V(x)$ for the division of Equation 13-6 by Equation 13-7.

$$U(x) = 3 + 2x$$

$$V(x) = 2 + \text{x}.$$

Polynomial Composition

Polynomial composition involves replacing the variable x in a polynomial with another polynomial. For example, replacing x in Equation 13-4 with the polynomial from Equation 13-5 results in the following equation.

$$\begin{aligned} P(Q(x)) &= a_0 + a_1 Q(x) + a_2 (Q(x))^2 + a_3 (Q(x))^3 \\ &= a_0 + Q(x)\{a_1 + Q(x)[a_2 + a_3 Q(x)]\} \end{aligned}$$

where $P(Q(x))$ denotes the composite polynomial.

Greatest Common Divisor of Polynomials

The greatest common divisor of two polynomials $P(x)$ and $Q(x)$ is a polynomial $R(x) = \gcd(P(x), Q(x))$ and has the following properties:

- $R(x)$ divides $P(x)$ and $Q(x)$.
- $C(x)$ divides $P(x)$ and $Q(x)$ where $C(x)$ divides $R(x)$.

The following equations define two polynomials $P(x)$ and $Q(x)$.

$$P(x) = U(x)R(x) \tag{13-12}$$

$$Q(x) = V(x)R(x) \tag{13-13}$$

where $U(x)$, $V(x)$, and $R(x)$ are polynomials.

The following conditions are true for Equations 13-12 and 13-13:

- $U(x)$ and $R(x)$ are factors of $P(x)$.
- $V(x)$ and $R(x)$ are factors of $Q(x)$.
- $P(x)$ is a multiple of $U(x)$ and $R(x)$.
- $Q(x)$ is a multiple of $V(x)$ and $R(x)$.
- $R(x)$ is a common factor of polynomials $P(x)$ and $Q(x)$.

If $P(x)$ and $Q(x)$ have the common factor $R(x)$, and if $R(x)$ is divisible by any other common factors of $P(x)$ and $Q(x)$ such that the division does not result in a remainder, $R(x)$ is the greatest common divisor of $P(x)$ and $Q(x)$. If the greatest common divisor $R(x)$ of polynomials $P(x)$ and $Q(x)$ is equal to a constant, $P(x)$ and $Q(x)$ are coprime.

You can find the greatest common divisor of two polynomials by using Euclid's division algorithm and an iterative procedure of polynomial division. If the order of $P(x)$ is larger than $Q(x)$, you can complete the following steps to find the greatest common divisor $R(x)$.

1. Divide $P(x)$ by $Q(x)$ to obtain the quotient polynomial $Q_1(x)$ and remainder polynomial $R_1(x)$.

$$P(x) = Q(x)Q_1(x) + R_1(x)$$

2. Divide $Q(x)$ by $R_1(x)$ to obtain the new quotient polynomial $Q_2(x)$ and new remainder polynomial $R_2(x)$.

$$Q(x) = R_1(x)Q_2(x) + R_2(x)$$

3. Divide $R_1(x)$ by $R_2(x)$ to obtain $Q_3(x)$ and $R_3(x)$

$$R_1(x) = R_2(x)Q_3(x) + R_3(x)$$

$$R_2(x) = R_3(x)Q_4(x) + R_4(x)$$

$$\vdots$$

If the remainder polynomial becomes zero, as shown by the following equation,

$$R_{n-1}(x) = R_n(x)Q_{n+1}(x),$$

the greatest common divisor $R(x)$ of polynomials $P(x)$ and $Q(x)$ equals $R_n(x)$.

Least Common Multiple of Two Polynomials

Finding the least common multiple of two polynomials involves finding the smallest polynomial that is a multiple of each polynomial.

$P(x)$ and $Q(x)$ are polynomials defined by Equations 13-12 and 13-13, respectively. If $L(x)$ is a multiple of both $P(x)$ and $Q(x)$, $L(x)$ is a common multiple of $P(x)$ and $Q(x)$. In addition, if $L(x)$ has the lowest order among

all the common multiples of $P(x)$ and $Q(x)$, $L(x)$ is the least common multiple of $P(x)$ and $Q(x)$.

If $L(x)$ is the least common multiple of $P(x)$ and $Q(x)$ and if $R(x)$ is the greatest common divisor of $P(x)$ and $Q(x)$, dividing the product of $P(x)$ and $Q(x)$ by $R(x)$ obtains $L(x)$, as shown by the following equation.

$$L(x) = \frac{P(x)Q(x)}{R(x)} = \frac{U(x)R(x)V(x)R(x)}{R(x)} = U(x)V(x)R(x)$$

Derivatives of a Polynomial

Finding the derivative of a polynomial involves finding the sum of the derivatives of the terms of the polynomial.

Equation 13-14 defines an n^{th}-order polynomial, $T(x)$.

$$T(x) = c_0 + c_1x + c_2x^2 + \ldots + c_nx^n \tag{13-14}$$

The first derivative of $T(x)$ is a polynomial of order $n - 1$, as shown by the following equation.

$$\frac{d}{dx}T(x) = c_1 + 2c_2x + 3c_3x^2 + \ldots + nc_nx^{n-1}$$

The second derivative of $T(x)$ is a polynomial of order $n - 2$, as shown by the following equation.

$$\frac{d^2}{dx^2}T(x) = 2c_2 + 6c_3x + \ldots + n(n-1)c_nx^{n-2}$$

The following equation defines the k^{th} derivative of $T(x)$.

$$\frac{d^k}{dx^k}T(x) = k!c_k + \frac{(k+1)!}{1!}c_{k+1}x + \frac{(k+2)!}{2!}c_{k+2}x^2 + \ldots + \frac{n!}{(n-k)!}c_nx^{n-k}$$

where $k \leq n$.

The Newton-Raphson method of finding the zeros of an arbitrary equation is an application where you need to determine the derivative of a polynomial.

Integrals of a Polynomial

Finding the integral of a polynomial involves the summation of integrals of the terms of the polynomial.

Indefinite Integral of a Polynomial

The following equation yields the indefinite integral of the polynomial $T(x)$ from Equation 13-14.

$$\int T(x)dx = c + c_0 x + \frac{1}{2}c_1 x^2 + \ldots + \frac{1}{n+1}c_n x^{n+1}$$

Because the derivative of a constant is zero, c can be an arbitrary constant. For convenience, you can set c to zero.

Definite Integral of a Polynomial

Subtracting the evaluations at the two limits of the indefinite integral obtains the definite integral of the polynomial, as shown by the following equation.

$$\begin{aligned}\int_a^b T(x)dx &= \left(c_0 x + \frac{1}{2}c_1 x^2 + \ldots + \frac{1}{n+1}c_n x^{n+1}\right)\Bigg|_{x=b} \\ &\quad -\left(c_0 x + \frac{1}{2}c_1 x^2 + \ldots + \frac{1}{n+1}c_n x^{n+1}\right)\Bigg|_{x=a} \\ &= \left(c_0 x + \frac{1}{2}c_1 x^2 + \ldots + \frac{1}{n+1}c_n x^{n+1}\right)\Bigg|_a^b\end{aligned}$$

Number of Real Roots of a Real Polynomial

For a real polynomial, you can find the number of real roots of the polynomial over a certain interval by applying the Sturm function.

If

$$P_0(x) = P(x)$$

and

$$P_1(x) = \frac{d}{dx}P(x),$$

the following equation defines the Sturm function.

$$P_i(x) = -\left\{P_{i-2}(x) - P_{i-1}(x)\left[\frac{P_{i-2}(x)}{P_{i-1}(x)}\right]\right\}, \qquad i = 2, 3, \ldots$$

where $P_i(x)$ is the Sturm function and

$$\left[\frac{P_{i-2}(x)}{P_{i-1}(x)}\right]$$

represents the quotient polynomial resulting from the division of $P_{i-2}(x)$ by $P_{i-1}(x)$.

