NATIONAL INSTRUMENTS™
LabVIEW™

Analysis Concepts

NATIONAL INSTRUMENTS™

July 2000 Edition
Part Number 370192A-01

Important Information

Contents

Figures

Tables

About This Manual

This manual provides information about analysis and mathematical concepts in LabVIEW.

Conventions

The following conventions appear in this manual:

»	The » symbol leads you through nested menu items and dialog box options to a final action. The sequence **File»Page Setup»Options** directs you to pull down the **File** menu, select the **Page Setup** item, and select **Options** from the last dialog box.
✎	This icon denotes a note, which alerts you to important information.
bold	Bold text denotes items that you must select or click on 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, and code excerpts.

Related Documentation

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

- *LabVIEW Measurements Manual*
- *LabVIEW Help*, available by selecting **Help»Contents and Index**
- *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)

1

Signal Generation

This chapter explains how to produce signals using normalized frequency and how to build a simulated function generator. Signal generation VIs are available on the **Functions»Analyze»Signal Processing»Signal Generation** palette.

Some of the applications for signal generation are:

- Simulating 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).
- Generating signals to apply to a D/A converter.

Normalized Frequency

In the analog world, a signal frequency is measured in 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:

digital frequency = analog frequency / sampling frequency

This digital frequency is known as the *normalized* frequency. Its units are cycles/sample.

Some of the Signal Generation VIs use an input frequency control, *f*, that is assumed to use *normalized frequency* units of *cycles per sample*. This frequency ranges from 0.0 to 1.0, which corresponds to a real frequency range of 0 to the sampling frequency f_s. This frequency also wraps around 1.0, so that a normalized frequency of 1.1 is equivalent to 0.1. As an example, a signal that is sampled at the Nyquist rate (f_s/2) means that it is sampled twice per cycle (that is, two samples/cycle). This will correspond 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 that the signal is sampled in one 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/sample.

You must use these normalized units 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/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 2 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 may need to use frequency units of Hz (cycles/s). If you need to convert from Hz (or cycles/s) to cycles/sample, divide your frequency in cycles/s by the sampling rate given in samples/s.

$$\frac{\text{cycles/s}}{\text{samples/s}} = \frac{\text{cycles}}{\text{sample}}$$

For example, you divide a frequency of 60 Hz by a sampling rate of 1000 Hz to get the normalized frequency of $f = 0.06$ cycles/sample. Therefore, it takes almost 17 (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 can also use the signal generation VIs in conjunction with National Instruments hardware to generate analog output signals.

Wave and Pattern VIs

You will notice that the names of most of the signal generation VIs have the word *wave* or *pattern* in them. There is a basic difference in the operation of the two different types of VIs. It has to do with whether or not the VI can keep track of the phase of the signal that it generates each time it is called.

Phase Control

The wave VIs have a **phase in** control where you can specify the initial phase (in degrees) of the first sample of the generated waveform. They also have a **phase out** indicator that specifies what the phase of the next sample of the generated waveform is going to be. In addition, a **reset phase** control decides whether or not the phase of the first sample generated when the wave VI is called is the phase specified at the **phase in** control, or whether it is the phase available at the **phase out** control when the VI last executed. A TRUE value of **reset phase** sets the initial phase to **phase in**, whereas a FALSE value sets it to the value of **phase out** when the VI last executed.

The wave VIs are all reentrant (can keep track of phase internally) and accept frequency in normalized units (cycles/sample). The only pattern VI that presently uses normalized units is the Chirp Pattern VI. Setting the **reset phase** Boolean to FALSE allows for continuous sampling simulation.

Note Wave VIs are reentrant and accept the frequency input in terms of normalized units.

2

Digital Signal Processing

This chapter describes the fundamentals of the Fast Fourier Transform (FFT) and the Discrete Fourier Transform (DFT) and how they are used in spectral analysis. Refer to the examples in `examples\analysis\dspxmpl.llb` for examples of using the digital signal processing VIs, available on the **Functions»Analyze»Signal Processing»Frequency Domain** palette.

The Fast Fourier Transform (FFT)

The samples of a signal obtained from a DAQ device constitute the *time domain* representation of the signal. This representation gives the amplitudes of the signal at the instants of time during which it had been sampled. However, in many cases you want to know the frequency content of a signal rather than the amplitudes of the individual samples. The representation of a signal in terms of its individual frequency components is known as the *frequency domain* representation of the signal. The frequency domain representation could give more insight about the signal and the system from which it was generated.

The algorithm used to transform samples of the data from the time domain into the frequency domain is known as the *discrete Fourier transform* or 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.

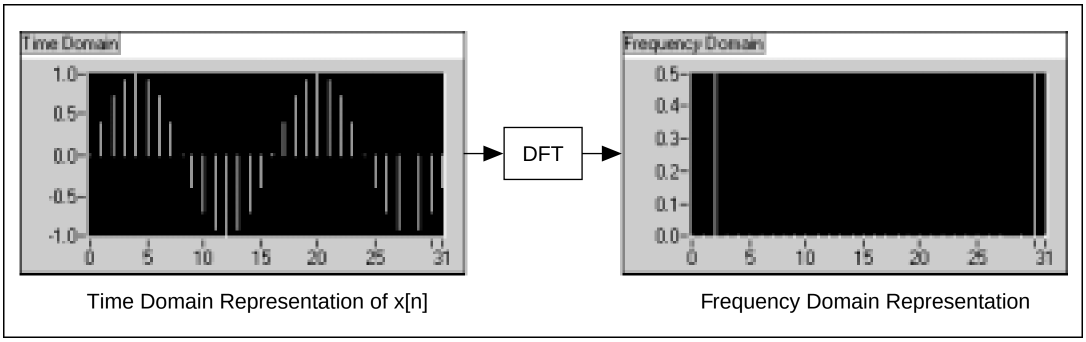


Figure 2-1. Discrete Fourier Transform

Suppose you have 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 is also of length *N* samples, but the information it contains is of the frequency domain representation. The relationship between the *N* samples in the time domain and the *N* samples in the frequency domain is explained below.

If the signal is sampled at a sampling rate of f_s Hz, then the time interval between the samples (that is, the sampling interval) is Δt, where

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

The sample signals are denoted by $x[i],\ 0 \le i \le N-1$ (that is, you have a total of *N* samples). When the discrete Fourier transform, given by

$$X_k = \sum_{i=0}^{N-1} x_i e^{-j2\pi ik/N} \qquad (2\text{-}1)$$

for

$$k = 0, 1, 2, \ldots, N-1$$

is applied to these *N* samples, the resulting output ($X[k],\ 0 \le k \le N-1$) is the frequency domain representation of $x[i]$. Notice that both the time domain *x* and the frequency domain *X* have a total of *N* samples. Analogous to the time spacing of Δt between the samples of x in the time domain, you have a frequency spacing of

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

between the components of X in the frequency domain. Δf is also known as the *frequency resolution*. To increase the frequency resolution (smaller Δf) you must either increase the number of samples N (with f_s constant) or decrease the sampling frequency f_s (with N constant).

In the following example, you will go through the mathematics of Equation 2-1 to calculate the DFT for a D.C. signal.

DFT Calculation Example

In the next section, you will see the exact frequencies to which the N samples of the DFT correspond. For the present discussion, assume that $X[0]$ corresponds to D.C., or the average value, of the signal. To see the result of calculating the DFT of a waveform with the use of Equation 2-1, consider a D.C. signal having a constant amplitude of +1 V. Four samples of this signal are taken, as shown in Figure 2-2.

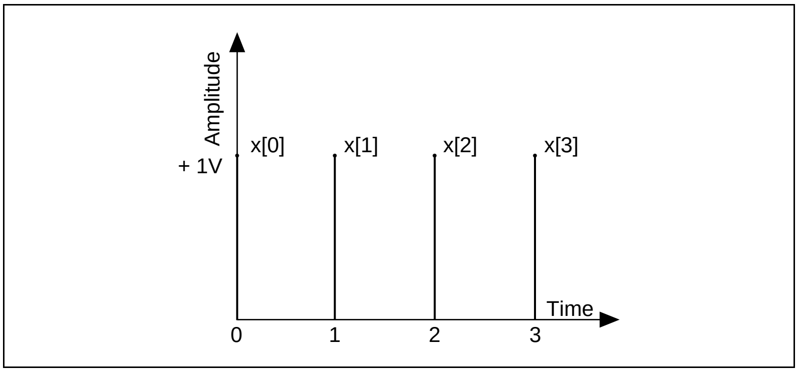


Figure 2-2. DFT Samples

Each of the samples has a value +1, giving the time sequence

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

Using Equation 2-1 to calculate the DFT of this sequence and making use of Euler's identity,

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

you get:

$$X[0] = \sum_{i=0}^{N-1} x_i e^{-j2\pi i0/N} = x[0] + x[1] + x[2] + x[3] = 4$$

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

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

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

Therefore, except for the DC component, *X*[0], all the other values are zero, which is as expected. However, the calculated value of *X*[0] depends on the value of *N* (the number of samples). Because you had *N* = 4, X[0] = 4. If *N* = 10, then you would have calculated *X*[0] = 10. This dependency of *X*[ ] on *N* also occurs for the other frequency components. Thus, you usually divide the DFT output by *N*, so as to obtain the correct magnitude of the frequency component.

Magnitude and Phase Information

You have seen that *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. From Equation 2-1, you see 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). Thus, because the DFT is complex, it contains two pieces of information—the amplitude and the phase. It turns out that for real signals (*x*[*i*] real) such as those obtained from the output of one channel of a DAQ device, the DFT is symmetric with the following properties:

$$|X[\mathrm{k}]| = |X[N-k]|$$

and

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

The terms used to describe this symmetry are that the magnitude of $X[k]$ is *even symmetric*, and phase($X[k]$) is *odd symmetric*. An even symmetric signal is one that is symmetric about the y-axis, whereas an odd symmetric signal is symmetric about the origin. This is shown in the following figures.

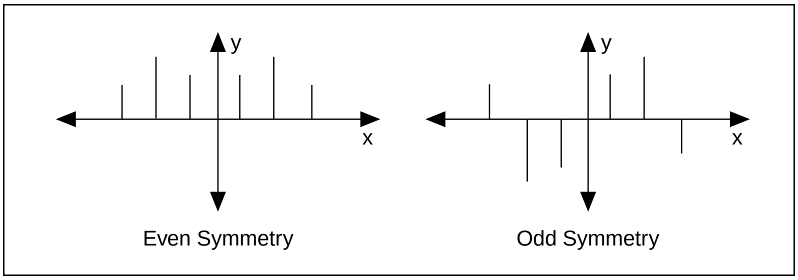


Figure 2-3. Signal Symmetry about the y-axis

The net effect of this symmetry is that there is repetition of information contained in the N samples of the DFT. Because of this repetition of information, only half of the samples of the DFT actually need to be computed or displayed, as the other half can be obtained from this repetition. If the input signal is complex, the DFT will be nonsymmetric and you cannot use this trick.

Frequency Spacing between DFT/FFT Samples

If the sampling interval is Δt seconds, and the first ($k = 0$) data sample is at 0 seconds, then the k^{th} ($k > 0$, k integer) data sample is at $k\Delta t$ seconds. Similarly, if the frequency resolution is Δf *Hz*.

($\Delta f = \frac{f_s}{N}$) then the k^{th} sample of the DFT occurs at a frequency of $k\Delta f$ *Hz*. (Actually, as you will soon see, 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, suppose N is even and let $p = \frac{N}{2}$. Table 2-1 shows the frequency to which each format element of the complex output sequence X corresponds.

Note that the p^{th} element, $X[p]$, corresponds to the Nyquist frequency. The negative entries in the second column beyond the Nyquist frequency represent negative frequencies.

For example, if $N = 8$, $p = N/2 = 4$, then Δf is shown in Table 2-1 for $X[p]$ for $N = 8$.

Table 2-1. $X[p]$ 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)
$X[5]$	$-3\Delta f$
$X[6]$	$-2\Delta f$
$X[7]$	$-\Delta f$

Here, $X[1]$ and $X[7]$ will have the same magnitude, $X[2]$ and $X[6]$ will have the same magnitude, and $X[3]$ and $X[5]$ will have the same magnitude. The difference is that whereas $X[1]$, $X[2]$, and $X[3]$ correspond to positive frequency components, $X[5]$, $X[6]$, and $X[7]$ correspond to negative frequency components. Note that $X[4]$ is at the Nyquist frequency.

Figure 2-4 represents this complex sequence for $N = 8$.

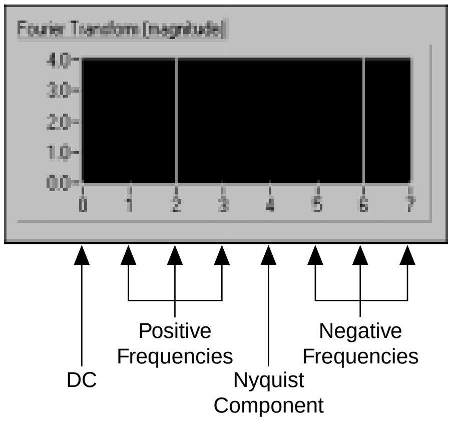


Figure 2-4. Complex Sequence for $N = 8$

Such a representation, where you see both the positive and negative frequencies, is known as the *two-sided* transform.

