Linear Systems in LabVIEW®

J. Kodosky, E. Pérez Ph.D.

Introduction

This application note describes the difference equations for discrete-time, time-invariant linear systems and how to use LabVIEW block diagrams to implement these equations. This application note also discusses the Z transform, signal flow graphs, discrete input and output sequences, and an example that models a simple, linear, time-invariant system.

Signals and Systems

The concepts of signals and systems arise in a wide variety of fields such as telecommunications, aerospace engineering, biomedical engineering, acoustics, chemistry, geophysics, and image processing. A signal is a variation of some form, and a system is any process which transforms a signal (Figure 1).

![Figure 1. Signals and System](image)

The mathematical model of a signal is simply a function of one or more independent variables, for example h(t) and g(x,y), and the mathematical model of a system is a transformation \( T\{\cdot\} \) that operates on a function. The discussion in this application note is limited to one independent variable because the same principles apply to two or more independent variables. The independent variable \( t \) is referred to as time.

The relationship between signals and system is summarized in equation (1) for the case of a single independent variable

\[ y(t) = T\{x(t)\} \]

where \( x(t) \) is the input signal and \( y(t) \) is the system's response to the input signal.
A linear system is a system that obeys the superposition principle. That is, if the input to the system is a linear combination of signals, then the output is the linear combination (superposition) of the responses generated by each signal. The superposition principle is mathematically summarized in equations (2) through (4).

\[
\begin{align*}
    x(t) &= a_1 x_1(t) + a_2 x_2(t) + \ldots + a_n x_n(t) \quad (2) \\
    y(t) &= a_1 y_1(t) + a_2 y_2(t) + \ldots + a_n y_n(t) \quad (3) \\
    y_i(t) &= T[x_i(t)], \quad i = 1, 2, \ldots, n \quad (4)
\end{align*}
\]

where all \( a_i \) are scalar values.

A time-invariant system is a system whose shifted input signal causes a shifted output signal.

\[
y(t - t_0) = T[x(t - t_0)] \quad (5)
\]

where \( t_0 \) is the time shift.

Naturally occurring linear systems do not exist, and strict mathematical modeling of nonlinear systems can be complicated and in many cases unrealistic. However, for most practical applications, you can study nonlinear systems using piecewise linear approximations, and if the region of interest is small, a single linear approximation can model the system.

Depending upon the values taken by the independent variable, signals can be of two types: continuous-time or discrete-time. Continuous-time signals are defined for a continuum of independent variable values. Discrete-time signals are defined only at discrete values of the independent variables. You can sample a continuous-time signal with an analog-to-digital converter to obtain a discrete-time signal. Similarly, you can reconstruct a continuous-time signal from a discrete-time signal using a digital-to-analog converter.

It is now common practice to use microprocessors, floating-point coprocessors, floating-point digital signal processors, and efficient discrete-time, time-invariant linear models to analyze and/or synthesize physical systems. In fact, many analog processing systems are now being replaced by digital counterparts. LabVIEW contains all the elements you need to develop discrete-time, time-invariant systems and models quickly, easily, and efficiently.

**Difference Equations**

A set of linear differential equations describes the behavior of a continuous-time, time-invariant linear system. The solution to the differential equations establishes the system's response to a given input signal. To model a discrete-time, time-invariant linear system using a computer, you must set up the corresponding differential equations as a system of difference equations. The solution to the difference equations establishes the discrete-time system response.
A discrete-time system has a difference equation of the form:

\[ y_k + \sum_{i=1}^{n} b_i y_{k-i} = \sum_{j=0}^{m} a_j x_{k-j} \],

\( k = 0, 1, 2, \ldots \)

where
- \( k \) is the time interval of interest,
- \( y_k \) is the system output at the \( k \)th time interval,
- \( x_k \) is the system input at the \( k \)th time interval,
- \( m \) is the order of the system's forward components,
- \( n \) is the order of the system's feedback components,
- \( a_j \) is the \( j \)th scalar forward coefficient, and
- \( b_i \) is the \( i \)th scalar feedback coefficient.

If all \( b_i \) are zero, then the system has a finite impulse response (FIR). That is, by the \( m \)th time interval after the input goes to zero the output is guaranteed to be zero. Otherwise the system has an infinite impulse response (IIR).