You can calculate $P_i(x)$ until it becomes a constant. For example, the following equations show the calculation of the Sturm function over the interval (–2,1).

$$P_0(x) = P(x) = 1 - 4x + 2x^3$$

$$P_1(x) = \frac{d}{dx}P(x) = -4 + 6x^2$$

$$P_2(x) = -\left\{P_0(x) - P_1(x)\left[\frac{P_0(x)}{P_1(x)}\right]\right\}$$

$$= -\left\{P_0(x) - P_1(x)\frac{1}{3}x\right\}$$

$$= -1 + \frac{8}{3}x$$

$$P_3(x) = -\left\{P_1(x) - P_2(x)\left[\frac{P_1(x)}{P_2(x)}\right]\right\}$$

$$= -\left\{P_1(x) - P_2(x)\left(\frac{27}{32} + \frac{9}{4}x\right)\right\}$$

$$= \frac{101}{32}$$

To evaluate the Sturm functions at the boundary of the interval (–2,1), you do not have to calculate the exact values in the evaluation. You only need to know the signs of the values of the Sturm functions. Table 13-1 lists the signs of the Sturm functions for the interval (–2,1).

Table 13-1. Signs of the Sturm Functions for the Interval (–2, 1)

x	$P_0(x)$	$P_1(x)$	$P_2(x)$	$P_3(x)$	Number of Sign Changes
–2	–	+	–	+	3
1	–	+	+	+	1

In Table 13-1, notice the number of sign changes for each boundary. For $x = -2$, the evaluation of $P_i(x)$ results in three sign changes. For $x = 1$, the evaluation of $P_i(x)$ results in one sign change.

The difference in the number of sign changes between the two boundaries corresponds to the number of real roots that lie in the interval. For the calculation of the Sturm function over the interval (–2,1), the difference in the number of sign changes is two, which means two real roots of polynomial $P(x)$ lie in the interval (–2,1). Figure 13-1 shows the result of evaluating $P(x)$ over (–2,1).

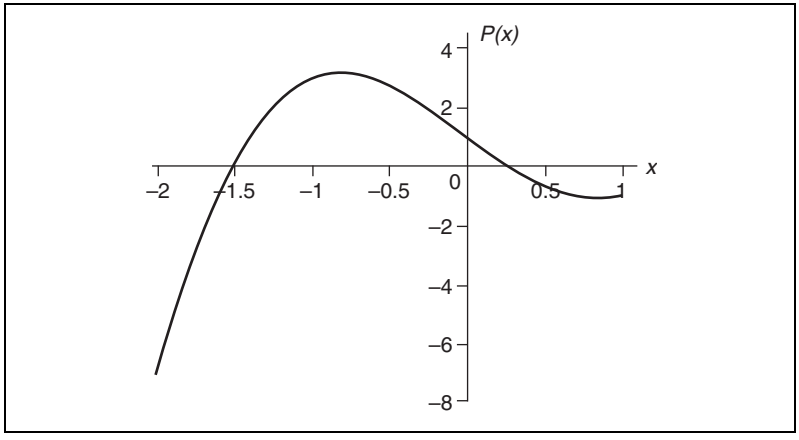


Figure 13-1. Result of Evaluating $P(x)$ over (–2, 1)

In Figure 13-1, the two real roots lie at approximately –1.5 and 0.26.

Rational Polynomial Function Operations

Rational polynomial functions have many applications, such as filter design, system theory, and digital image processing. In particular, rational polynomial functions provide the most common way of representing the *z*-transform. A rational polynomial function takes the form of the division of two polynomials, as shown by the following equation.

$$F(x) = \frac{B(x)}{A(x)} = \frac{b_0 + b_1 x + b_2 x^2 + \ldots + b_m x^m}{a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n}$$

where $F(x)$ is the rational polynomial, $B(x)$ is the numerator polynomial, $A(x)$ is the denominator polynomial, and $A(x)$ cannot equal zero.

The roots of $B(x)$ are the zeros of $F(x)$. The roots of $A(x)$ are the poles of $F(x)$.

The following equations define two rational polynomials used in the following sections.

$$F_1(x) = \frac{B_1(x)}{A_1(x)} \tag{13-15}$$

$$F_2(x) = \frac{B_2(x)}{A_2(x)}$$

Rational Polynomial Function Addition

The following equation shows the addition of two rational polynomials.

$$F_1(x) + F_2(x) = \frac{B_1(x)A_2(x) + B_2(x)A_1(x)}{A_1(x)A_2(x)}$$

Rational Polynomial Function Subtraction

The following equation shows the subtraction of two rational polynomials.

$$F_1(x) - F_2(x) = \frac{B_1(x)A_2(x) - B_2(x)A_1(x)}{A_1(x)A_2(x)}$$

Rational Polynomial Function Multiplication

The following equation shows the multiplication of two rational polynomials.

$$F_1(x)F_2(x) = \frac{B_1(x)B_2(x)}{A_1(x)A_2(x)}$$

Rational Polynomial Function Division

The following equation shows the division of two rational polynomials.

$$\frac{F_1(x)}{F_2(x)} = \frac{B_1(x)A_2(x)}{A_1(x)B_2(x)}$$

Negative Feedback with a Rational Polynomial Function

Figure 13-2 shows a diagram of a generic system with negative feedback.

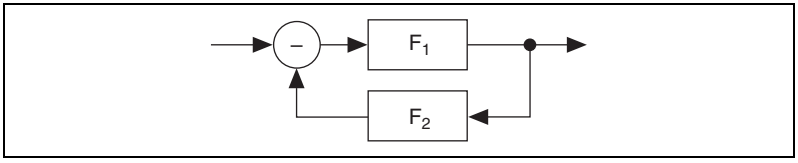


Figure 13-2. Generic System with Negative Feedback

For the system shown in Figure 13-2, the following equation yields the transfer function of the system.

$$H(x) = \frac{F_1(x)}{1 + F_1(x)F_2(x)} = \frac{B_1(x)A_2(x)}{A_1(x)A_2(x) + B_1(x)B_2(x)}$$

Positive Feedback with a Rational Polynomial Function

Figure 13-3 shows a diagram of a generic system with positive feedback.

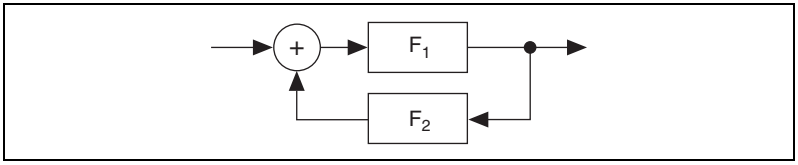


Figure 13-3. Generic System with Positive Feedback

For the system shown in Figure 13-3, the following equation yields the transfer function of the system.

$$H(x) = \frac{F_1(x)}{1 - F_1(x)F_2(x)} = \frac{B_1(x)A_2(x)}{A_1xA_2x - B_1(x)B_2(x)}$$

Derivative of a Rational Polynomial Function

The derivative of a rational polynomial function also is a rational polynomial function. Using the quotient rule, you obtain the derivative of a rational polynomial function from the derivatives of the numerator and denominator polynomials. According to the quotient rule, the following equation yields the first derivative of the rational polynomial function $F_1(x)$ defined in Equation 13-15.

$$\frac{d}{dx}F_1(x) = \frac{A_1(x)\frac{d}{dx}B_1(x) - B_1(x)\frac{d}{dx}A_1(x)}{(A_1(x))^2}$$

You can derive the second derivative of a rational polynomial function from the first derivative, as shown by the following equation.

$$\frac{d^2}{dx^2}F_1(x) = \frac{d}{dx}\left(\frac{d}{dx}F_1(x)\right)$$

You continue to derive rational polynomial function derivatives such that you derive the jth derivative of a rational polynomial function from the $(j - 1)$th derivative.

Partial Fraction Expansion

Partial fraction expansion involves splitting a rational polynomial into a summation of low order rational polynomials. Partial fraction expansion is a useful tool for z-transform and digital filter structure conversion.

Heaviside Cover-Up Method

The Heaviside cover-up method is the easiest of the partial fraction expansion methods.

The following actions and conditions illustrate the Heaviside cover-up method:

- Define a rational polynomial function $F(x)$ with the following equation.

$$F(x) = \frac{B(x)}{A(x)} = \frac{b_0 + b_1 x + b_2 x^2 + \ldots + b_m x^m}{a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n}$$

  where $m < n$, meaning, without loss of generality, the order of $B(x)$ is lower than the order of $A(x)$.