Note that when N is odd, there is no component at the Nyquist frequency.

For example, if $N = 7$, $p = (N–1)/2 = (7–1)/2 = 3$, then Δf is shown in Table 2-2 for $X[p]$ for $N = 7$.

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

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

Now $X[1]$ 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, whereas $X[1]$, $X[2]$, and $X[3]$ correspond to positive frequencies, $X[4]$, $X[5]$, and $X[6]$ correspond to negative frequencies. Because N is odd, there is no component at the Nyquist frequency.

Figure 2-5 illustrates Table 2-2 for $N = 7$.

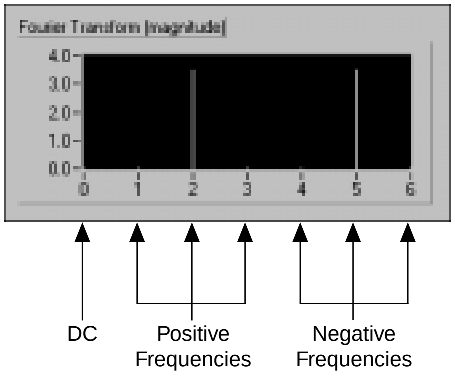


Figure 2-5. $X[p]$ for $N = 7$

This is also a two-sided transform, because you have both the positive and negative frequencies.

Fast Fourier Transforms

Direct implementation of the DFT on N data samples requires approximately N^2 complex operations and is a time-consuming process. However, when the size of the sequence is a power of 2,

$$N = 2^m \text{ for } m = 1, 2, 3, \ldots$$

you can implement the computation of the DFT with approximately $N \log_2(N)$ operations. This makes the calculation of the DFT much faster, and DSP literature refers to these algorithms as fast Fourier transforms (FFTs). The FFT is nothing but a fast algorithm for calculating the DFT when the number of samples (N) is a power of 2.

The advantages of the FFT include speed and memory efficiency, because the VI can compute the FFT in place, that is, no additional memory buffers are needed to compute the output. The size of the input sequence, however, must be a power of 2. The DFT can efficiently process any size sequence, but the DFT is slower than the FFT and uses more memory, because it must allocate additional buffers for storing intermediate results during processing.

Zero Padding

A technique employed to make the input sequence size equal to a power of 2 is to add zeros to the end of the sequence so that the total number of samples is equal to the next higher power of 2. 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 (= 2^4—a power of 2), as shown in Figure 2-6.

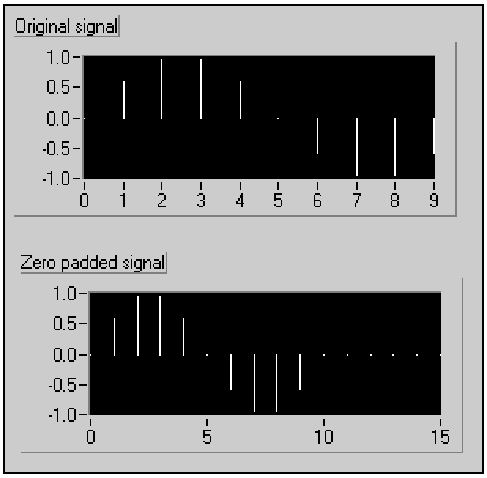


Figure 2-6. Zero Padding

The addition of zeros to the end of the time domain waveform does not affect the spectrum of the signal. 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 also helps in increasing the frequency resolution (recall that $\Delta f = f_s/N$) by increasing the number of samples, N.

FFT VIs

The **Functions»Analyze»Signal Processing»Frequency Domain** palette contains two VIs that compute the FFT of a signal. They are the Real FFT VI and Complex FFT VI.

The difference between the two VIs is that the Real FFT VI computes the FFT of a real-valued signal, whereas the Complex FFT VI computes the FFT of a complex-valued signal. However, keep in mind that the outputs of both VIs are complex.

Most real-world signals are real valued, and hence you can use the Real FFT VI for most applications. Of course, you could also use the Complex FFT VI by setting the imaginary part of the signal to zero.

An example of an application where you could use the Complex FFT VI is when the signal consists of both a real and imaginary component. Such a type of signal 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 2-7.

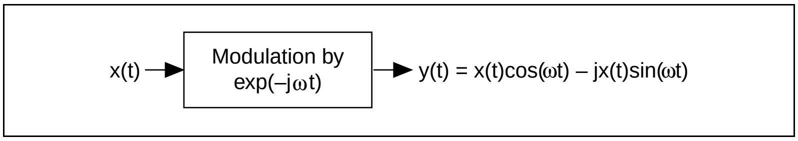


Figure 2-7. Modulation by a Complex Exponential

The Power Spectrum

You have seen that the DFT (or FFT) of a real signal is a complex number, having a real and an imaginary part. The *power* in each frequency component represented by the DFT/FFT can be obtained by squaring the magnitude of that frequency component. Thus, the power in the k^{th} frequency component (the k^{th} element of the DFT/FFT) is given by $|X[k]|^2$. The plot showing the power in each of the frequency components is known as the *power spectrum*. Because the DFT/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 $-k\Delta f$ (DC and Nyquist components not included). The total power in the DC

and Nyquist components are $|X[0]|^2$ and $\left|X\left[\frac{N}{2}\right]\right|^2$, respectively.

Loss of Phase Information

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

You can use the power spectrum in applications where phase information is not necessary (for example, to calculate 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.

Frequency Spacing between Samples

You can use the Power Spectrum **VI**, available on the **Functions»Analyze»Signal Processing»Frequency Domain** palette, to calculate the power spectrum of the time domain data samples. Just like the DFT/FFT, the number of samples from the Power Spectrum VI output is the same as the number of data samples applied at the input. Also, the frequency spacing between the output samples is $\Delta f = f_s/N$.

Summary

The time domain representation (sample values) of a signal can be converted into the frequency domain representation by means of an algorithm known as the discrete Fourier transform (DFT). To have fast calculation of the DFT, an algorithm known as the fast Fourier transform (FFT) is used. You can use this algorithm when the number of signal samples is a power of two.

The output of the conventional DFT/FFT is two-sided because it contains information about both the positive and the negative frequencies. This output can be converted into a one-sided DFT/FFT by using only half the DFT/FFT output points. The frequency spacing between the samples of the DFT/FFT is $\Delta f = f_s/N$.

The power spectrum can be calculated from the DFT/FFT by squaring the magnitude of the individual frequency components. The Power Spectrum VI in the advanced analysis library does this automatically for you. The units of the output of the Power Spectrum VI are in $V_{rms}{}^2$. However, the power spectrum does not provide any phase information.

The DFT, FFT, and power spectrum are useful for measuring the frequency content of stationary or transient signals. The FFT provides the average frequency content of the signal over the entire time that the signal was acquired.

3 Smoothing Windows

This chapter explains how using windows prevents spectral leakage and improves the analysis of acquired signals. Refer to the example in `examples\analysis\windxmpl.llb` for an example of how to use the analysis window VIs, available on the **Functions»Analyze»Signal Processing»Windows** palette.

Introduction to Smoothing Windows

In practical signal-sampling applications, you can obtain only a finite record of the signal, even when you carefully observe the sampling theorem and sampling conditions. Unfortunately for the discrete-time system, the finite sampling record results in a truncated waveform that has different spectral characteristics from the original continuous-time signal. These discontinuities produce leakage of spectral information, resulting in a discrete-time spectrum that is a smeared version of the original continuous-time spectrum.

A simple way to improve the spectral characteristics of a sampled signal is to apply smoothing windows. When performing Fourier or spectral analysis on finite-length data, you can use windows to minimize the transition edges of your truncated waveforms, thus reducing spectral leakage. When used in this manner, smoothing windows act like predefined, narrowband, lowpass filters.

About Spectral Leakage and Smoothing Windows

When you use the DFT/FFT to find the frequency content of a signal, it is inherently assumed that the data that you have is a single period of a periodically repeating waveform, as shown in Figure 3-1. The first period shown is the one sampled. The waveform corresponding to this period is then repeated in time to produce the periodic waveform.

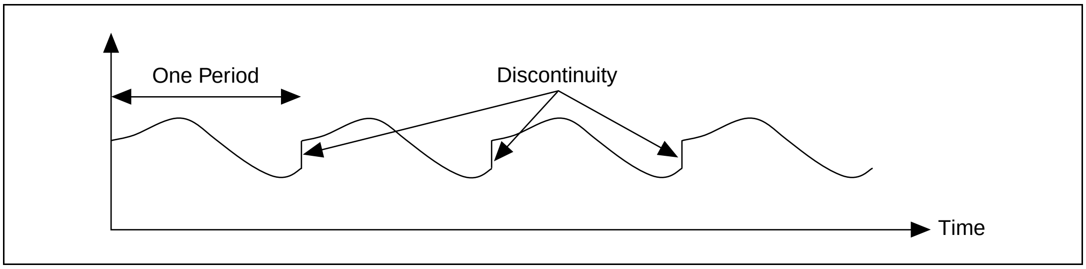
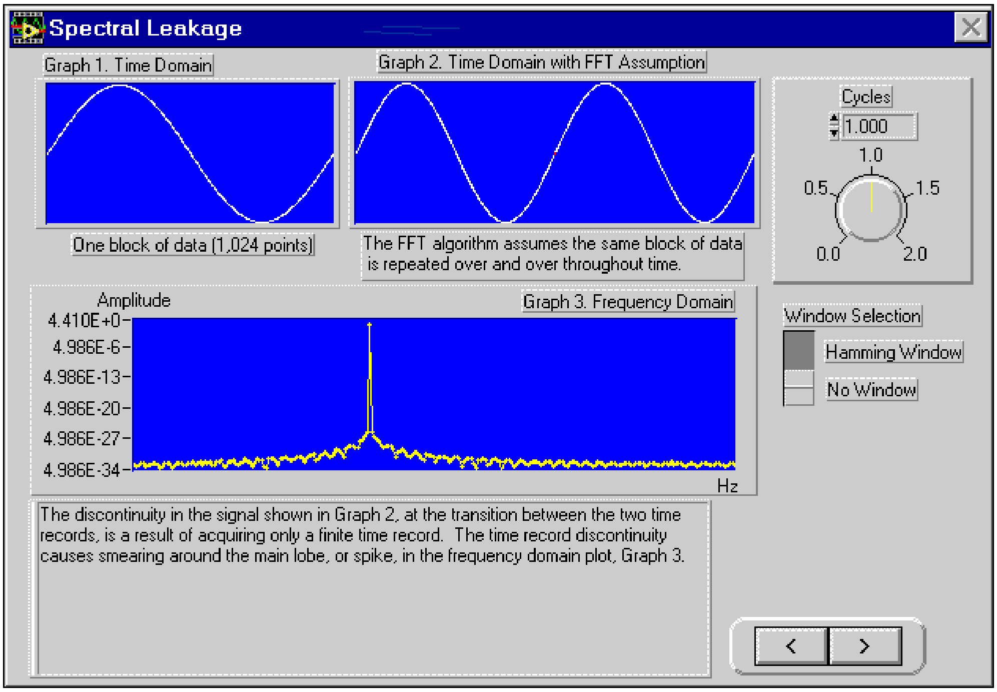


Figure 3-1. Periodic Waveform Created from Sampled Period

Because of the assumption of periodicity of the waveform, discontinuities between successive periods will occur. This happens when you sample a noninteger number of cycles. These *artificial* discontinuities turn up as very high frequencies in the spectrum of the signal, frequencies that were not present in the original signal. These frequencies could be much higher than the Nyquist frequency, and as you have seen before, will be aliased somewhere between 0 and $f_s/2$. The spectrum you get by using the DFT/FFT therefore will not be the actual spectrum of the original signal, but will be a smeared version. It appears as if the energy at one frequency has *leaked out* into all the other frequencies. This phenomenon is known as *spectral leakage*.

Figure 3-2 shows a sine wave and its corresponding Fourier transform. The sampled time domain waveform is shown in Graph 1. Because the Fourier transform assumes periodicity, you repeat this waveform in time, and the periodic time waveform of the sine wave of Graph 1 is shown in Graph 2. The corresponding spectral representation is shown in Graph 3. Because the time record in Graph 2 is periodic, with no discontinuities, its spectrum is a single line showing the frequency of the sine wave. The reason that the waveform in Graph 2 does not have any discontinuities is because you have sampled an integer number of cycles (in this case, 1) of the time waveform.