You can directly implement equation (6) in LabVIEW using two shift-registers—one to keep previous input values and the other to keep previous output values. Figure 2 shows the block diagram for the case where \( n = m = 3 \).

![Figure 2. Block Diagram for Difference Equation](image-url)
By introducing the new sequence $u_k$ such that

$$y_k = \sum_{j=0}^{m} a_j u_{k-j}$$

(7)

and using convolution, equation (6) becomes a difference equation in $u$ of the form:

$$u_k = x_k \cdot \sum_{i=1}^{n} b_i u_{k-i}$$

(8)

The block diagram implementation of equations (7) and (8) is shown in Figure 3. Notice that this implementation uses a single shift register as opposed to the implementation of equation (6) (Figure 2), which uses two shift registers. The single shift register approach requires only $\max(n,m)$ units of memory, while using two shift registers requires $(n+m)$ units of memory. The solution in terms of the $u_k$ sequence is therefore slightly more space efficient.

The values contained in the left terminals of the shift register constitute the state of the linear system at any given point in discrete-time.

**The Z-Transform**

The Z-Transform is an important mathematical tool for studying discrete-time linear systems. You can use the Z-Transform to map discrete-time domain models into discrete-frequency domain models, often revealing features not evident in the time domain. Furthermore, the convolution operations that arise from setting up difference equations correspond to multiplication operations in the Z-Transform domain.
The one-sided Z-Transform of a discrete sequence $x_n$ is

$$X(z) = Z\{x_n\} = \sum_{i=0}^{\infty} x_i z^{-i} \quad (9)$$

where $x_n = \{x_0, x_1, x_2, \ldots, x_n\}$.

The Z-Transform of a delayed sequence $x_{n-d}$ is related to the Z-Transform of the original sequence $x_n$ by the expression

$$Z\{x_{n-d}\} = z^{-d} Z\{x_n\} \quad (10)$$

where $d$ is an integer number of discrete delays.

Applying the Z-Transform to equation (6)

$$Y(z) (1 + \sum_{i=1}^{n} b_i z^{-i}) = X(z) \sum_{j=0}^{m} a_j z^{-i}, \quad (11)$$

solving for the ratio $H(z) = Y(z) / X(z)$ and expanding the summation terms yields

$$H(z) = \frac{Y(z)}{X(z)} = \frac{a_0 + a_1 z^{-1} + a_2 z^{-2} + \cdots + a_m z^{-m}}{1 + b_1 z^{-1} + b_2 z^{-2} + \cdots + b_n z^{-n}}. \quad (12)$$

$H(z)$ is known as the system function and it describes the relationship between the input and the output sequence.

Thus, if a linear system can be described in terms of a system function, then the system's difference equation can be set up using $a = \{a_0, a_1, a_2, \ldots, a_m\}$ as the forward coefficients and $b = \{b_0, b_1, b_2, \ldots, b_n\}$ as the feedback coefficients.

**Signal Flow Graphs**

You typically use signal flow graphs to represent difference equations pictorially. They are composed of the symbols shown in figures 4, 5, and 6, which represent summation (addition of all the input elements to produce an output), one unit discrete-time delay, and multiplication by a constant, respectively.

- Figure 4. Summation
- Figure 5. Unit Time Delay
- Figure 6. Multiplication

The signal flow graph in Figure 7 represents equation (6), and the signal flow graph in Figure 8 represents equations (7) and (8). The signal paths with nonzero $b_i$ coefficients form the feedback cycles and create an infinite impulse response.
Figure 7. Flow Graph of Equation (6)

Figure 8. Flow Graph of Equations (7) & (8)
Converting Signal Flow Graphs to LabVIEW Diagrams

You can easily convert signal flow graphs into LabVIEW block diagrams by applying some simple transformations. This section presents a few of the many ways you can convert signal flow diagrams to LabVIEW block diagrams.

To transform a signal flow graph into a LabVIEW block diagram, replace the signal flow elements shown in the left column of the following table with the LabVIEW functions shown in the right column.