- Assume that $A(x)$ has one repeated root r_0 of multiplicity k and use the following equation to express $A(x)$ in terms of its roots.

$$A(x) = a_n(x - r_0)^k(x - r_1)(x - r_2) \ldots (x - r_{n-k})$$

- Rewrite $F(x)$ as a sum of partial fractions.

$$\begin{aligned} F(x) &= \frac{B(x)}{a_n(x - r_0)^k(x - r_1)\ldots(x - r_{n-k})} \\ &= \frac{\beta_0}{x - r_0} + \frac{\beta_1}{(x - r_0)^2} + \ldots + \frac{\beta_{k-1}}{(x - r_0)^k} \\ &\quad + \frac{\alpha_1}{x - r_1} + \frac{\alpha_2}{x - r_2} + \ldots + \frac{\alpha_{n-k}}{x - r_{n-k}} \end{aligned}$$

  where

$$\alpha_i = (x - r_i)F(x)\Big|_{x = r_i} \qquad i = 1, 2, \ldots, n - k$$

$$\beta_j = \frac{1}{(k - j - 1)!}\frac{d^{(k-j-1)}}{dx^{(k-j-1)}}((x - r_0)^k F(x))\Big|_{x = r_0} \qquad j = 0, 1, \ldots, k - 1$$

Orthogonal Polynomials

A set of polynomials $P_i(x)$ are orthogonal polynomials over the interval $a < x < b$ if each polynomial in the set satisfies the following equations.

$$\begin{cases} \int_a^b w(x)P_n(x)P_m(x)dx = 0, & n \neq m \\ \int_a^b w(x)P_n(x)P_n(x)dx \neq 0, & n = m \end{cases}$$

The interval (a, b) and the weighting function $w(x)$ vary depending on the set of orthogonal polynomials. One of the most important applications of orthogonal polynomials is to solve differential equations.

Chebyshev Orthogonal Polynomials of the First Kind

The recurrence relationship defines Chebyshev orthogonal polynomials of the first kind $T_n(x)$, as shown by the following equations.

$$T_0(x) = 1$$

$$T_1(x) = x$$

$$T_n(x) = 2xT_{n-1}(x) - T_{n-2}(x) \qquad n = 2, 3, \ldots$$

Chebyshev orthogonal polynomials of the first kind satisfy the following equations.

$$\begin{cases} \int_{-1}^{1} \frac{1}{\sqrt{1-x^2}} T_n(x)T_m(x)dx = 0, & n \neq m \\ \int_{-1}^{1} \frac{1}{\sqrt{1-x^2}} T_n(x)T_n(x)dx = \begin{cases} \frac{\pi}{2}, & n \neq 0 \\ \pi, & n = 0 \end{cases} \end{cases}$$

Chebyshev Orthogonal Polynomials of the Second Kind

The recurrence relationship defines Chebyshev orthogonal polynomials of the second kind $U_n(x)$, as shown by the following equations.

$$U_0(x) = 1$$

$$U_1(x) = 2x$$

$$U_n(x) = 2xU_{n-1}(x) - U_{n-2}(x) \qquad n = 2, 3, \ldots$$

Chebyshev orthogonal polynomials of the second kind satisfy the following equations.

$$\begin{cases} \int_{-1}^{1} \sqrt{1-x^2}\, U_n(x) U_m(x) dx = 0 & n \neq m \\ \int_{-1}^{1} \sqrt{1-x^2}\, U_n(x) U_n(x) dx = \frac{\pi}{2} & n = m \end{cases}$$

Gegenbauer Orthogonal Polynomials

The recurrence relationship defines Gegenbauer orthogonal polynomials $C_n^a(x)$, as shown by the following equations.

$$C_0^a(x) = 1$$

$$C_1^a(x) = 2ax$$

$$C_n^a(x) = \frac{2(n+a-1)}{n} x C_{n-1}^a(x) - \frac{n+2a-2}{n} C_{n-2}^a(x) \qquad \begin{matrix} n = 2, 3, \ldots \\ a \neq 0 \end{matrix}$$

Gegenbauer orthogonal polynomials satisfy the following equations.

$$\begin{cases} \int_{-1}^{1} (1-x^2)^{a-1/2} C_n^a(x) C_m^a(x) dx = 0 & n \neq m \\ \int_{-1}^{1} (1-x^2)^{a-1/2} C_n^a(x) C_n^a(x) dx = \begin{cases} \dfrac{\pi 2^{1-2a} \Gamma(n+2a)}{n!(n+a)\Gamma^2(a)} & a \neq 0 \\ \dfrac{2\pi}{n^2} & a = 0 \end{cases} \end{cases}$$

where $\Gamma(z)$ is a gamma function defined by the following equation.

$$\Gamma(z) = \int_0^\infty t^{z-1} e^{-t} dt$$

Hermite Orthogonal Polynomials

The recurrence relationship defines Hermite orthogonal polynomials $H_n(x)$, as shown by the following equations.

$$H_0(x) = 1$$

$$H_1(x) = 2x$$

$$H_n(x) = 2xH_{n-1}(x) - 2(n-1)H_{n-2}(x) \qquad n = 2, 3, \ldots$$

Hermite orthogonal polynomials satisfy the following equations.

$$\begin{cases} \int_{-\infty}^{\infty} e^{-x^2} H_n(x) H_m(x) dx = 0 & n \neq m \\ \int_{-\infty}^{\infty} e^{-x^2} H_n(x) H_n(x) dx = \sqrt{\pi} 2^n n! & n = m \end{cases}$$

Laguerre Orthogonal Polynomials

The recurrence relationship defines Laguerre orthogonal polynomials $L_n(x)$, as shown by the following equations.

$$L_0(x) = 1$$

$$L_1(x) = -x + 1$$

$$L_n(x) = \frac{2n-1-x}{n} L_{n-1}(x) - \frac{n-1}{n} L_{n-2}(x) \qquad n = 2, 3, \ldots$$

Laguerre orthogonal polynomials satisfy the following equations.

$$\begin{cases} \int_0^\infty e^{-x} L_n(x) L_m(x) dx = 0 & n \neq m \\ \int_0^\infty e^{-x} L_n(x) L_n(x) dx = 1 & n = m \end{cases}$$

Associated Laguerre Orthogonal Polynomials

The recurrence relationship defines associated Laguerre orthogonal polynomials $L_n^a(x)$, as shown by the following equations.

$$L_0^a(x) = 1$$

$$L_1^a(x) = -x + a + 1$$

$$L_n^a(x) = \frac{2n + a - 1 - x}{n} L_{n-1}^a(x) - \frac{n + a - 1}{n} L_{n-2}^a(x) \qquad n = 2, 3, \ldots$$

Associated Laguerre orthogonal polynomials satisfy the following equation.

$$\begin{cases} \int_0^\infty e^{-x} x^a L_n^a(x) L_m^a(x)\, dx = 0 & n \neq m \\ \int_0^\infty e^{-x} x^a L_n^a(x) L_n^a(x)\, dx = \dfrac{\Gamma(a + n + 1)}{n!} & n = m \end{cases}$$

Legendre Orthogonal Polynomials

The recurrence relationship defines Legendre orthogonal polynomials $P_n(x)$, as shown by the following equations.

$$P_0(x) = 1$$

$$P_1(x) = x$$

$$P_n(x) = \frac{2n - 1}{n} x P_{n-1}(x) - \frac{n - 1}{n} P_{n-2}(x) \qquad n = 2, 3, \ldots$$

Legendre orthogonal polynomials satisfy the following equations.

$$\begin{cases} \int_{-1}^{1} P_n(x) P_m(x)\, dx = 0 & n \neq m \\ \int_{-1}^{1} P_n(x) P_n(x)\, dx = \dfrac{2}{2n + 1} & n = m \end{cases}$$

Evaluating a Polynomial with a Matrix

The matrix evaluation of a polynomial differs from 2D polynomial evaluation.