One block of data (1,024 points)

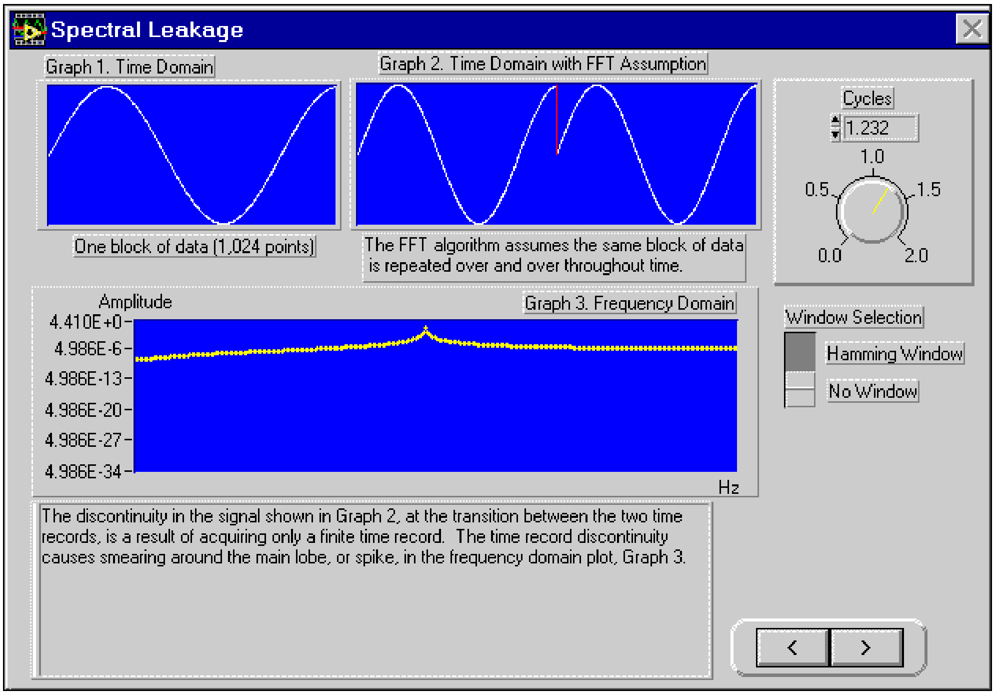
The FFT algorithm assumes the same block of data is repeated over and over throughout time.

Cycles: 1.000

The discontinuity in the signal shown in Graph 2, at the transition between the two time records, is a result of acquiring only a finite time record. The time record discontinuity causes smearing around the main lobe, or spike, in the frequency domain plot, Graph 3.

Figure 3-2. Sine Wave and Corresponding Fourier Transform

In Figure 3-3, you see the spectral representation when you sample a noninteger number of cycles of the time waveform (namely 1.25). Graph 1 now consists of 1.25 cycles of the sine wave. When you repeat this periodically, the resulting waveform, as shown in Graph 2, consists of discontinuities. The corresponding spectrum is shown in Graph 3. Notice how the energy is now spread over a wide range of frequencies. This smearing of the energy is spectral leakage. The energy has leaked out of one of the FFT lines and smeared itself into all the other lines.



The FFT algorithm assumes the same block of data is repeated over and over throughout time.

The discontinuity in the signal shown in Graph 2, at the transition between the two time records, is a result of acquiring only a finite time record. The time record discontinuity causes smearing around the main lobe, or spike, in the frequency domain plot, Graph 3.

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

Leakage exists because of the finite time record of the input signal. To overcome leakage, one solution is to take an infinite time record, from –infinity to +infinity. Then the FFT would calculate one single line at the correct frequency. Waiting for infinite time is, however, not possible in practice. So, because you are limited to having a finite time record, another technique, known as *windowing*, is used to reduce the spectral leakage.

The amount of spectral leakage depends on the amplitude of the discontinuity. The larger the discontinuity, the more the leakage, and vice versa. You can use windowing to reduce the amplitude of the discontinuities at the boundaries of each period. It consists of multiplying the time record by a finite length window whose amplitude varies smoothly and gradually towards zero at the edges. This is shown in Figure 3-4, where the original time signal is windowed using a *Hamming* window. Notice that the time waveform of the windowed signal gradually tapers to zero at the ends. Therefore, when performing Fourier or spectral analysis on finite-length data, you can use windows to minimize the transition edges of your sampled waveform. A smoothing window function applied to the data before it is transformed into the frequency domain minimizes spectral leakage.

Note that if the time record contains an integral number of cycles, as shown in Figure 3-2, then the assumption of periodicity does not result in any discontinuities, and thus there is no spectral leakage. The problem arises only when you have a nonintegral number of cycles.

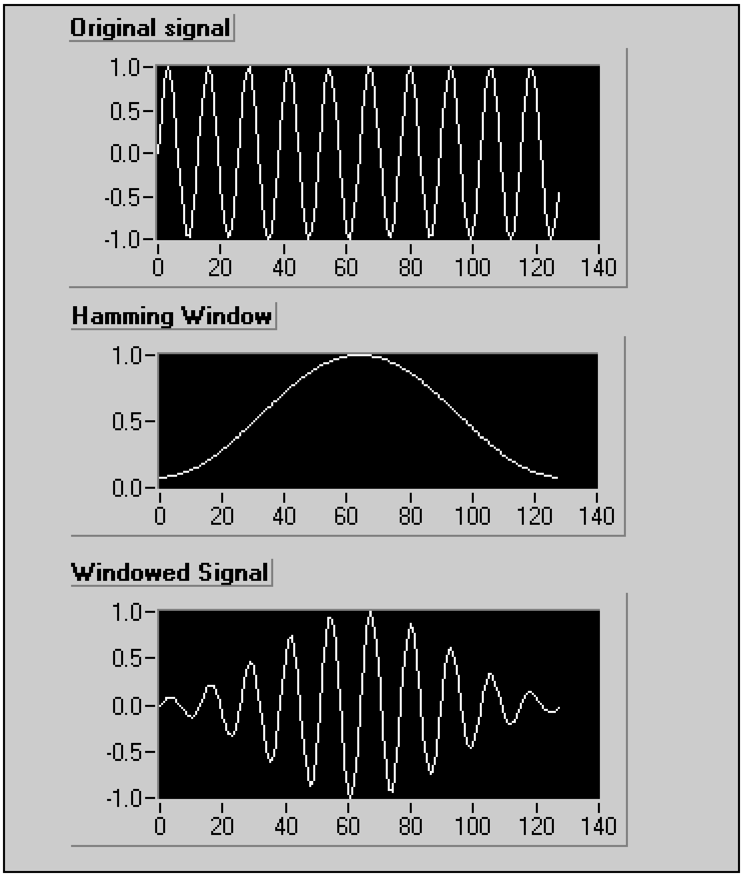


Figure 3-4. Time Signal Windowed Using a Hamming Window

Windowing Applications

There are several reasons to use windowing. Some of these are:

- To define the duration of the observation.
- Reduction of spectral leakage.
- Separation of a small amplitude signal from a larger amplitude signal with frequencies very close to each other.

Characteristics of Different Types of Window Functions

Applying a window to (windowing) a signal in the time domain is equivalent to multiplying the signal by the window function. Because multiplication in the time domain is equivalent to convolution in the frequency domain, the spectrum of the windowed signal is a convolution of the spectrum of the original signal with the spectrum of the window. Thus, windowing changes the shape of the signal in the time domain, as well as affecting the spectrum that you see.

Many different types of windows are available on the **Functions»Analyze» Signal Processing»Windows** palette. Depending on your application, one may be more useful than the others. Some of these windows are:

Rectangular (None)

The rectangular window has a value of one over its time interval. Mathematically, it can be written as:

$$w(n) = 1.0$$

for

$$n = 0, 1, 2........N-1$$

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

The rectangular window for $N = 32$ is shown in Figure 3-5.

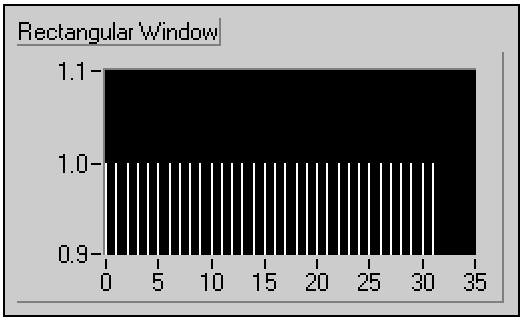


Figure 3-5. Rectangular Window

The rectangular window is useful for analyzing transients that have a duration shorter than that of the window. It is also used in *order tracking*, where the effective sampling rate is proportional to the speed of the shaft in rotating machines. In this application, it detects the main mode of vibration of the machine and its harmonics.

Hanning

This window has a shape similar to that of half a cycle of a cosine wave. Its defining equation is

$$w(n) = 0.5 - 0.5\cos\frac{2\pi n}{N}$$

for

$$n = 0, 1, 2, ...N-1$$

A Hanning window with $N = 32$ is shown in Figure 3-6.

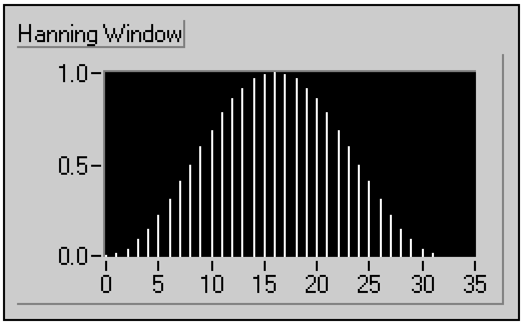


Figure 3-6. Hanning Window

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

Hamming

This window is a modified version of the Hanning window. Its shape is also similar to that of a cosine wave. It can be defined as

$$w(n) = 0.54 - 0.46\cos\frac{2\pi n}{N}$$

for

$$n = 0, 1, 2, ...N-1$$

A Hamming window with $N = 32$ is shown in Figure 3-7.

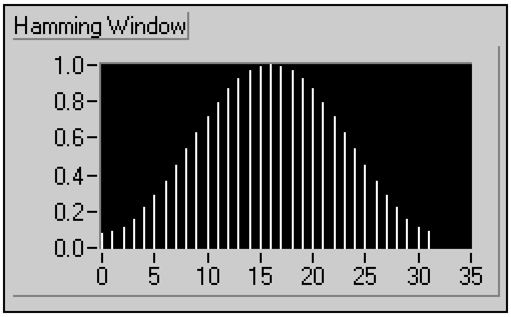


Figure 3-7. Hamming Window

You see that the Hanning and Hamming windows are somewhat similar. However, note that in the time domain, the Hamming window does not get as close to zero near the edges as does the Hanning window.

Kaiser-Bessel

This window is a “flexible” window whose shape the user can modify by adjusting the parameter *beta*. Thus, depending on your application, you can change the shape of the window to control the amount of spectral leakage. The Kaiser-Bessel window for different values of beta are shown in Figure 3-8.

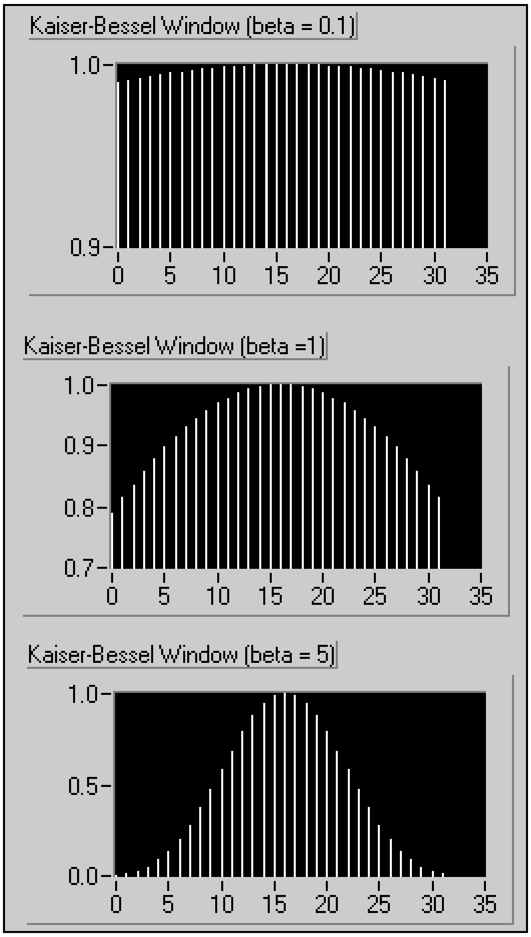


Figure 3-8. Kaiser-Bessel Window

Note that 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.

This window is good for detecting two signals of almost the same frequency, but significantly different amplitudes.

Triangle

The shape of this window is that of a triangle. It is given by

$$w(n) = 1 - \left|\frac{2n - N}{N}\right|$$

for

$$n = 0, 1, 2, ..., n-1$$

A triangle window for $N = 32$ is shown in Figure 3-9.