<table>
<thead>
<tr>
<th>Signal Flow Element</th>
<th>LabVIEW Equivalent</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Signal Flow Element 1" /></td>
<td><img src="image2.png" alt="LabVIEW Equivalent 1" /></td>
</tr>
<tr>
<td><img src="image3.png" alt="Signal Flow Element 2" /></td>
<td><img src="image4.png" alt="LabVIEW Equivalent 2" /></td>
</tr>
<tr>
<td><img src="image5.png" alt="Signal Flow Element 3" /></td>
<td><img src="image6.png" alt="LabVIEW Equivalent 3" /></td>
</tr>
</tbody>
</table>

You can easily implement the transformation for a unit time delay using either a For Loop or a While Loop structure with shift registers. To create the unit delay, enclose the entire graph in a loop structure, which can index the input sequence elements and collect the results into the output sequence. Add a shift register to provide a unit delay, and add a left shift register terminal for every unit delay required. The resulting block diagram is shown in Figure 9.

![Figure 9](image7.png)

Figure 9. $y_i = 0.25 \times (x_i + x_{i-1} + x_{i-2} + x_{i-3})$
You need to use a loop structure for two reasons. First, it makes the implicit storage in the delay element explicit. Second, the shift register converts the explicit cycle of the signal flow graph to an implicit cycle.

An alternative way to implement a unit time delay is to replace the implicit unit delay with the block diagram segment shown in Figure 10.

Using this approach, the signal flow graph shown in Figure 11 corresponds to the LabVIEW block diagram shown in Figure 12. Both the signal flow diagram and the LabVIEW block diagram implement the equation \( y = a_0x_i + a_1x_{i-1} + a_2x_{i-2} \).
You can also hide the unit delay While Loops in a reentrant unit delay virtual instrument (VI). Reentrant execution is a mode in which multiple calls to a VI can execute in parallel, and each instance of the VI retains its own memory. You use the VI Setup... dialog box to enable reentrant execution.

Figure 13 shows the block diagram corresponding to the signal flow graph in Figure 11 using the reentrant Unit Delay VI.

There is no alternative for removing the explicit cycles in LabVIEW, so any recursive or IIR structure requires a top-level loop to hold the feedback elements in shift registers. The signal flow diagram shown in Figure 14 requires a top level loop and a shift register.

Figure 14. Recursive Structure

Figure 15 shows the corresponding LabVIEW block diagram that implements the recursive signal flow diagram shown in Figure 14.
Input and Output Sequences

The previous sections of this application note assumed that you acquired the input data as a block, that you processed the elements using the automatic indexing feature in LabVIEW, and that you reassembled the results into a block as shown in Figure 16. This type of processing is called block-mode processing.

In real-time systems the VI must process each value as it is acquired, perhaps from an analog-to-digital converter, and immediately return the result, perhaps to a digital-to-analog converter. In these cases the looping structures must include the acquisition and the result generation. This type of processing is called real-time processing.

Figure 17 shows a real-time block diagram for processing a finite number of samples.
Figure 18 shows a real-time block diagram for processing an indefinite number of samples.

![Figure 18. Point-By-Point Approach with Front Panel Switch](image)

The approach in Figure 18 is convenient if you use a front panel switch to terminate the operation, while the approach in Figure 17 is convenient if the operation processes a fixed number of samples.

**Linear System Modeling Example**

Consider a second order filtering system whose transfer function $H(s)$ is given by

$$H(s) = \frac{w_a^2}{s^2 + 1.414w_a s + w_a^2}, \quad (13)$$

where $H(s)$ is the Laplace Transform of the physical impulse response $h(t)$ of the linear system.

A system function, $H(z)$, can be obtained by letting

$$s = \frac{1 - z^{-1}}{1 + z^{-1}}. \quad (14)$$

This transformation is known as the bilinear transformation and it is often used to design discrete-time digital filters from frequency domain mathematical models of analog filters.

Substituting equation (14) in (13):

$$H(z) = \frac{a_0 + a_1 z^{-1} + a_2 z^{-2}}{1 + b_1 z^{-1} + b_2 z^{-2}} \quad (15)$$

where

$$a_0 = \frac{w_a^2}{w_a^2 + 0.414} \quad (16)$$
The difference equation describing this linear filtering operation is

\[
y_k = a_0 x_k + a_1 x_{k-1} + a_2 x_{k-2} - b_1 y_{k-1} - b_2 y_{k-2}.
\]  

The corresponding block diagram in LabVIEW is shown in Figure 19.

**Figure 19.** Example of Linear System Modeling

**References**