When performing matrix evaluation of a polynomial, you must use a square matrix. The following equations define a second-order polynomial $P(x)$ and a square 2×2 matrix G.

$$P(x) = a_0 + a_1 x + a_2 x^2 \tag{13-16}$$

$$G = \begin{bmatrix} g_1 & g_2 \\ g_3 & g_4 \end{bmatrix} \tag{13-17}$$

In 2D polynomial evaluation, you evaluate $P(x)$ at each element of matrix G, as shown by the following equation.

$$P(G) = \begin{bmatrix} P(x)\big|_{x = g_1} & P(x)\big|_{x = g_2} \\ P(x)\big|_{x = g_3} & P(x)\big|_{x = g_4} \end{bmatrix}$$

When performing matrix polynomial evaluation, you replace the variable x with matrix G, as shown by the following equation.

$$P([G]) = a_o I + a_1 G + a_1 GG$$

where I is the identity matrix of the same size as G.

In the following equations, actual values replace the variables a and g in Equations 13-16 and 13-17.

$$P(x) = 5 + 3x + 2x^2 \tag{13-18}$$

$$G = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \tag{13-19}$$

The following equation shows the matrix evaluation of the polynomial $P(x)$ from Equation 13-18 with matrix G from Equation 13-19.

$$P([G]) = 5\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix} + 3\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix} + 2\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}$$
$$= \begin{bmatrix}5 & 0\\ 0 & 5\end{bmatrix} + \begin{bmatrix}3 & 6\\ 9 & 12\end{bmatrix} + \begin{bmatrix}14 & 20\\ 30 & 44\end{bmatrix}$$
$$= \begin{bmatrix}22 & 26\\ 39 & 61\end{bmatrix}$$

Polynomial Eigenvalues and Vectors

For every operator, a collection of functions exists that when operated on by the operator produces the same function, modified only by a multiplicative constant factor. The members of the collection of functions are eigenfunctions. The multiplicative constants modifying the eigenfunctions are eigenvalues. The following equation illustrates the eigenfunction/eigenvalue relationship.

$$\hat{A} f(x) = a f(x),$$

where $f(x)$ is an eigenfunction of A and a is the eigenvalue of $f(x)$.

Some applications lead to a polynomial eigenvalue problem. Given a set of square matrices, the problem becomes determining a scalar λ and a nonzero vector x such that Equation 13-20 is true.

$$\Psi(\lambda)x = (C_0 + \lambda C_1 + \ldots + \lambda^{n-1} C_{n-1} + \lambda^n C_n)x = \bar{0} \qquad (13\text{-}20)$$

The following conditions apply to Equation 13-20:

- $\Psi(\lambda)$ is the matrix polynomial whose coefficients are square matrixes.
- C_i is a square matrix of size $m \times m$, $i = 0, 1, \ldots, n$.
- λ is the eigenvalue of $\Psi(\lambda)$.
- x is the corresponding eigenvector of $\Psi(\lambda)$ and has length m.
- $\bar{0}$ is the zero vector and has length m.

You can write the polynomial eigenvalue problem as a generalized eigenvalue problem, as shown by the following equation.

$$Az = \lambda Bz$$

where

$$A = \begin{bmatrix} \bar{\bar{0}} & I & \bar{\bar{0}} & \cdots & \bar{\bar{0}} \\ \bar{\bar{0}} & \bar{\bar{0}} & I & \cdots & \bar{\bar{0}} \\ \vdots & \vdots & & \ddots & \vdots \\ \vdots & \vdots & \vdots & & I \\ -C_0 & -C_1 & -C_2 & \cdots & -C_{n-1} \end{bmatrix}$$ and is an $nm \times nm$ matrix;

$$B = \begin{bmatrix} I & & & \\ & I & & \\ & & \ddots & \\ & & & I \\ & & & & C_n \end{bmatrix}$$ and is an $nm \times nm$ matrix;

$$z = \begin{bmatrix} x \\ \lambda x \\ \lambda^2 x \\ \vdots \\ \lambda^{n-1} x \end{bmatrix}$$ and is an nm matrix;

$\bar{\bar{0}}$ is the zero matrix of size $m \times m$;

I is the identity matrix of size $m \times m$.

Entering Polynomials in LabVIEW

Use the Polynomial and Rational Polynomial VIs to perform polynomial operations.

LabVIEW uses 1D arrays for polynomial inputs and outputs. The 1D array stores the polynomial coefficients. When entering polynomial coefficient values into an array, maintain a consistent method for entering the values. The order in which LabVIEW displays the results of polynomial operations reflects the order in which you enter the input polynomial coefficient values. National Instruments recommends entering polynomial coefficient values in ascending order of power. For example, the following equations define polynomials $P(x)$ and $Q(x)$.

$$P(x) = 1 - 3x + 4x^2 + 2x^3$$

$$Q(x) = 1 - 2x + x^2$$

You can describe $P(x)$ and $Q(x)$ by vectors P and Q, as shown in the following equations.

$$P = \begin{bmatrix} 1 \\ -3 \\ 4 \\ 2 \end{bmatrix}$$

$$Q = \begin{bmatrix} 1 \\ -2 \\ 1 \end{bmatrix}$$

Figure 13-4 shows the front panel of a VI that uses the Add Polynomials VI to add $P(x)$ and $Q(x)$.

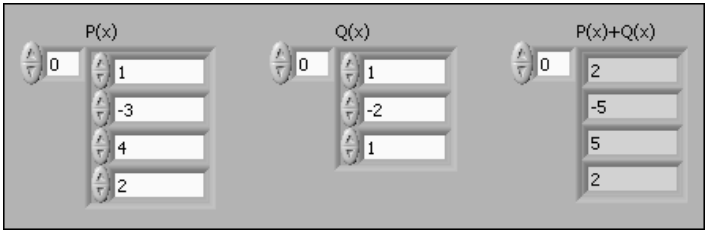


Figure 13-4. Adding $P(x)$ and $Q(x)$

In Figure 13-4, you enter the polynomial coefficients into the array controls, **P(x)** and **Q(x)**, in ascending order of power. Also, the VI displays the results of the addition in **P(x) + Q(x)** in ascending order of power, based on the order of the two input arrays.

Part III

Point-By-Point Analysis

This part describes the concepts of point-by-point analysis, answers frequently asked questions about point-by-point analysis, and describes a case study that illustrates the use of the Point By Point VIs.

14

Point-By-Point Analysis

This chapter describes the concepts of point-by-point analysis, answers frequently asked questions about point-by-point analysis, and describes a case study that illustrates the use of the Point By Point VIs. Use the NI Example Finder to find examples of using the Point By Point VIs.

Introduction to Point-By-Point Analysis

Point-by-point analysis is a method of continuous data analysis in which analysis occurs for each data point, point by point. Point-by-point analysis is ideally suited to real-time data acquisition. When your data acquisition system requires real-time, deterministic performance, you can build a program that uses point-by-point versions of array-based LabVIEW analysis VIs.

Real-time performance is a reality for data acquisition. With point-by-point analysis, data analysis also can utilize real-time performance. The discrete stages of array-based analysis, such as buffer preparation, analysis, and output, can make array-based analysis too slow for higher speed, deterministic, real-time systems.

Point-by-point analysis enables you to accomplish the following tasks:

- Track and respond to real-time events.
- Connect the analysis process directly to the signal for speed and minimal data loss.
- Perform programming tasks more easily, because you do not allocate arrays and you make fewer adjustments to sampling rates.
- Synchronize analysis with data acquisition automatically, because you work with a single signal instantaneously.

Using the Point By Point VIs

The Point By Point VIs correspond to each array-based analysis VI that is relevant to continuous data acquisition. However, you must account for programming differences. You usually have fewer programming tasks when you use the Point By Point VIs. Table 14-1 describes characteristic inputs and outputs of the Point By Point VIs.

Table 14-1. Characteristic Inputs and Outputs for Point By Point VIs

Parameter	Description
input data	Incoming data
output data	Outgoing, analyzed data
initialize	Routine that resets the internal state of a VI
sample length	Setting for your data acquisition system or computation system that best represents the area of interest in the data

Refer to the *Case Study of Point-By-Point Analysis* section of this chapter for an example of a point-by-point analysis system.