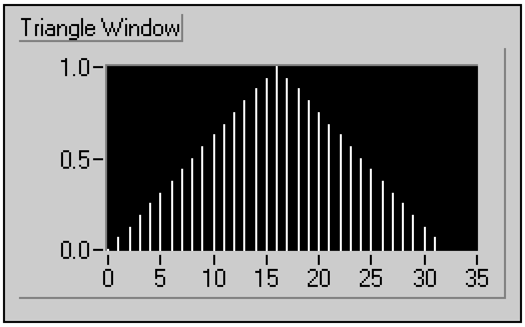


Figure 3-9. Triangle Window

Flat Top

This window has the best amplitude accuracy of all the window functions. The increased amplitude accuracy (± 0.02 dB for signals exactly between integral cycles) is at the expense of frequency selectivity. The Flattop window is most useful in accurately measuring the amplitude of single frequency components with little nearby spectral energy in the signal. The Flattop window can be defined as

$$w(n) = a_0 - \left(a_1 \cos\frac{2\pi n}{N} + a_2 \cos\frac{4\pi n}{N}\right)$$

where

$$a_0 = 0.2810638602$$
$$a_1 = 0.5208971735$$
$$a_2 = 0.1980389663$$

A flattop window is shown in Figure 3-10.

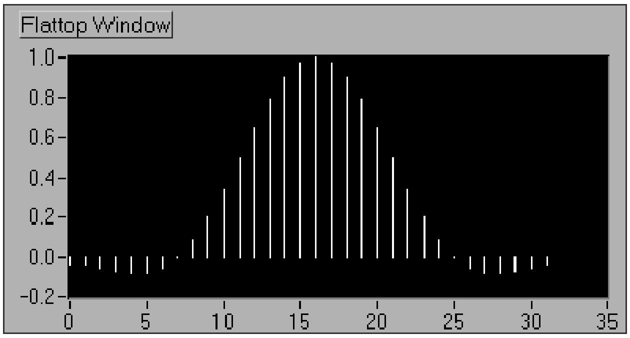


Figure 3-10. Flattop Window

Exponential

The shape of this window is that of a decaying exponential. It can be mathematically expressed as:

$$w[n] = e^{\left(\frac{n\ln(f)}{N-1}\right)} = f^{\left(\frac{n}{N-1}\right)}$$

for

$$n = 0, 1, 2.......N-1$$

where *f* is the final value. The initial value of the window is one, and it gradually decays towards zero. The final value of the exponential can be adjusted to between 0 and 1. The exponential window for N = 32, with the final value specified as 0.1, is shown in Figure 3-11.

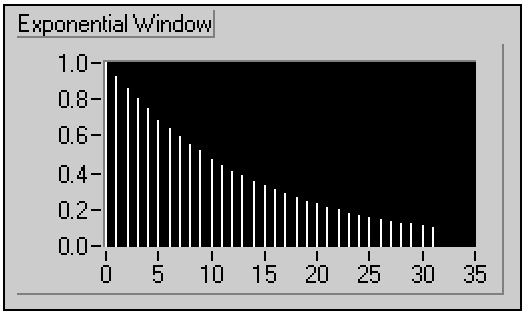


Figure 3-11. Exponential Window

This window is useful in analyzing transients (signals that exist only for a short time duration) whose duration is longer than the length of the window. This window can be applied to signals that decay exponentially, such as the response of structures with light damping that are excited by an impact (for example, a hammer).

Windows for Spectral Analysis Versus Windows for Coefficient Design

The window VIs in LabVIEW are designed for spectral analysis applications. In these applications, the input signal is windowed by passing it through one of the window VIs. The windowed signal is then passed to a DFT-based VI for frequency-domain display and analysis.

The window functions designed for spectral analysis must be *DFT-even*, a term defined by Fredric J. Harris in his paper *On the Use of Windows for Harmonic Analysis with the Discrete Fourier Transform* (*Proceedings of the IEEE*, Volume 66, No. 1, January 1978). A window function is DFT-even if its dot product (inner product) with integral cycles of sine sequences is identically zero. Another way to think of a DFT-even sequence is that its DFT has no imaginary component.

Figure 3-12 and Figure 3-13 illustrate the Hanning window and one cycle of a sine pattern for a sample size of 8. You can see that the DFT-even Hanning window is not symmetric about its midpoint and its last point is not equal to its first point, much like one complete cycle of a sine pattern.

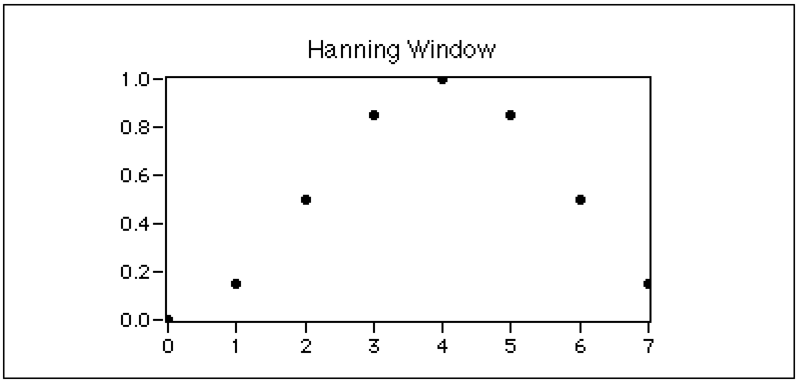


Figure 3-12. Hanning Window for Sample Size 8

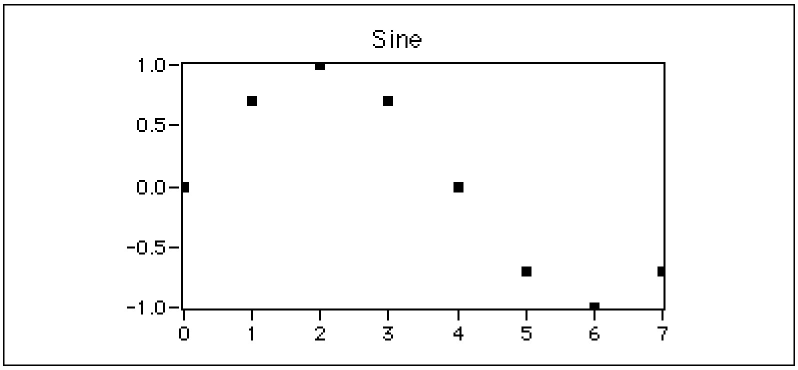


Figure 3-13. Sine Pattern for Sample Size 8

Finally, the DFT considers input sequences to be periodic—that the signal being analyzed is actually a concatenation of the input signal. Figure 3-14 shows three such cycles of the previous sequences, demonstrating the smooth periodic extension of the DFT-even window and the single-cycle sine pattern.

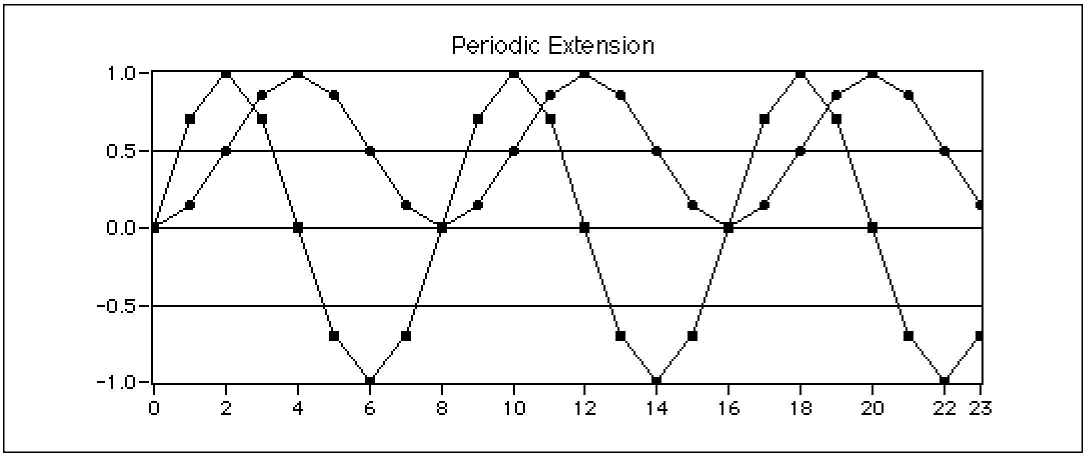


Figure 3-14. Periodic Extension

Another type of window application is that of FIR filter design. This application requires windows that are symmetric about their midpoint. Refer to Part III, *Measurement Analysis in LabVIEW*, of the *LabVIEW Measurements Manual* for more information about filtering.

The following equations of the Hanning window function illustrate the difference between the DFT-even window function (spectral analysis) and the symmetrical window function (coefficient design).

Hanning window function for spectral analysis:

$$w[i] = 0.5\left(1 - \cos\left(\frac{2\pi i}{N}\right)\right)$$

for

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

Hanning window function for symmetrical coefficient design:

$$w[i] = 0.5\left(1 - \cos\left(\frac{2\pi i}{N-1}\right)\right)$$

for

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

The two equations above show that you can implement the symmetrical window functions by slightly modifying the use of the DFT-even window functions.

What Type of Window Do I Use?

Now that you have seen several of the many different types of windows that are available, you may ask, "What type of window should I use?"
The answer depends on the type of signal you have and what you are looking for. Choosing the correct window requires some prior knowledge of the signal that you are analyzing. In summary, Table 3-1 shows the different types of signals and the appropriate windows that you can use with them.

Table 3-1. 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
Order tracking	Rectangular
System 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

In many cases, you may not have sufficient prior knowledge of the signal, so you need to experiment with different windows to find the best one.

4 Curve Fitting

This chapter describes how to extract information from a data set to obtain a functional description. Refer to the examples in `examples\analysis\regressn.llb` for examples of how to use the regression VIs, available on the **Functions»Mathematics»Curve Fitting** palette.

Introduction to Curve Fitting

Curve fitting analysis is a technique for extracting a set of curve parameters or coefficients from the data set to obtain a functional description of the data set. The algorithm that fits a curve to a particular data set is known as the Least Squares Method and is discussed in most introductory textbooks in probability and statistics. The error is defined as

$$e(a) = [f(x,a) - y(x)]^2 \tag{4-1}$$

where $e(a)$ is the 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 which best describes the curve.

For example, let $a = \{a_0, a_1\}$. Then the functional description of a line is

$$f(x,a) = a_0 + a_1 x$$

The least squares algorithm finds a by solving the system

$$\frac{\partial}{\partial a} e(a) = 0 \tag{4-2}$$

To solve this system, you set up and solve the Jacobian system generated by expanding Equation 4-2. After you solve the system for a, you can obtain an estimate of the observed data set for any value of x using the functional description $f(x, a)$.

In LabVIEW, the curve fitting VIs automatically set up and solve the Jacobian system and return the set of coefficients that best describes your data set. You can concentrate on the functional description of your data and not worry about solving the system in Equation 4-2.

Two input sequences, *Y* Values and *X* Values, represent the data set $y(x)$. A sample or point in the data set is

$$(x_i, y_i)$$

where x_i is the i^{th} element of the sequence *X* Values, and y_i is the i^{th} element of the sequence *Y* Values.

In general, for each predefined type of curve fit, there are two types of VIs, unless otherwise specified. One type returns only the coefficients, so that you can further manipulate the data. The other type returns the coefficients, the corresponding expected or fitted curve, and the mean squared error (MSE). Because it is a discrete system, the VI calculates the MSE, which is a relative measure of the residuals between the expected curve values and the actual observed values, using the formula

$$\text{MSE} = \frac{1}{n}\sum_{i=0}^{n-1}(f_i - y_i)^2 \tag{4-3}$$

where *f* is the sequence representing the fitted values, *y* is the sequence representing the observed values, and *n* is the number of sample points observed.

The **Functions»Mathematics»Curve Fitting** palette offers both linear and nonlinear curve fitting algorithms. The different types of curve fitting in LabVIEW are outlined below:

- *Linear Fit*—fits experimental data to a straight line of the form $y = mx + c$

$$y[i]=a_0+a_1x[i]$$

- *Exponential Fit*—fits data to an exponential curve of the form $y = a^b$

$$y[i] = a_0^{a_1 x[i]}$$

- *General Polynomial Fit*—fits data to a polynomial function of the form

$$y = a + bx + cx\text{^}2 + \ldots$$
$$y[i] = a_0+a_1x[i]+a_2x[i]^2\ldots$$

- *General Linear Fit*—fits data to

$$y[i] = a_0+a_1f_1(x[i])+a_2f_2(x[i]) + \ldots$$

where $y[i]$ is a linear combination of the parameters a_0, a_1, a_2.... The general linear fit also features selectable algorithms for better precision and accuracy. For example, $y = a_0 + a_1\sin(x)$ is a linear fit because y has a linear relationship with parameters a_0 and a_1. Polynomial fits are always linear fits for the same reason. But special algorithms can be designed for the polynomial fit to speed up the fitting processing and improve accuracy.

- *Nonlinear Levenberg-Marquardt Fit*—fits data to

$$y[i] = f(x[i], a_0, a_1, a_2 \ldots)$$

where a_0, a_1, a_2... are the parameters. This method is the most general method and does not require y to have a linear relationship with a_0, a_1, a_2.... It can be used to fit linear or nonlinear curves, but is almost always used to fit a nonlinear curve, because the general linear fit method is better suited to linear curve fitting. The Levenberg-Marquardt method does not always guarantee a correct result, so it is absolutely necessary to verify the results.

Applications of Curve Fitting

The practical applications of curve fitting are numerous. Some of them are listed below.

- Removal of measurement noise.
- Filling in missing data points (for example, if one or more measurements were missed or improperly recorded).
- Interpolation (estimation of data between data points; for example, if the time between measurements is not small enough).
- Extrapolation (estimation of data beyond data points; for example, if you are looking for data values before or after the measurements were taken).
- Differentiation of digital data. (For example, if you need to find the derivative of the data points. The discrete data can be modeled by a polynomial, and the resulting polynomial equation can be differentiated.)
- Integration of digital data (for example, to find the area under a curve when you have only the discrete points of the curve).
- To obtain the trajectory of an object based on discrete measurements of its velocity (first derivative) or acceleration (second derivative).

General LS Linear Fit Theory

The General LS Linear Fit Problem can be described as follows.

Given a set of observation data, find a set of coefficients that fit the linear "model."

$$y_i = b_o x_{i0} + \dots + b_{k-1} x_{ik-1}$$

$$= \sum_{j=0}^{k-1} b_j x_{ij} \quad i=0, 1, \dots, n-1 \qquad (4\text{-}4)$$

where *B* is the set of **Coefficients**, *n* is the number of elements in **Y Values** and the number of rows of **H**, and *k* is the number of **Coefficients**.

x_{ij} is your observation data, which is contained in **H**.

$$H = \begin{bmatrix} x_{00} & x_{01} \cdots & x_{0k-1} \\ x_{10} & x_{11} \cdots & x_{1k-1} \\ . & & \\ . & & \\ . & & \\ . & & \\ x_{n-10} & x_{n-12} \cdots & x_{n-1k-1} \end{bmatrix}$$

Equation 4-4 can also be written as *Y* = *HB*.

This is a multiple linear regression model, which uses several variables $x_{i0}, x_{i1}, \dots, x_{ik-1}$, to predict one variable y_i. In contrast, the Linear Fit, Exponential Fit, and Polynomial Fit VIs are all based on a single predictor variable, which uses one variable to predict another variable.

In most cases, we have more observation data than coefficients. The equations in 4-4 may not have the solution. The fit problem becomes to find the coefficient *B* that minimizes the difference between the observed data, y_i and the predicted value:

$$z_i = \sum_{j=0}^{k-1} b_j x_{ij}$$

This VI uses the least chi-square plane method to obtain the coefficients in 4-4, that is, finding the solution, *B*, which minimizes the quantity:

$$\chi^2 = \sum_{i=0}^{n-1}\left(\frac{y_i - z_i}{\sigma_i}\right)^2 = \sum_{i=0}^{n-1}\left(\frac{y_i - \sum_{j=0}^{k-1} b_j x_{ij}}{\sigma_i}\right)^2 = |H_0B - Y_0|^2 \qquad (4\text{-}5)$$

where

$$h_{oij} = \frac{x_{ij}}{\sigma_i},\ y_{oi} = \frac{y_i}{\sigma_i},\ i=0, 1, \dots, n-1; j=0, 1, \dots, k-1$$

In this equation, σ_i is the **Standard Deviation**. If the measurement errors are independent and normally distributed with constant standard deviation $\sigma_i = \sigma$, the preceding equation is also the least square estimation.

There are different ways to minimize χ^2. One way to minimize χ^2 is to set the partial derivatives of χ^2 to zero with respect to $b_0, b_1, \dots, b_{k-1}$.

$$\begin{cases} \dfrac{\partial\chi^2}{\partial b_0} = 0 \\ \dfrac{\partial\chi^2}{\partial b_1} = 0 \\ . \\ . \\ . \\ . \\ \dfrac{\partial\chi^2}{\partial b_{k-1}} = 0 \end{cases}$$

The preceding equations can be derived to:

$$H_0^T H_0 B = H_0^T Y \qquad (4\text{-}6)$$

Where H_0^T is the transpose of $\boldsymbol{H}_0$.

The equations in 4-6 are also called normal equations of the least-square problems. You can solve them using LU or Cholesky factorization algorithms, but the solution from the normal equations is susceptible to roundoff error.

An alternative, and preferred way to minimize χ^2 is to find the least-square solution of equations

$$H_0B=Y_0$$

You can use QR or SVD factorization to find the solution, *B*. For QR factorization, you can choose Householder, Givens, and Givens2 (also called fast Givens).

Different algorithms can give you different precision, and in some cases, if one algorithm cannot solve the equation, perhaps another algorithm can. You can try different algorithms to find the best one based on your observation data.

The **Covariance** matrix *C* is computed as

$$C= (H_0^TH_0)^{-1}$$

The **Best Fit** *Z* is given by

$$z_i = \sum_{j=0}^{k-1} b_jx_{ij}$$

The mse is obtained using the following formula:

$$\boldsymbol{mse} = \frac{1}{n} \sum_{i=0}^{n-1}\left(\frac{y_i - z_i}{\sigma_i}\right)^2$$

The polynomial fit that has a single predictor variable can be thought of as a special case of multiple regression. If the observation data sets are $\{x_i, y_i\}$ where i = 0, 1, …, n–1, the model for polynomial fit is

$$y_i = \sum_{j=0}^{k-1} b_j x_i^j = b_0 + b_1 x_i + b_2 x_i^2 + \ldots + b_{k-1} x_i^{k-1} \tag{4-7}$$

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

Comparing equations 4-4 and 4-7 shows that $x_{ij} = x_i^j$. In other words,

$$\begin{aligned} x_{i0} &= x_i^0 \\ &= 1 \end{aligned}, \; x_{i1} = x_i, \;\; x_{i2} = x_i^2, \ldots \;\; x_{ik-1} = x_i^{k-1}$$

In this case, you can build ***H*** as follows:

$$H = \begin{Bmatrix} 1 & x_0 & x_0^2 & \ldots & x_0^{k-1} \\ 1 & x_1 & x_1^2 & \ldots & x_1^{k-1} \\ . \\ . \\ . \\ . \\ 1 & x_{n-1} & x_{n-1}^2 & \ldots & x_{n-1}^{k-1} \end{Bmatrix}$$

Instead of using $x_{ij} = x_j^i$, you can also 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 follows:

$$H = \begin{Bmatrix} f_0(x_0) & f_1(x_0) & f_2(x_0) & \dots & f_{k-1}(x_0) \\ f_0(x_1) & f_1(x_1) & f_2(x_1) & \dots & f_{k-1}(x_1) \\ . & & & & \\ . & & & & \\ . & & & & \\ . & & & & \\ f_0(x_{n-1}) & f_1(x_{n-1}) & f_2(x_{n-1}) & \dots & f_{k-1}(x_{n-1}) \end{Bmatrix}$$

Your fit model is:

$$y_i = b_0 f_0(x) + b_1 f_1(x) + \dots + b_{k-1} f_{k-1}(x)$$

How to Use the General LS Linear Fit VI

The Linear Fit VI calculates the coefficients a_0 and a_1 that best fits the experimental data ($x[i]$ and $y[i]$) to a straight line model given by

$$y[i] = a_0 + a_1 x[i]$$

Here, $y[i]$ is a linear combination of the coefficients a_0 and a_1. You can extend this concept further so that the multiplier for a_1 is some function of x. For example:

$$y[i] = a_0 + a_1 \sin(\omega x[i])$$

or

$$y[i] = a_0 + a_1 (x[i])^2$$

or

$$y[i] = a_0 + a_1 \cos(\omega x[i]^2)$$

where ω is the angular frequency. In each of these cases, $y[i]$ is a linear combination of the coefficients a_0 and a_1. This is the basic idea behind the General LS Linear Fit VI, where the $y[i]$ can be linear combinations of several coefficients, each of which may be multiplied by some function of the $x[i]$. Therefore, you can use it to calculate coefficients of the functional models that can be represented as linear combinations of the coefficients, such as

$$y = a_0 + a_1\sin(\omega x)$$

or

$$y = a_0 + a_1x^2 + a_2\cos(\omega x^2)$$

$$y = a_0 + a_1(3\sin(\omega x)) + a_2x^3 + \frac{a_3}{x} + \ldots$$

In each case, note that y is a *linear* function of the coefficients (although it may be a nonlinear function of x).

You will now see how to use the General LS Linear Fit VI to find the best linear fit to a set of data points. The inputs and outputs of the General LS Linear Fit VI are shown in Figure 4-1.

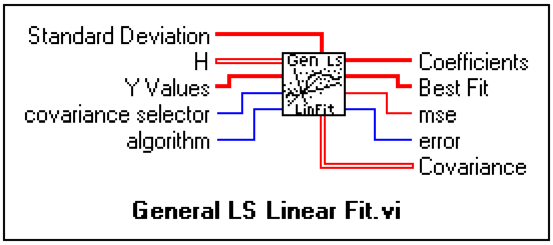


Figure 4-1. General LS Linear Fit VI

The data that you collect ($x[i]$ and $y[i]$) is to be given to the inputs **H** and **Y Values**. The **Covariance** output is the matrix of covariances between the coefficients a_k, where c_{ij} is the covariance between a_i and a_j, and c_{kk} is the variance of a_k. At this stage, you need not be concerned about the inputs **Standard Deviation**, **covariance selector,** and **algorithm**. For now, you will just use their default values. Refer to LabVIEW Help, available by selecting **Help»Contents and Index**, for more information about these inputs.

The matrix ***H*** is known as the *Observation Matrix* and will be explained in more detail later. **Y Values** is the set of observed data points $y[i]$. For example, suppose you have collected samples (**Y Values**) from a transducer and you want to solve for the coefficients of the model:

$$y = a_o + a_1\sin(\omega x) + a_2\cos(\omega x) + a_3x^2$$

You see that the multiplier for each a_j is a different function. For example, a_0 is multiplied by 1, a_1 is multiplied by sin(ωx), a_2 is multiplied by cos(ωx), and so on. To build ***H***, you set each column of ***H*** to the independent functions evaluated at each *x* value, *x*[*i*]. Assuming there are 100 *x* values, ***H*** would be:

$$H = \begin{bmatrix} 1 & \sin(\omega x_0) & \cos(\omega x_0) & x_0^2 \\ 1 & \sin(\omega x_1) & \cos(\omega x_1) & x_1^2 \\ 1 & \sin(\omega x_2) & \cos(\omega x_2) & x_2^2 \\ \dots & \dots & \dots & \dots \\ 1 & \sin(\omega x_{99}) & \cos(\omega x_{99}) & x_{99}^2 \end{bmatrix}$$

If you have *N* data points and *k* coefficients (a_0, a_1, ...a_{k-1}) for which to solve, ***H*** will be an *N*-by-*k* matrix with *N* rows and *k* columns. Thus, the number of rows of ***H*** is equal to the number of elements in **Y Values**, whereas the number of columns of ***H*** is equal to the number of coefficients for which you are trying to solve.

In practice, ***H*** is not available and must be built. Given that you have the *N* independent **X Values** and observed **Y Values**, use the General LS Linear Fit VI to build ***H***.

Nonlinear Lev-Mar Fit Theory

The Nonlinear Lev-Mar Fit VI, available on the **Functions»Mathematics» Curve Fitting** palette, determines the set of coefficients that minimize the chi-square quantity:

$$\chi^2 = \sum_{i=0}^{N-1}\left(\frac{y_i - f(x_i;a_1\dots a_M)}{\sigma_i}\right)^2 \tag{4-8}$$

In this equation, (x_i, y_i) are the input data points, and $f(x_i;a_1...a_M) = f(X, A)$ is the nonlinear function where $a_1...a_M$ are coefficients. If the measurement errors are independent and normally distributed with constant, standard deviation $\sigma_i = \sigma$, this is also the least-square estimation.

You must specify the nonlinear function $f = f(X, A)$ in the Formula Node on the block diagram of the Target Fnc & Deriv NonLin VI, which is a subVI of the Nonlinear Lev-Mar Fit VI.

This VI provides two ways to calculate the Jacobian (partial derivatives with respect to the coefficients) needed in the algorithm. These two methods follow:

- Numerical calculation—Uses a numerical approximation to compute the Jacobian.
- Formula calculation—Uses a formula to compute the Jacobian. You need to specify the Jacobian function $\partial f / \partial A$ in the Formula Node on the block diagram of the Target Fnc & Deriv NonLin VI, as well as the nonlinear function $f = f(X, A)$. This is a more efficient computation than the numerical calculation, because it does not require a numerical approximation to the Jacobian.

The input arrays **X** and **Y** define the set of input data points. The VI assumes that you have prior knowledge of the nonlinear relationship between the x and y coordinates. That is, $f = f(X, A)$, where the set of coefficients, A, is determined by the Levenberg-Marquardt algorithm.

Using this function successfully sometimes depends on how close your initial guess coefficients are to the solution. Therefore, it is always worth taking effort and time to obtain good initial guess coefficients to the solution from any available resources before using the function.