Initializing Point By Point VIs

This section describes when and how to use the point-by-point **initialize** parameter of many Point By Point VIs. This section also describes the First Call? function.

Purpose of Initialization in Point By Point VIs

Using the **initialize** parameter, you can reset the internal state of Point By Point VIs without interrupting the continuous flow of data or computation. You can reset a VI in response to events such as the following:

- A user changing the value of a parameter
- The application generating a specific event or reaching a threshold

For example, the Value Has Changed PtByPt VI can respond to change events such as the following:

- Receiving the input data
- Detecting the change

- Generating a Boolean TRUE value that triggers initialization in another VI
- Transferring the input data to another VI for processing

Figure 14-1 shows the Value Has Changed PtByPt VI triggering initialization in another VI and transferring data to that VI. In this case, the input data is a parameter value for the target VI.

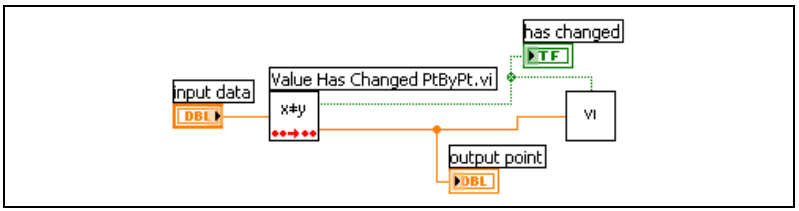


Figure 14-1. Typical Role of the Value Has Changed PtByPt VI

Many point-by-point applications do not require use of the **initialize** parameter because initialization occurs automatically whenever an operator quits an application and then starts again.

Using the First Call? Function

Where necessary, use the First Call? function to build point by point VIs. In a VI that includes the First Call? function, the internal state of the VI is reset once, the first time you call the VI. The value of the **initialize** parameter in the First Call? function is always TRUE for the first call to the VI. The value remains FALSE for the remainder of the time you run the VI. Figure 14-2 shows a typical use of the First Call? function with a While Loop.

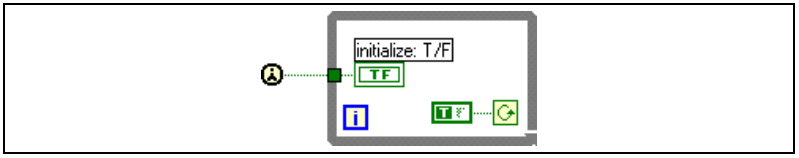


Figure 14-2. Using the First Call? Function with a While Loop

Error Checking and Initialization

The Point By Point VIs generate errors to help you identify flaws in the configuration of the applications you build. Several point-by-point error codes exist in addition to the standard LabVIEW error codes.

Error codes usually identify invalid parameters and settings. For higher-level error checking, configure your program to monitor and respond to irregularities in data acquisition or in computation. For example, you create a form of error checking when you range check your data.

A Point By Point VI generates an error code once at the initial call to the VI or at the first call to the VI after you initialize your application. Because Point By Point VIs generate error codes only once, they can perform optimally in a real-time, deterministic application.

The Point By Point VIs generate an error code to inform you of any invalid parameters or settings when they detect an error during the first call. In subsequent calls, the Point By Point VIs set the error code to zero and continue running, generating no error codes. You can program your application to take one of the following actions in response to the first error:

- Report the error and continue running.
- Report the error and stop.
- Ignore the error and continue running. This is the default behavior.

The following programming sequence describes how to use the Value Has Changed PtByPt VI to build a point-by-point error checking mechanism for Point By Point VIs that have an **error** parameter.

1. Choose a parameter that you want to monitor closely for errors.
2. Wire the parameter value as **input data** to the Value Has Changed PtByPt VI.
3. Transfer the **output data**, which is always the unchanged **input data** in Value Has Changed PtByPt VI, to the target VI.
4. Pass the TRUE event generated by the Value Has Changed PtByPt VI to the target VI to trigger initialization, as shown in Figure 14-1. The Value Has Changed PtByPt VI outputs a TRUE value whenever the input parameter value changes.

For the first call that follows initialization of the target VI, LabVIEW checks for errors. Initialization of the target VI and error checking occurs every time the input parameter changes.

Frequently Asked Questions

This section answers frequently asked questions about point-by-point analysis.

What Are the Differences between Point-By-Point Analysis and Array-Based Analysis in LabVIEW?

Tables 14-2 and 14-3 compare array-based LabVIEW analysis to point-by-point analysis from multiple perspectives. In Table 14-2, the differences between two automotive fuel delivery systems, carburation and fuel injection, demonstrate the differences between array-based data analysis and point-by-point analysis.

Table 14-2. Comparison of Traditional and Newer Paradigms

Traditional Paradigm	Newer Paradigm
Automotive Technology	
Carburation • Fuel accumulates in a float bowl. • Engine vacuum draws fuel through a single set of metering valves that serve all combustion chambers. • Somewhat efficient combustion occurs.	**Fuel Injection** • Fuel flows continuously from gas tank. • Fuel sprays directly into each combustion chamber at the moment of combustion. • Responsive, precise combustion occurs.
Data Analysis Technology	
Array-Based Analysis • Prepare a buffer unit of data. • Analyze data. • Produce a buffer of analyzed data. • Generate report.	**Point-By-Point Analysis** • Receive continuous stream of data. • Filter and analyze data continuously. • Generate real-time events and reports continuously.

Table 14-3 presents other comparisons between array-based and point-by-point analysis.

Table 14-3. Comparison of Array-Based and Point-By-Point Data Analysis

Characteristic	Array-Based Analysis	Data Acquisition and Analysis with Point By Point VIs
Compatibility	Limited compatibility with real-time systems	Compatible with real-time systems; backward compatible with array-based systems
Data typing	Array-oriented	Scalar-oriented
Interruptions	Interruptions critical	Interruptions tolerated
Operation	You observe, offline	You control, online
Performance and programming	Compensate for startup data loss (4–5 seconds) with complex “state machines”	Startup data loss does not occur; initialize the data acquisition system once and run continuously
Point of view	Reflection of a process, like a mirror	Direct, natural flow of a process
Programming	Specify a buffer	No explicit buffers
Results	Output a report	Output a report and an event in real time
Run-time behavior	Delayed processing	Real time
Run-time behavior	Stop	Continue
Run-time behavior	Wait	Now
Work style	Asynchronous	Synchronous

Why Use Point-By-Point Analysis?

Point-by-point analysis works well with computer-based real-time data acquisition. In array-based analysis, the input-analysis-output process takes place for subsets of a larger data set. In point-by-point analysis, the input-analysis-output process takes place continuously, in real time.

What Is New about Point-By-Point Analysis?

When you perform point-by-point analysis, keep in mind the following concepts:

- **Initialization**—You must initialize the point-by-point analysis application to prevent interference from settings you made in previous sessions of data analysis.
- **Re-Entrant Execution**—You must enable LabVIEW re-entrant execution for point-by-point analysis. Re-entrant execution allocates fixed memory to a single analysis process, guaranteeing that two processes that use the same analysis function never interfere with each other.

Note If you create custom VIs to use in your own point-by-point application, be sure to enable re-entrant execution. Re-entrant execution is enabled by default in almost all Point By Point VIs.

- **Deterministic Performance**—Point-by-point analysis is the natural companion to many deterministic systems, because it efficiently integrates with the flow of a real-time data signal.

What Is Familiar about Point-By-Point Analysis?

The approach used for most point-by-point analysis operations in LabVIEW remains the same as array-based analysis. You use filters, integration, mean value algorithms, and so on, in the same situations and for the same reasons that you use these operations in array-based data analysis. In contrast, the computation of zeroes in polynomial functions is not relevant to point-by-point analysis, and point-by-point versions of these array-based VIs are not necessary.

How Is It Possible to Perform Analysis without Buffers of Data?

Analysis functions yield solutions that characterize the behavior of a data set. In array-based data acquisition and analysis, you might analyze a large set of data by dividing the data into 10 smaller buffers. Analyzing those 10 sets of data yields 10 solutions. You can further resolve those 10 solutions into one solution that characterizes the behavior of the entire data set.