Using the Nonlinear Lev-Mar Fit VI

So far, you have seen VIs that are used when there is a linear relationship between y and the coefficients $a_0, a_1, a_2, \ldots$. However, when a nonlinear relationship exists, you can use the Nonlinear Lev-Mar Fit VI to determine the coefficients. This VI uses the Levenberg-Marquardt method, which is very robust, to find the coefficients $\boldsymbol{A} = \{a_0, a_1, a_2, \ldots, a_k\}$ of the nonlinear relationship between $\boldsymbol{A}$ and $y[i]$. The VI assumes that you have prior knowledge of the nonlinear relationship between the x and y coordinates.

As a preliminary step, you need to specify the nonlinear function in the Formula Node on the block diagram of one of the subVIs of the Nonlinear Lev-Mar Fit VI. This particular subVI is the Target Fnc and Deriv NonLin VI.

When using the Nonlinear Lev-Mar Fit VI, you also must specify the nonlinear function in the Formula Node on the block diagram of the Target Fnc and Deriv NonLin VI.

The connections to the Nonlinear Lev-Mar Fit VI are shown below:

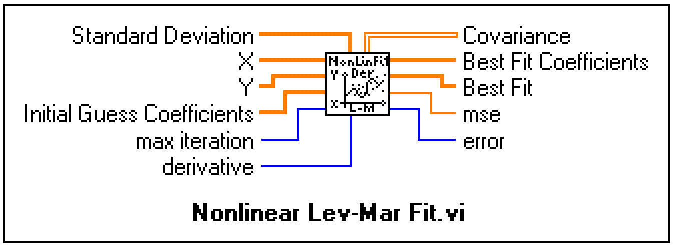


Figure 4-2. Nonlinear Lev-Mar Fit VI

X and **Y** are the input data points $x[i]$ and $y[i]$.

Initial Guess Coefficients is your initial guess as to what the coefficient values are. The coefficients are those used in the formula that you entered in the **Formula Node** of the Target Fnc and Deriv NonLin VI. Using the Nonlinear Lev-Mar Fit VI successfully sometimes depends on how close your initial guess coefficients are to the actual solution. Therefore, it is always worth taking the time and effort to obtain a good initial guess to the solution from any available resource.

For now, you can leave the other inputs to their default values. Refer to LabVIEW Help, available by selecting **Help»Contents and Index**, for more information about these inputs.

Best Fit Coefficients: the values of the coefficients (a_0, a_1, ...) that best fit the model of the experimental data.

5

Linear Algebra

This chapter explains how to use the linear algebra VIs to perform matrix computation and analysis. Refer to the examples in `examples\analysis\linxmpl.llb` for examples of how to use the linear algebra VIs, available on the **Functions»Mathematics»Linear Algebra** palette.

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 may occur naturally or may be 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 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 represented by a 2D array of elements. These elements may 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 $(i,j)^{th}$ element located in the i^{th} row and the j^{th} column. In general, such a matrix is called a *rectangular matrix*. When $m = n$, so that the number of rows is equal to the number of columns, it is called a *square matrix*. An $m \times 1$ matrix (m rows and one column) is called a *column vector*. A *row vector* is a $1 \times n$ matrix (1 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 called 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 called an *identity matrix*, also known as *unit matrix*. If all the elements below the main diagonal are zero, then the matrix is known as an *upper triangular matrix*. On the other hand, if all the elements above the main diagonal are zero, then the matrix is known as a *lower triangular matrix*. When all the elements are real numbers, the matrix is referred to as a *real matrix*. On the other hand, when at least one of the elements of the matrix is a complex number, the matrix is referred to as 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 x 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

$$|A| = \left|\begin{bmatrix} 2 & 5 & 3 \\ 6 & 1 & 7 \\ 1 & 6 & 9 \end{bmatrix}\right| = \left(2\left|\begin{bmatrix} 1 & 7 \\ 6 & 9 \end{bmatrix}\right| - 5\left|\begin{bmatrix} 6 & 7 \\ 1 & 9 \end{bmatrix}\right| + 3\left|\begin{bmatrix} 6 & 1 \\ 1 & 6 \end{bmatrix}\right|\right) =$$

$$2(-33) - 5(47) + 3(35) = -196$$

The determinant tells many important properties of the matrix. For example, if the determinant of the matrix is zero, then 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 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 $\mathbf{A}$ defined above,

$$B = A^T = \begin{bmatrix} 2 & 6 & 1 \\ 5 & 1 & 6 \\ 3 & 7 & 9 \end{bmatrix}$$

In 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 called 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 $C = C^H$, then C is called a *Hermitian matrix*.

Linear Independence

A set of vectors $x_1, x_2, ..., x_n$ is said to be *linearly dependent* if and only if there exist scalars $\alpha_1, \alpha_2, ..., \alpha_n$, not all zero, such that

$$\alpha_1 x_1 + \alpha_2 x_2 + \ldots + \alpha_n x_n = 0$$

In simpler terms, if one of the vectors can be written in terms of a linear combination of the others, then the vectors are said to be linearly dependent.

If the only set of α_i for which the above equation holds is $\alpha_1 = 0$, $\alpha_2 = 0$, ..., $\alpha_n = 0$, then the set of vectors x_1, x_2, ..., x_n is said to be *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, the above equation always holds for $\alpha_1 = 0$, $\alpha_2 = 0$, ..., $\alpha_n = 0$. Therefore, to show the linear independence of the set, you must show that $\alpha_1 = 0$, $\alpha_2 = 0$, ..., $\alpha_n = 0$ is the only set of α_i for which the above equation holds.

For example, first consider the vectors

$$x = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \qquad y = \begin{bmatrix} 3 \\ 4 \end{bmatrix}$$

Notice that $\alpha_1 = 0$ and $\alpha_2 = 0$ are the only values, for which the relation $\alpha_1 x + \alpha_2 y = 0$ holds true. Hence, these two vectors are linearly independent of each other. Let us now look at vectors

$$x = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \qquad y = \begin{bmatrix} 2 \\ 4 \end{bmatrix}$$

Notice that, if $\alpha_1 = -2$ and $\alpha_2 = 1$, then $\alpha_1 x + \alpha_2 y = 0$. Therefore, these two vectors are linearly dependent on each other. You must completely understand this definition of linear independence of vectors to fully appreciate the concept of the rank of the matrix as discussed next.

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 will 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. Remember from the earlier discussion that 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

$$\left| \begin{bmatrix} 1 & 2 & 3 \\ 0 & 1 & -1 \\ 1 & 0 & 1 \end{bmatrix} \right| = -1$$

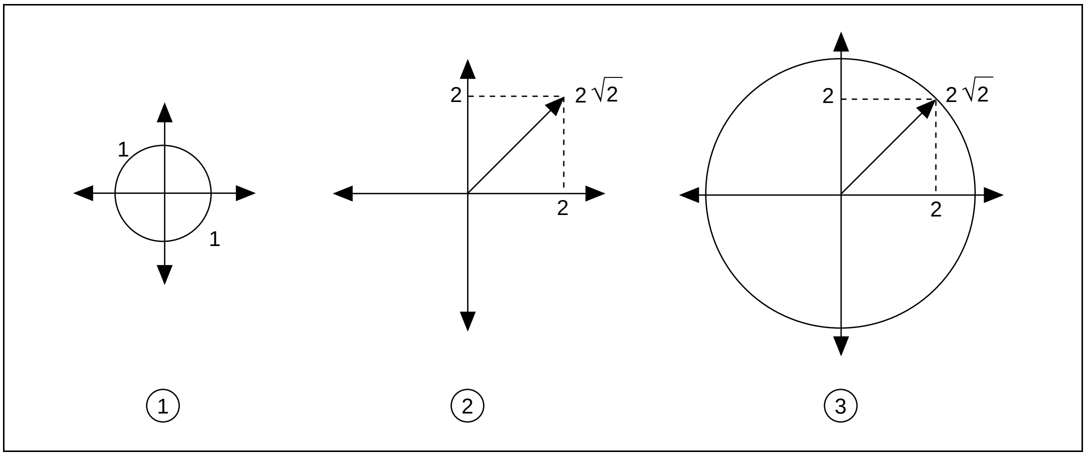
Hence, the rank of *B* is 3. A square matrix has full rank if and only if its determinant is different from zero. Matrix *B* is not a full-rank matrix.

"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 $x = \begin{bmatrix} x1 & x2 \end{bmatrix}$ and $y = \begin{bmatrix} y1 & y2 \end{bmatrix}$, 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. This is shown in Figure 5-1.



1 unit ball of radius = 1 unit	2 vector of length $\sqrt{2^2 + 2^2} = \sqrt{8} = 2\sqrt{2}$	3 unit ball expanded by a factor of $2\sqrt{2}$

Figure 5-1. Euclidean Norm of a Vector

Figure 1a shows a unit ball of radius = 1 unit. Figure 1b shows a vector of length $\sqrt{2^2 + 2^2} = \sqrt{8} = 2\sqrt{2}$. As shown in Figure 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 may become an ellipsoidal (ellipse in 3-D), 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 called the 1-norm. In mathematical terms, the 1-norm is simply the maximum absolute column sum of the matrix.

$$\|\boldsymbol{A}\|_1 = max_j \sum_{i=0}^{n-1} |a_{i,j}|$$

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

$$\|\boldsymbol{A}\|_\infty = max_i \sum_{j=0}^{n-1} |a_{i,j}|$$

In this case, you add the magnitudes of all elements in each row of the matrix. The maximum value that you get is called the inf-norm. For the above example matrix,

$$\|\boldsymbol{A}\|_\infty = max(4, 6) = 6$$

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 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

$$cond(\boldsymbol{A}) = \|\boldsymbol{A}\|_p \cdot \|A^{-1}\|_p$$

where p can be one of the four norm types discussed above. 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 mentioned earlier, 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 47168, and hence the matrix is close to being singular. As you might recall, 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.

In this section, you have become familiar with some basic notation and fundamental matrix concepts such as determinant of a matrix and its rank.

Basic Matrix Operations and Eigenvalues-Eigenvector Problems

In this section, consider some very basic matrix operations. Two matrices, *A* and *B*, are said to be equal if they have the same number of rows and columns and their corresponding elements are all 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 if and only if they have the same number of rows and columns. If both matrices *A* and *B* have *m* rows and *n* columns, then their sum *C* is an *m*-by-*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, then their product *C* is an *m*-by-*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 5-2.

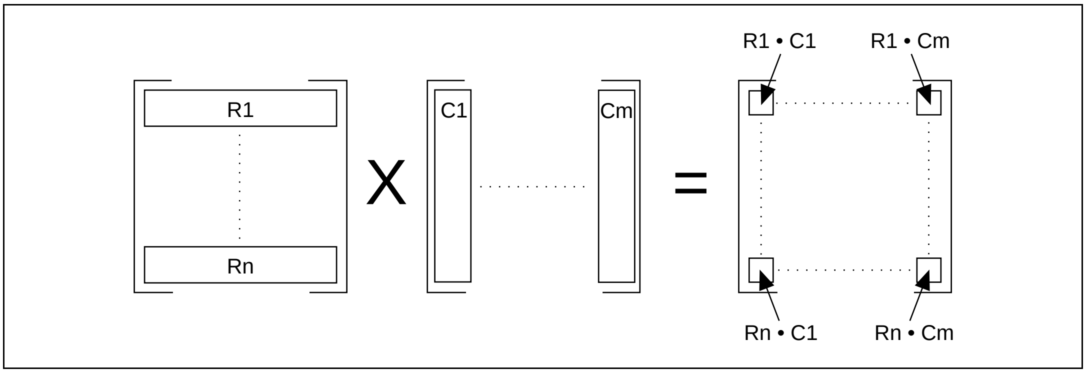


Figure 5-2. Matrix Multiplication

Matrix multiplication, in general, is not commutative, that is, $AB \neq BA$. Also, remember that 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, then 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*. Note that 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, illustrated in Figure 5-3.

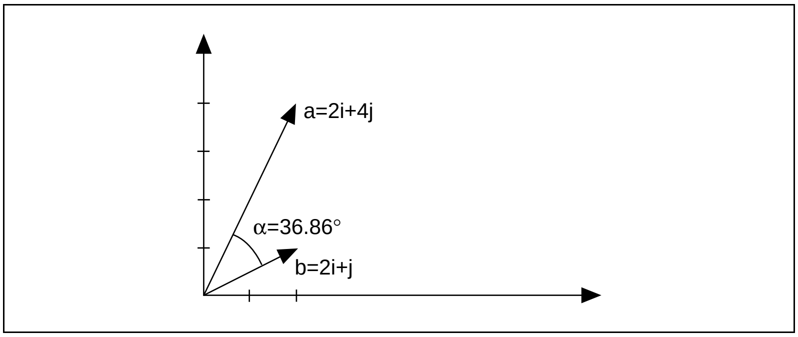


Figure 5-3. Vectors *a* and *b*

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 5-4. 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,

$$W = |a||d|\cos\alpha = a \bullet d$$

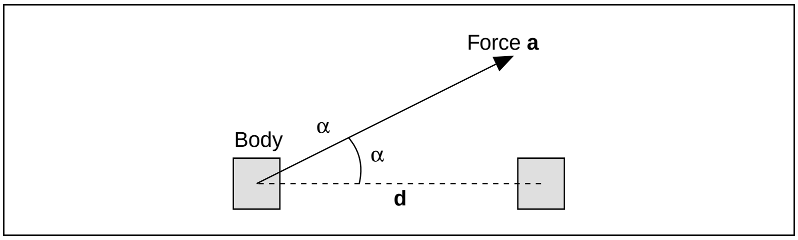


Figure 5-4. Force Vector

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

$$Ax = \lambda x$$

Such a scalar λ is called an *eigenvalue*, and *x* is a corresponding *eigenvector*.

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 the above equation 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 is also 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.
- As discussed earlier, 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 are also 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 is shown in Figure 5-5. The **Input Matrix** is an *N*-by-*N* real square matrix. **Matrix type** determines the type of the input matrix. **Matrix type** 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.

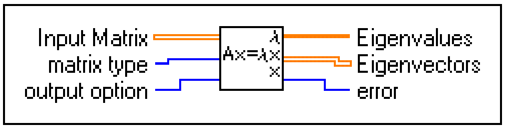


Figure 5-5. EigenValues and Vectors VI

Output option determines what needs to be computed. Output option = 0 indicates that only the eigenvalues need to be computed. Output option = 1 indicates that both the eigenvalues and the eigenvectors should be computed. It is computationally very expensive to compute both the eigenvalues and the eigenvectors. So, it is important that you use the output option control in the EigenValues and Vectors VI very 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 an nonsymmetric matrix. So, choose the matrix type control 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 if and 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. You can, however, compute the *pseudoinverse* of a rectangular matrix. Refer to the *Matrix Factorization* section later in 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 a $n \times n$ matrix and *b* is a given *n*-vector. The aim is to determine *x*, the unknown solution *n*-vector. There are two important questions to be asked about the existence of such a solution. Does such a solution exist, and if it does is it unique? The answer to both of these questions lies in determining the singularity or nonsingularity of the matrix *A*.

As discussed earlier, a matrix is said to be 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, then the number of solutions is determined by the right-hand-side vector *b*. If *A* is singular and $Ax = b$, then $A(x + \Upsilon z) = b$ for any scalar Υ, where the vector *z* is as in the last definition above. Thus, if a singular system has a solution, then the solution cannot be unique.

It is not a good idea to explicitly compute the inverse of a matrix, because such a computation is prone to numerical inaccuracies. Therefore, it is not a good strategy to 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 three basic steps involved in the Gaussian Elimination technique are as follows. 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 known as LU factorization. Given this, the linear system $Ax = b$ can be expressed as $LUx = b$. Such a system can then 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 easily determined 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 easily determined and then used to determine the other elements sequentially. Hence, the name back-substitution. So far, this chapter has discussed 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.

The **Functions»Mathematics»Linear Algebra** palette provides VIs for computing the inverse of a matrix, computing LU decomposition of a matrix, and solving 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 if and only if it is symmetric and 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*), then the VI finds the *least square* solution *x*. The least square solution is the one which minimizes the norm of $Ax - b$. The same holds true also for nonsquare matrices.

Matrix Factorization

The previous section discussed 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. You looked at one such technique, the *LU decomposition* technique, in which you factorized 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, then an LU factorization can be computed such that $A = U^T U$, where U is an upper triangular matrix. This is called *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. It is easy to 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. The following section discusses this last application.

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 can now 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

$$A^{\dagger} = VS^{\dagger}U^{T}$$

Note that the pseudoinverse exists regardless of whether the matrix is square or rectangular. If *A* is square and nonsingular, then the pseudoinverse is the same as the usual matrix inverse. The **Functions»Mathematics»Linear Algebra** palette includes a VI for computing the pseudoinverse of real and complex matrices.

Summary

- A matrix can be considered as a two-dimensional array of *m* rows and *n* columns. Determinant, rank, and condition number are some important attributes of a matrix.
- The condition number of a matrix affects the accuracy of the final solution.
- The determinant of a diagonal matrix, an upper triangular matrix, or a lower triangular matrix is the product of its diagonal elements.
- Two matrices can be multiplied only if the number of columns of the first matrix is equal to the number of rows in the second matrix.
- An eigenvector of a matrix is a nonzero vector that does not rotate when the matrix is applied to it. Similar matrices have the same eigenvalues.
- The existence of a unique solution for a system of equations depends on whether the matrix is singular or nonsingular.

6

Probability and Statistics

This chapter explains some fundamental concepts on probability and statistics and shows how to use these concepts in solving real-world problems. Refer to the examples in `examples\analysis\statxmpl.llb` for examples of how to use the probability and statistics VIs, available on the **Functions»Mathematics»Probability and Statistics** palette.

Probability and Statistics

Facts and figures form an important part of life. Statements such as "There is a 60% chance of thunderstorms," "Joe was ranked among the top five in the class," "Michael Jordan has an average of 30 points a game this season," and so on are common. These statements give a lot of information, but we seldom think how this information was obtained. Was there a lot of data involved in obtaining this information? If there was, how did someone condense it to single numbers such as *60% chance* and *average of 30 points* or terms such as *top five*. The answer to all these questions brings up the very interesting field of statistics.

First, consider how information (data) is generated. Consider the 1997 basketball season. Michael Jordan of the Chicago Bulls played 51 games, scoring a total of 1568 points. This includes the 45 points he posted, including the game-winning buzzer three-pointer, in a 103-100 victory over the Charlotte Hornets; his 36 points in an 88-84 victory over the Portland Trail Blazers; a season high of 51 points in an 88-87 victory over the New York Nicks; 45 points, 7 rebounds, 5 assists and 3 steals in a 102-97 victory over the Cleveland Cavaliers; and his 40 points, 6 rebounds, and 6 assists in a 107-104 victory over the Milwaukee Bucks. The point is not that Jordan is a great player, but that a single player can generate lots of data in a single season. The question is how to condense all this data so that it brings out all the essential information and is yet easy to remember. This is where the term *statistics* comes into the picture.

To condense all the data, single numbers must make it more intelligible and help draw useful inferences. For example, consider the number of points that Jordan scored in different games. It is difficult to remember how many points he scored in each game. But if you divide the total number of points that Jordan scored (1568) by the number of games he has played (51), you have a single number of 30.7 and can call it points per game *average*.

Suppose you want to rate Jordan’s free throw shooting skills. It might be difficult to do so by looking at his performance in each game. However, you can divide the number of free throws he has scored in all the games by the total number of free throws he was awarded. This shows he has a free throw *percentage* of 84.4%. You can obtain this number for all the NBA players and then rank them. Thus, you can condense the information for all the players into single numbers representing free throw percentage, points per game, and three-point average. Based on this information, you can rank players in different categories. You can further weight these different numbers and come up with a single number for each player. These single numbers can then help us in judging the Most Valuable Player (MVP) for the season. Thus, in a broad sense, the term statistics implies different ways to summarize data to derive useful and important information from it.

The next question is, what is probability? You have looked at ways to summarize lots of data into single numbers. These numbers then help draw conclusions for the present. For example, looking at Jordan’s statistics for the 1996 season helped elect him the MVP for that season. But can you say anything about the future? Can you measure the degree of accuracy in the inference and use it for making future decisions? The answer lies in the theory of probability. Whereas, in laymen’s terms, one would say that it is *probable* that Jordan will continue to be the best in the years to come, you can use different concepts in the field of probability, as discussed later in this chapter, to make more quantitative statements.

In a completely different scenario, there may be certain experiments whose outcomes cannot be predetermined, but certain outcomes may be more probable. This once again leads to the notion of probability. For example, if you flip an unbiased coin in the air, what is the chance that it will land heads up? The chance or probability is 50%. That means, if you repeatedly flip the coin, half the time it will land heads up. Does this mean that 10 tosses will result in exactly five heads? Will 100 tosses result in exactly 50 heads? Probably not. But in the long run, the probability will work out to be 0.5.

To summarize, whereas statistics allows you to summarize data and draw conclusions for the present, probability allows you to measure the degree of accuracy in those conclusions and use them for the future.

Statistics

In this section, you will look at different concepts and terms commonly used in statistics and see how to use the Analysis VIs in different applications.

Mean

Consider a data set *X* consisting of *n* samples $x_0, x_1, x_2, x_3, \ldots, x_{n-1}$. The mean value (a.k.a. average) is denoted by $\bar{x}$ and is defined by the formula

$$\bar{x} = \frac{1}{n}(x_0 + x_1 + x_2 + x_3 + \ldots + x_{n-1})$$

In other words, it is the sum of all the sample values divided by the number of samples. As in the Michael Jordan example, the data set consisted of 51 samples. Each sample was equal to the number of points that Jordan scored in each game. The total of all these points was 1568, divided by the number of samples (51) to get a mean or average value of 30.7.

The input-output connections for the Mean VI are shown below.

Median

Let $S = \{s_0, s_1, s_2, \ldots, s_{n-1}\}$ represent the sorted sequence of the data set *X*. The sequence can be sorted either in the ascending order or in descending order. The median of the sequence is denoted by x_{median} and is obtained by the formula

$$x_{median} = \begin{cases} s_i & \text{n is odd} \\ 0.5(s_{k-1} + s_k) & \text{n is even} \end{cases}$$

where

$$i = \frac{n-1}{2} \text{ and } k = \frac{n}{2}$$

In words, the median of a data sequence is the *midpoint* value in the sorted version of that sequence. For example, consider the sequence $\{5, 4, 3, 2, 1\}$ consisting of five (odd number) samples. This sequence is already sorted in the descending order. In this case, the median is the midpoint value, 3. Consider a different sequence $\{1, 2, 3, 4\}$ consisting of four (even number) samples. This sequence is already sorted in the ascending order. In this case, there are two midpoint values, 2 and 3. As per the formula above, the median is equal to $0.5 \times (2 + 3) = 2.5$. If a student X scored 4.5 points on a test and another student Y scored 1 point on the same test, the median is a very useful quantity for making qualitative statements such as “X lies in the top half of the class” or “Y lies in the bottom half of the class.”

Sample Variance

The sample variance of the data set X consisting of n samples is denoted by s^2 and is defined by the formula

$$s^2 = \frac{1}{n-1}[(x_1 - \bar{x})^2 + (x_2 - \bar{x})^2 + \ldots + (x_n - \bar{x})^2]$$

where $\bar{x}$ denotes the mean of the data set. Hence, the sample variance is equal to the sum of the squares of the deviations of the sample values from the mean divided by *n–1*.

Note The above formula does not apply for n=1. However, it does not mean anything to compute the sample variance if there is only one sample in the data set.

In other words, the sample variance measures the spread or dispersion of the sample values. If the data set consists of the scores of a player from different games, the sample variance can be used as a measure of the consistency of the player. It is always positive, except when all the sample values are equal to each other and in turn equal to the mean.

There is one more type of variance called population variance. The formula to compute population variance is similar to the one above to compute sample variance, except for the (*n–1*) in the denominator replaced by *n*.

The Sample Variance VI computes sample variance, whereas the Variance VI computes the population variance. Statisticians and mathematicians prefer to use the latter, engineers the former. It really does not matter for large values of n, say $n \geq 30$.

Use the proper type of VI suited for your application.

Standard Deviation

The positive square root of the sample variance s^2 is denoted by s and is called the standard deviation of the sample.

Mode

The mode of a sample is a sample value that occurs most frequently in the sample. For example, if the input sequence *X* is

$$X = \{0, 1, 3, 3, 4, 4, 4, 5, 5, 7\}$$

then the mode of *X* is 4, because that is the value that most often occurs in *X*.

Moment About Mean

If *X* represents the input sequence with *n* number of elements in it, and $\bar{x}$ is the mean of this sequence, then the *m*th-order moment can be calculated using the formula

$$\sigma_x^{\ m} = \frac{1}{n}\sum_{i=0}^{n-1}(x_i - \bar{x})^m$$

In other words, the moment about mean is a measure of the deviation of the elements in the sequence from the mean. Note that for $m = 2$, the moment about mean is equal to the population variance.

Histogram

So far, this chapter has discussed different ways to extract important features of a data set. The data is usually stored in a table format, which many people find difficult to grasp. The visual display of data helps us gain insights into the data. Histogram is one such graphical method for displaying data and summarizing key information. Consider a data sequence $X = \{0, 1, 3, 3, 4, 4, 4, 5, 5, 8\}$. Divide the total range of values into 8 intervals. These intervals are 0 – 1, 1 – 2, 2 – 3, ..., 7 – 8. The histogram for the sequence *X* then plots the number of data samples that lie in that interval, not including the upper boundary.