In point-by-point analysis, you analyze an entire data set in real-time. A sample unit of a specific length replaces a buffer. The point-by-point sample unit can have a length that matches the length of a significant event in the data set that you are analyzing. For example, the application in

the *Case Study of Point-By-Point Analysis* section of this chapter acquires a few thousand samples per second to detect defective train wheels. The input data for the train wheel application comes from the signal generated by a train that is moving at 60 km to 70 km per hour. The sample length corresponds to the minimum distance between wheels.

A typical point-by-point analysis application analyzes a long series of sample units, but you are likely to have interest in only a few of those sample units. To identify those crucial samples of interest, the point-by-point application focuses on transitions, such as the end of the relevant signal.

The train wheel detection application in the *Case Study of Point-By-Point Analysis* section of this chapter uses the end of a signal to identify crucial samples of interest. The instant the application identifies the transition point, it captures the maximum amplitude reading of the current sample unit. This particular amplitude reading corresponds to the complete signal for the wheel on the train whose signal has just ended. You can use this real-time amplitude reading to generate an event or a report about that wheel and that train.

Why Is Point-By-Point Analysis Effective in Real-Time Applications?

In general, when you must process continuous, rapid data flow, point-by-point analysis can respond. For example, in industrial automation settings, control data flows continuously, and computers use a variety of analysis and transfer functions to control a real-world process. Point-by-point analysis can take place in real time for these engineering tasks.

Some real-time applications do not require high-speed data acquisition and analysis. Instead, they require simple, dependable programs. Point-by-point analysis offers simplicity and dependability, because you do not allocate arrays explicitly, and data analysis flows naturally and continuously.

Do I Need Point-By-Point Analysis?

As you increase the samples-per-seconds rate by factors of ten, the need for point-by-point analysis increases. The point-by-point approach simplifies the design, implementation, and testing process, because the flow of a point-by-point application closely matches the natural flow of the real-world processes you want to monitor and control.

You can continue to work without point-by-point analysis as long as you can control your processes without high-speed, deterministic, point-by-point data acquisition. However, if you dedicate resources in a real-time data acquisition application, use point-by-point analysis to achieve the full potential of your application.

What Is the Long-Term Importance of Point-By-Point Analysis?

Real-time data acquisition and analysis continue to demand more streamlined and stable applications. Point-by-point analysis is streamlined and stable because it directly ties into the acquisition and analysis process. Streamlined and stable point-by-point analysis allows the acquisition and analysis process to move closer to the point of control in field programmable gate array (FPGA) chips, DSP chips, embedded controllers, dedicated CPUs, and ASICs.

Case Study of Point-By-Point Analysis

The case study in this section uses the Train Wheel PtByPt VI and shows a complete point-by-point analysis application built in LabVIEW with Point By Point VIs. The Train Wheel PtByPt VI is a real-time data acquisition application that detects defective train wheels and demonstrates the simplicity and flexibility of point-by-point data analysis. The Train Wheel PtByPt VI is located in the `labview\examples\ptbypt\PtByPt_No_HW.llb`.

Point-By-Point Analysis of Train Wheels

In this example, the maintenance staff of a train yard must detect defective wheels on a train. The current method of detection consists of a railroad worker striking a wheel with a hammer and listening for a different resonance that identifies a flaw. Automated surveillance must replace manual testing, because manual surveillance is too slow, too prone to error, and too crude to detect subtle defects. An automated solution also adds the power of dynamic testing, because the train wheels can be in service during the test, instead of standing still.

The automated solution to detect potentially defective train wheels needs to have the following characteristics:

- Detect even subtle signs of defects quickly and accurately.
- Gather data when a train travels during a normal trip.
- Collect and analyze data in real time to simplify programming and to increase speed and accuracy of results.

The Train Wheel PtByPt VI offers a solution for detecting defective train wheels. Figures 14-3 and 14-4 show the front panel and the block diagram, respectively, for the Train Wheel PtByPt VI.

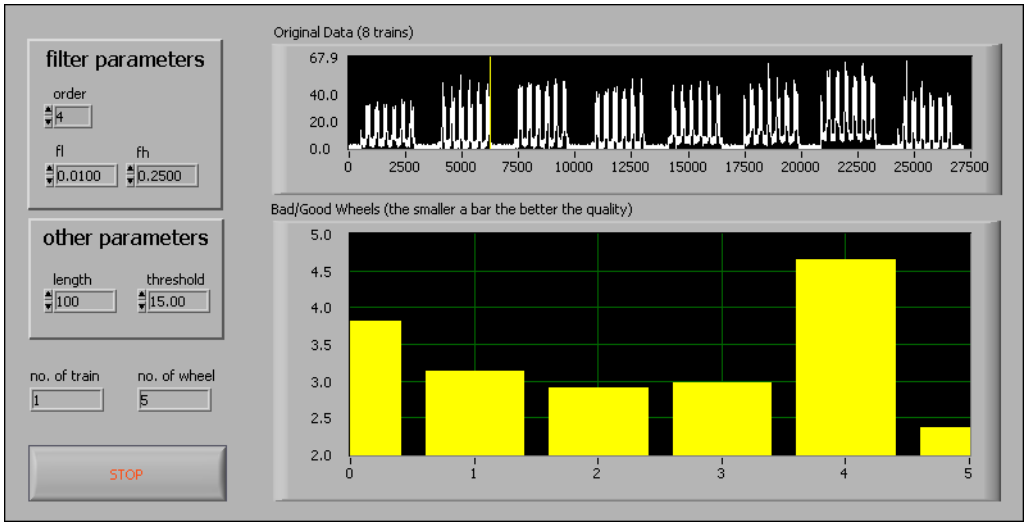


Figure 14-3. Front Panel of the Train Wheel PtByPt VI

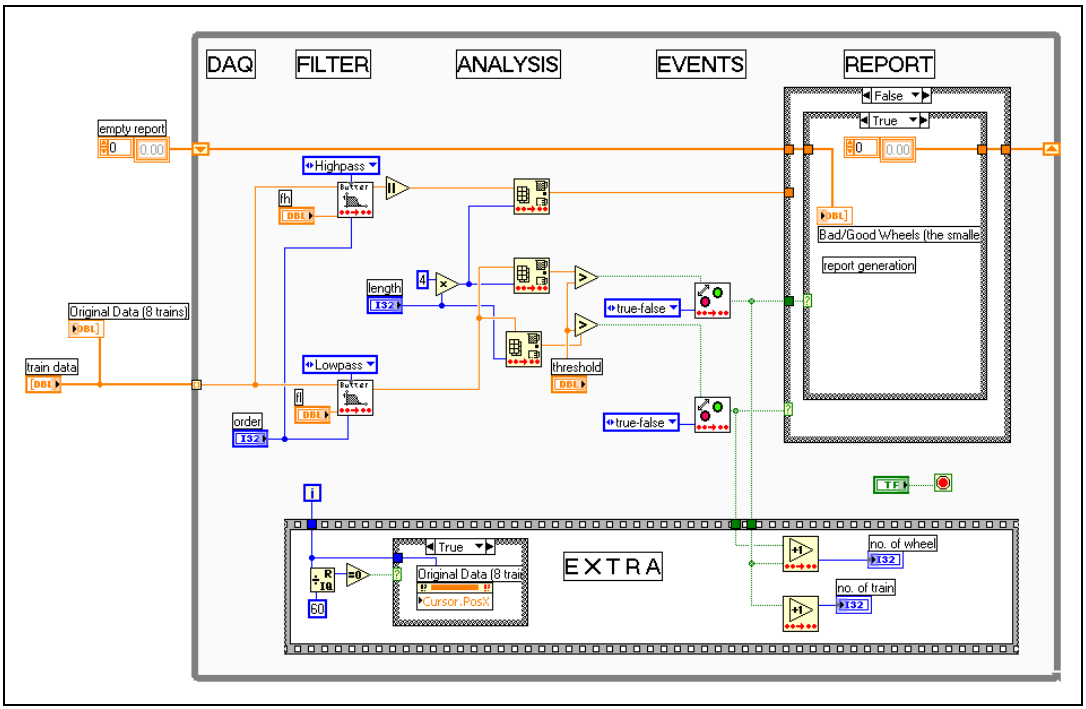


Figure 14-4. Block Diagram of the Train Wheel PtByPt VI

Note This example focuses on implementing a point-by-point analysis program in LabVIEW. The issues of ideal sampling periods and approaches to signal conditioning are beyond the scope of this example.