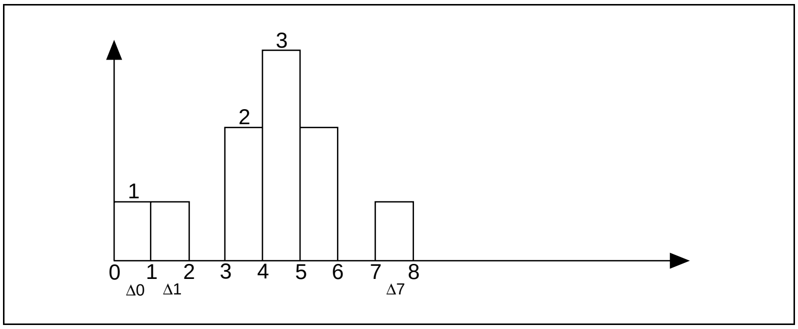


Figure 6-1. Histogram

Figure 6-1 shows that one data sample lies in the range 0 – 1 and 1 – 2, respectively. However, there is no sample in the interval 2 – 3. Similarly, two samples lie in the interval 3 – 4, and three samples lie in the range 4 – 5. Examine the data sequence *X* above and be sure you understand this concept.

There are different ways to compute data for histogram. Next you will see how it is done in the Histogram VI using the sequence *X*.

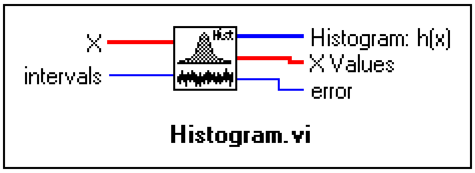


Figure 6-2. Histogram VI

As shown in Figure 6-2, the inputs to this VI are the **input sequence** *X* and the **number of intervals** *m*. The VI obtains **Histogram:h(x)** as follows. It scans *X* to determine the range of values in it. Then the VI establishes the interval width, Δx, according to the specified value of *m*

$$\Delta x = \frac{max - min}{m}$$

where *max* is the maximum value found in *X*, min is the minimum value found in *X*, and *m* is the specified number of intervals.

Let

$$m = 8$$

Then

$$\Delta x = \frac{8-0}{8} = 1$$

Let χ represent the output sequence *X* Values. The histogram is a function of *X*. This VI evaluates the elements of χ using

$$\chi_i = min + 0.5\Delta x + i\Delta x \qquad for \qquad i = 0, 1, 2, \ldots, m-1$$

For this example,

$$\chi_0 = 0.5, \chi_1 = 1.5, \ldots, \chi_7 = 7.5$$

The VI then defines the *i*th interval to be in the range of values from $\chi_i - 0.5\Delta x$ up to but not including $\chi_i + 0.5\Delta x$,

$$\Delta_i = [(\chi_i - 0.5\Delta x), (\chi_i + 0.5\Delta x)], for \qquad i = 0, 1, 2, \ldots, m-1$$

and defines the function $y_i(x) = 1$ for *x* belonging to Δ_i and zero elsewhere. The function has unity value if the value of *x* falls within the specified interval, not including the boundary. Otherwise, it is zero. Notice that the interval is centered about χ_i and its width is Δ_x. If a value is equal to max, it is counted as belonging to the last interval.

For our example,

$$\Delta_0 = [0, 1], \Delta_1 = [1, 2], \ldots, \Delta_7 = [7, 8]$$

and as an example

$$y_0(0) = 1$$

and

$$y_0(1) = y_0(3) = y_0(4) = y_0(5) = y_0(8) = 0 .$$

Finally, the VI evaluates the histogram sequence *H* using

$$h_i = \sum_{j=0}^{n-1} y_i(x_j) \qquad for \qquad i = 0, 1, 2, \ldots, m-1$$

where h_i represents the elements of the output sequence *Histogram: h(X)* and *n* is the number of elements in the input sequence *X*. For this example, $h_0 = 1, h_4 = 3, \ldots, h_7 = 1$.

The **Functions»Mathematics»Probability and Statistics** palette also has a General Histogram VI that is more advanced than the Histogram VI. Refer to *LabVIEW Help*, available by selecting **Help»Contents and Index**, for more information about the probability and statistics VIs.

Mean Square Error (MSE)

If X and Y represent two input sequences, then the mean square error is the average of the sum of the square of the difference between the corresponding elements of the two input sequences. The following formula is used to find the mse.

$$mse = \frac{1}{n}\sum_{i=0}^{n-1}(x_i - y_i)^2$$

where *n* is the number of data points.

Consider a digital signal *x* fed to a system, S_1. The output of this system is y_1. Now you acquire a new system, S_2, which is theoretically known to generate the same result as S_1 but has two times faster response time. Before replacing the old system, you want to be absolutely sure that the output response of both the systems is the same. If the sequences y_1 and y_2 are very large, it is difficult to compare each element in the sequences. In such a scenario, you can use the MSE VI to calculate the mean square error (mse) of the two sequences y_1 and y_2. If the mse is smaller than an acceptable tolerance, then the system S_1 can be reliably replaced by the new system S_2.

Root Mean Square (RMS)

The root mean square Ψ_x of a sequence X is 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 this new squared sequence, and then take the square root of this quantity. The formula used to compute the rms value is

$$\Psi_x = \sqrt{\frac{1}{n}\sum_{i=0}^{n-1}x_i^2}$$

where n is the number of elements in X.

RMS is a widely used quantity in the case of analog signals. For a sine voltage waveform, if V_p is the peak amplitude of the signal, then the root mean square voltage V_{rms} is given by $\frac{V_p}{\sqrt{2}}$.

Probability

In any random experiment, there is always a chance that a particular event will or will not occur. A number between 0 and 1 is assigned to measure this chance, or probability, that a particular event occurs. If you are absolutely sure that the event will occur, its probability is 100% or 1.0, but if you are sure that the event will not occur, its probability is 0.

Consider a simple example. If you roll a single unbiased die, there are six possible events that can occur—either a 1, 2, 3, 4, 5, or 6 can result. What is the probability that a 2 will result? This probability is one in six, or 0.16666. You can define probability in simple terms as the following: the probability that an event A will occur is the ratio of the number of outcomes favorable to A to the total number of equally likely outcomes.

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, number of voters favoring candidate A, and number of accidents at a particular intersection. The values of the numerical outcomes of this experiment can change from experiment to experiment and are called random variables. Random variables can be discrete (if they can take on only a finite number of possible values) or continuous. As an example of the latter, weights of patients coming into a clinic may be anywhere from, say, 80 to 300 pounds. Such random variables can take on any value in an interval of real numbers. Given such a situation, suppose you want to find the probability of encountering a patient weighing exactly 172.39 pounds. You will see how to calculate this probability next using an example.

Consider an experiment to measure the life lengths x of 50 batteries of a certain type. These batteries are selected from a larger population of such batteries. The histogram for observed data is shown below.

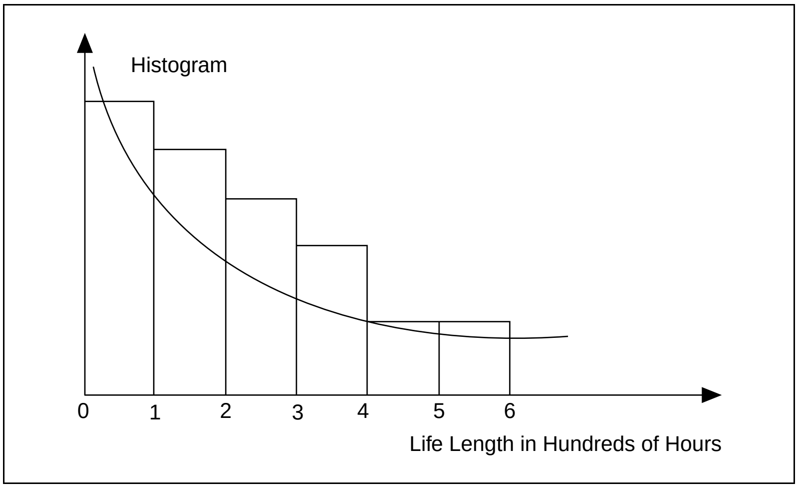


Figure 6-3. Life Lengths Histogram

Figure 6-3 shows that most of the life lengths are between zero and 100 hours, and the histogram values drop off smoothly when you look at larger life lengths.

You can approximate the histogram shown above by an exponentially decaying curve. You could take this function as 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 four hundred hours, this value can be approximated by the area under the curve to the right of the value 4. Such a function that models the histogram of the random variable is called the *probability density function*.

To summarize all the information above in terms of a definition, a random variable X is said to be *continuous* if it can take on the infinite number of possible values associated with intervals of real numbers, and there is a function *f(x)*, called the probability density function, such that

1. $f(x) \geq 0$ for all x
2. $\int_{-\infty}^{\infty} f(x)dx = 1$
3. $P(a \leq X \leq b) = \int_{a}^{b} f(x)dx$

Notice from equation (3) above, that for a specific value of the continuous random variable, that is for

$$X=a, P(X=a) = \int_a^a f(x)dx = 0$$

It should not be surprising that you assign a probability of zero to any specific value, because there are an infinite number of possible values that the random variable can take. Therefore, the chance that it will take on a specific value $X = a$ is extremely small.

The previous example used the exponential function model for the probability density function. There are a number of different choices for this function. One of these is the Normal Distribution, discussed below.

Normal Distribution

The normal distribution is one of the most widely used continuous probability distributions. This distribution function has a symmetric bell shape.

The curve is centered at the mean value $\bar{x} = 0$, and its spread is measured by the variance $s^2 = 1$. These two parameters completely determine the shape and location of the normal density function, whose functional form is given by

$$f(x) = \frac{1}{\sqrt{2\pi}s} e^{-(x-\bar{x})^2/(2s^2)}$$

Suppose a random variable Z has a normal distribution with mean equal to zero and variance equal to one. This random variable is said to have *standard normal distribution*.

The Normal Distribution VI computes the one-sided probability, *p*, of the normally distributed random variable *x*.

$$p = Prob(X \le x)$$

where *X* is a standard normal distribution with the mean value equal to zero and variance equal to one, *p* is the probability and *x* is the value.

Suppose you conduct an experiment in which you measure the heights of adult males. You conduct this experiment on 1000 randomly chosen men and obtain a data set *S*. The histogram distribution has many measurements clumped closely about a mean height, with relatively few very short and very tall males in the population. Therefore, the histogram can be closely approximated by a normal distribution. Now suppose that, among a different set of 1000 randomly chosen males, you want to find the probability that the height of a male is greater than or equal to 170 cms. You can use the Normal Distribution VI to find this probability. Set the input $x = 170$. Thus, the choice of the probability density function is fundamental to obtaining a correct probability value.

The Inverse Normal Distribution VI performs exactly the opposite function. Given a probability *p*, it finds the values *x* that have the chance of lying in a normally distributed sample. For example, you might want to find the heights that have a 60% chance of lying in a randomly chosen data set.

As mentioned earlier, there are different choices for the probability density function. The well-known and widely used ones are the Chi-Square distribution, the F distribution, and the T-distribution. The **Functions»Mathematics»Probability and Statistics** palette has VIs that compute the one-sided probability for these different types of distributions. In addition, it also has VIs that perform the inverse operation.

Summary

- Different concepts in statistics and probability help decipher information and data to make intelligent decisions.
- Mean, Median, Sample Variance, and Mode are some of the statistics techniques to help in making inferences from a sample to a population.
- Histograms are widely used as a simple but informative method of data display.
- Using the theory of probability, you can make inferences from a sample to a population and then measure the degree of accuracy in those inferences.

A

Technical Support Resources

Web Support

National Instruments Web support is your first stop for help in solving installation, configuration, and application problems and questions. Online problem-solving and diagnostic resources include frequently asked questions, knowledge bases, product-specific troubleshooting wizards, manuals, drivers, software updates, and more. Web support is available through the Technical Support section of `www.ni.com`

NI Developer Zone

The NI Developer Zone at `zone.ni.com` is the essential resource for building measurement and automation systems. At the NI Developer Zone, you can easily access the latest example programs, system configurators, tutorials, technical news, as well as a community of developers ready to share their own techniques.

Customer Education

National Instruments provides a number of alternatives to satisfy your training needs, from self-paced tutorials, videos, and interactive CDs to instructor-led hands-on courses at locations around the world. Visit the Customer Education section of `www.ni.com` for online course schedules, syllabi, training centers, and class registration.

System Integration

If you have time constraints, limited in-house technical resources, or other dilemmas, you may prefer to employ consulting or system integration services. You can rely on the expertise available through our worldwide network of Alliance Program members. To find out more about our Alliance system integration solutions, visit the System Integration section of `www.ni.com`

Worldwide Support

National Instruments has offices located around the world to help address your support needs. You can access our branch office Web sites from the Worldwide Offices section of `www.ni.com`. Branch office web sites provide up-to-date contact information, support phone numbers, e-mail addresses, and current events.

If you have searched the technical support resources on our Web site and still cannot find the answers you need, contact your local office or National Instruments corporate. Phone numbers for our worldwide offices are listed at the front of this manual.