Overview of the LabVIEW Point-By-Point Solution

As well as Point By Point VIs, the Train Wheel PtByPt VI requires standard LabVIEW programming objects, such as Case structures, While Loops, numeric controls, and numeric operators.

The data the Train Wheel PtByPt VI acquires flows continuously through a While Loop. The process carried out by the Train Wheel PtByPt VI inside the While Loop consists of five analysis stages that occur sequentially. The following list reflects the order in which the five analysis stages occur,

briefly describes what occurs in each stage, and corresponds to the labeled portions of the block diagram in Figure 14-4.

1. In the data acquisition stage (DAQ), waveform data flows into the While Loop.
2. In the Filter stage, separation of low- and high-frequency components of the waveform occurs.
3. In the Analysis stage, detection of the train, wheel, and energy level of the waveform for each wheel occurs.
4. In the Events stage, responses to signal transitions of trains and wheels occurs.
5. In the Report stage, the logging of trains, wheels, and trains that might have defective wheels occurs.

Characteristics of a Train Wheel Waveform

The characteristic waveform that train wheels emit determines how you analyze and filter the waveform signal point-by-point. A train wheel in motion emits a signal that contains low- and high-frequency components. If you mount a strain gauge in a railroad track, you detect a noisy signal similar to a bell curve. Figure 14-5 shows the low- and high-frequency components of this curve.

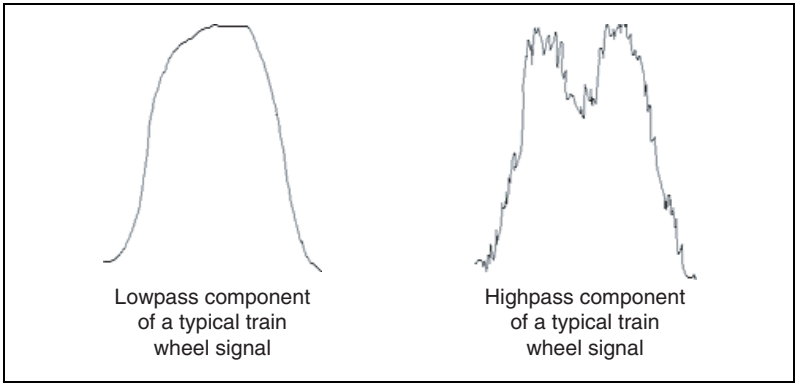


Figure 14-5. Low- and High-Frequency Components of a Train Wheel Signal

The low-frequency component of train wheel movement represents the normal noise of operation. Defective and normal wheels generate the same low-frequency component in the signal. The peak of the curve represents the moment when the wheel moves directly above the strain gauge. The lowest points of the bell curve represent the beginning and end of the wheel, respectively, as the wheel passes over the strain gauge.

The signal for a train wheel also contains a high-frequency component that reflects the quality of the wheel. In operation, a defective train wheel generates more energy than a normal train wheel. In other words, the high-frequency component for a defective wheel has greater amplitude.

Analysis Stages of the Train Wheel PtByPt VI

The waveform of all train wheels, including defective ones, falls within predictable ranges. This predictable behavior allows you to choose the appropriate analysis parameters. These parameters apply to the five stages described in the *Overview of the LabVIEW Point-By-Point Solution* section of this chapter. This section discusses each of the five analysis stages and the parameters use in each analysis stage.

Note You must adjust parameters for any implementation of the Train Wheel PtByPt VI because the characteristics of each data acquisition system differ.

DAQ Stage

Data moves into the Point By Point VIs through the **input data** parameter. The point-by-point detection application operates on the continuous stream of waveform data that comes from the wheels of a moving train. For a train moving at 60 km to 70 km per hour, a few hundred to a few thousand samples per second are likely to give you sufficient information to detect a defective wheel.

Filter Stage

The Train Wheel PtByPt VI must filter low- and high-frequency components of the train wheel waveform. Two Butterworth Filter PtByPt VIs perform the following tasks:

- Extract the low-frequency components of the waveform.
- Extract the high-frequency components of the waveform.

In the Train Wheel PtByPt VI, the Butterworth Filter PtByPt VIs use the following parameters:

- **order** specifies the amount of the waveform data that the VI filters at a given time and is the filter resolution. `2` is acceptable for the Train Wheel PtByPt.
- **fl** specifies the low cut-off frequency, which is the minimum signal strength that identifies the departure of a train wheel from the strain gauge. `0.01` is acceptable for the Train Wheel PtByPt.

- **fh** specifies the high cut-off frequency, which is the minimum signal strength that identifies the end of high-frequency waveform information. `0.25` is acceptable for the Train Wheel PtByPt.

Analysis Stage

The point-by-point detection application must analyze the low- and high-frequency components separately. The Array Max & Min PtByPt VI extracts waveform data that reveals the level of energy in the waveform for each wheel, the end of each train, and the end of each wheel.

Three separate Array Max & Min PtByPt VIs perform the following discrete tasks:

- Identify the maximum high-frequency value for each wheel.
- Identify the end of each train.
- Identify the end of each wheel.

Note The name Array Max & Min PtByPt VI contains the word array only to match the name of the array-based form of this VI. You do not need to allocate arrays for the Array Max & Min PtByPt VI.

In the Train Wheel PtByPt VI, the Array Max & Min PtByPt VIs use the following parameters and functions:

- **sample length** specifies the size of the portion of the waveform that the Train Wheel PtByPt VI analyzes. To calculate the ideal **sample length**, consider the speed of the train, the minimum distance between wheels, and the number of samples you receive per second. `100` is acceptable for the Train Wheel PtByPt VI. The Train Wheel PtByPt VI uses **sample length** to calculate values for all three Array Max & Min PtByPt VIs.
- The Multiply function sets a longer portion of the waveform to analyze. When this longer portion fails to display signal activity for train wheels, the Array Max & Min PtByPt VIs identify the end of the train. `4` is acceptable for the Train Wheel PtByPt VI.
- **threshold** provides a comparison point to identify when no train wheel signals exist in the signal that you are acquiring. **threshold** is wired to the Greater? function. `3` is an acceptable setting for **threshold** in the Train Wheel PtByPt VI.

Events Stage

After the Analysis stage identifies maximum and minimum values, the Events stage detects when these values cross a threshold setting.

The Train Wheel PtByPt VI logs every wheel and every train that it detects. Two Boolean Crossing PtByPt VIs perform the following tasks:

- Generate an event each time the Array Max & Min PtByPt VIs detect the transition point in the signal that indicates the end of a wheel.
- Generate an event every time the Array Max & Min PtByPt VIs detect the transition point in the signal that indicates the end of a train.

The Boolean Crossing PtByPt VIs respond to transitions. When the amplitude of a single wheel waveform falls below the **threshold** setting, the end of the wheel has arrived at the strain gauge. For the Train Wheel PtByPt VI, `3` is a good **threshold** setting to identify the end of a wheel. When the signal strength falls below the **threshold** setting, the Boolean Crossing PtByPt VIs recognize a transition event and pass that event to a report.

Analysis of the high-frequency signal identifies which wheels, if any, might be defective. When the Train Wheel PtByPt VI encounters a potentially defective wheel, the VI passes the information directly to the report at the moment the end-of-wheel event is detected.

In the Train Wheel PtByPt VI, the Boolean Crossing PtByPt VIs use the following parameters:

- **initialize** resets the VI for a new session of continuous data acquisition.
- **direction** specifies the kind of Boolean crossing.

Report Stage

The Train Wheel PtByPt VI reports on all wheels for all trains that pass through the data acquisition system. The Train Wheel PtByPt VI also reports any potentially defective wheels.

Every time a wheel passes the strain gauge, the Train Wheel PtByPt VI captures its waveform, analyzes it, and reports the event. Table 14-4 describes the components of a report on a single train wheel.

Table 14-4. Example Report on a Single Train Wheel

Information Source	Meaning of Results
Counter mechanism for waveform events	Stage One: Wheel number four has passed the strain gauge.
Analysis of highpass filter data	Stage Two: Wheel number four has passed the strain gauge and the wheel might be defective.
Counter mechanism for end-of-train events	Stage Three: Wheel number four in train number eight has passed the strain gauge, and the wheel might be defective.

The Train Wheel PtByPt VI uses point-by-point analysis to generate a report, not to control an industrial process. However, the Train Wheel PtByPt VI acquires data in real time, and you can modify the application to generate real-time control responses, such as stopping the train when the Train Wheel PtByPt VI encounters a potentially defective wheel.

Conclusion

When acquiring data with real-time performance, point-by-point analysis helps you analyze data in real time. Point-by-point analysis occurs continuously and instantaneously. While you acquire data, you filter and analyze it, point by point, to extract the information you need and to make an appropriate response. This case study demonstrates the effectiveness of the point-by-point approach for generation of both events and reports in real time.

A

References

This appendix lists the reference material used to produce the analysis VIs, including the signal processing and mathematics VIs. Refer to the following documents for more information about the theories and algorithms implemented in the analysis library.

Baher, H. *Analog & Digital Signal Processing*. New York: John Wiley & Sons, 1990.

Baily, David H. and Paul N. Swartztrauber. "The Fractional Fourier Transform and Applications." *Society of Industrial and Applied Mathematics Review* 33, no. 3 (September 1991): 389–404.

Bates, D. M. and D. G. Watts. *Nonlinear Regression Analysis and its Applications*. New York: John Wiley & Sons, 1988.

Bertsekas, Dimitri P. *Nonlinear Programming*. 2d ed. Belmont, Massachusetts: Athena Scientific, 1999.

Bracewell, R.N. "Numerical Transforms." *Science* 248, (11 May 1990).

Burden, R. L. and J. D. Faires. *Numerical Analysis*. 3d ed. Boston: Prindle, Weber & Schmidt, 1985.

Chen, C. H. et al. *Signal Processing Handbook*. New York: Marcel Decker, Inc., 1988.

Chugani, Mahesh L., Abhay R. Samant, and Michael Cerna. *LabVIEW Signal Processing*. Upper Saddle River, NJ: Prentice Hall PTR, 1998.

Crandall, R. E. *Projects in Scientific Computation*. Berlin: Springer, 1994.

DeGroot, M. *Probability and Statistics*. 2d ed. Reading, Massachusetts: Addison-Wesley Publishing Co., 1986.

Dowdy, S. and S. Wearden. *Statistics for Research*. 2nd ed. New York: John Wiley & Sons. 1991.

Dudewicz, E. J. and S. N. Mishra. *Modern Mathematical Statistics.* New York: John Wiley & Sons, 1988.

Duhamel, P. et al. “On Computing the Inverse DFT.” *IEEE Transactions*.

Dunn, O. and V. Clark. *Applied Statistics: Analysis of Variance and Regression*. 2nd ed. New York: John Wiley & Sons. 1987.

Ecker, Joseph G. and Michael Kupferschmid. *Introduction to Operations Research*. New York: Krieger Publishing, 1991.

Elliot, D. F. *Handbook of Digital Signal Processing Engineering Applications*. San Diego: Academic Press, 1987.

Fahmy, M. F. “Generalized Bessel Polynomials with Application to the Design of Bandpass Filters.” *Circuit Theory and Applications* 5, (1977): 337–342.

Gander, W. and J. Hrebicek. *Solving Problems in Scientific Computing using Maple and MATLAB*. Berlin: Springer, 1993.

Golub, G.H. and C. F. Van Loan. *Matrix Computations*. Baltimore: The John Hopkins University Press, 1989.

Harris, Fredric J. “On the Use of Windows for Harmonic Analysis with the Discrete Fourier Transform.” *Proceedings of the IEEE* 66, no. 1 (1978).

Lanczos, C. A. “Precision Approximation of the Gamma Function.” *Journal SIAM Numerical Analysis* series B, no. 1 (1964): 87–96.

Maisel, J. E. “Hilbert Transform Works With Fourier Transforms to Dramatically Lower Sampling Rates.” *Personal Engineering and Instrumentation News* 7, no. 2 (February 1990).

Miller, I. and J. E. Freund. *Probability and Statistics for Engineers*. Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1987.

Mitra, Sanjit K. and James F. Kaiser. *Handbook for Digital Signal Processing*. New York: John Wiley & Sons, 1993.

Neter, J. et al. *Applied Linear Regression Models*. Richard D. Irwin, Inc., 1983.

Neuvo, Y., C. Y. Dong, and S. K. Mitra. “Interpolated Finite Impulse Response Filters” *IEEE Transactions on ASSP* ASSP-32, no. 6 (June, 1984).

O’Neill, M. A. “Faster Than Fast Fourier.” *BYTE* (April 1988).

Oppenheim, A. V. and R. W. Schafer. *Discrete-Time Signal Processing*. Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1989.

Oppenheim, Alan V. and Alan S. Willsky. *Signals and Systems*. New York: Prentice-Hall, Inc., 1983.

Parks, T. W. and C. S. Burrus. *Digital Filter Design*. New York: John Wiley & Sons, Inc., 1987.

Pearson, C. E. *Numerical Methods in Engineering and Science*. New York: Van Nostrand Reinhold Co., 1986.

Press, William H., Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C, The Art of Scientific Computing*. Cambridge: Cambridge University Press, 1994.

Qian, Shie and Dapang Chen. *Joint Time-Frequency Analysis*. New York: Prentice-Hall, Inc., 1996.

Rabiner, L. R. and B. Gold. *Theory and Application of Digital Signal Processing*. Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1975.

Rockey, K. C., H. R. Evans, D. W. Griffiths, and D. A. Nethercot. *The Finite Element Method—A Basic Introduction for Engineers*. New York: John Wiley & Sons, 1983.

Sorensen, H. V. et al. "On Computing the Split-Radix FFT." *IEEE Transactions on ASSP* ASSP-34, no. 1 (February 1986).

Sorensen, H. V. et al. "Real-Valued Fast Fourier Transform Algorithms." *IEEE Transactions on ASSP* ASSP-35, no. 6 (June 1987).

Spiegel, M. *Schaum's Outline Series on Theory and Problems of Probability and Statistics*. New York: McGraw-Hill, 1975.

Stoer, J. and R. Bulirsch. *Introduction to Numerical Analysis*. New York: Springer-Verlag, 1987.

Vaidyanathan, P. P. *Multirate Systems and Filter Banks*. Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1993.

Wichman, B. and D. Hill. "Building a Random-Number Generator: A Pascal Routine for Very-Long-Cycle Random-Number Sequences." *BYTE* (March 1987): 127–128.

Wilkinson, J. H. and C. Reinsch. *Linear Algebra, Vol. 2 of Handbook for Automatic Computation*. New York: Springer, 1971.

Williams, John R. and Kevin Amaratunga. “Introduction to Wavelets in Engineering.” *International Journal for Numerical Methods in Engineering* 37, (1994): 2365–2388.

Zwillinger, Daniel. *Handbook of Differential Equations*. San Diego: Academic Press, 1992.

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 include the following:
  - **Self-Help Resources**—For immediate answers and solutions, visit our extensive library of technical support resources available in English, Japanese, and Spanish at ni.com/support. These resources are available for most products at no cost to registered users and include software drivers and updates, a KnowledgeBase, product manuals, step-by-step troubleshooting wizards, conformity documentation, example code, tutorials and application notes, instrument drivers, discussion forums, a measurement glossary, and so on.
  - **Assisted Support Options**—Contact NI engineers and other measurement and automation professionals by visiting ni.com/support. Our online system helps you define your question and connects you to the experts by phone, discussion forum, or email.
- **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, NI Alliance Program 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.