Chapter 6

6.1 Physics of a second-order mechanical system

Many mechanical systems, for example a car suspension, can be modeled with a second-order differential equation. In this case, the fixed surface is the road, the car is the mass, and the suspension is the spring-and-damper component. This is diagrammed in Figure 6.1, and in the lower section the position (as a function of time) is represented as $x(t)$. The expression governing position is based on Newton's Second Law, which in modern times we frequently couch simply as $F = ma$. In the mass-spring-damper problem, let us refer to the mass as M, the spring constant as k, and the damper value as D. Conservation tells us that the we must account for all forces on the mass, and that they will balance. So, what forces act on the mass?

- Gravity: This one is the most obvious, because this force is directly computed as $F = Ma$. However, we must note that acceleration, a (or $a(t)$ to be more accurate), is simply the second derivative of position, $x(t)$. So, for gravity we have that $F_{mass} = M\frac{d^2x(t)}{dt^2}$

- Damper: The damper is a rate-of-change based device. The faster the damper tries to change position, the more damping occurs (it is noteworthy that in a car, the shock absorber is the damper, but shock absorbers are designed to have different damping rates for compression and extension; in this example we will assume a single damping rate). Because the force related to damping is related to velocity, we need the first derivative of position, so that the damping force is $F_{damper} = D\frac{dx(t)}{dt}$.

- Spring: The spring rate is a constant related to the amount of spring displacement from rest. Therefore, the force related to the spring is directly proportional to the position, $x(t)$. We can express the force due to the spring as $F_{spring} = kx(t)$.

- External Force: The external force is generally the roughness of the road, and for our purposes can simply be called F_{ext}. In the absence of external force, the mass should not be moving and the forces due to the mass, spring, and damper should sum to 0.

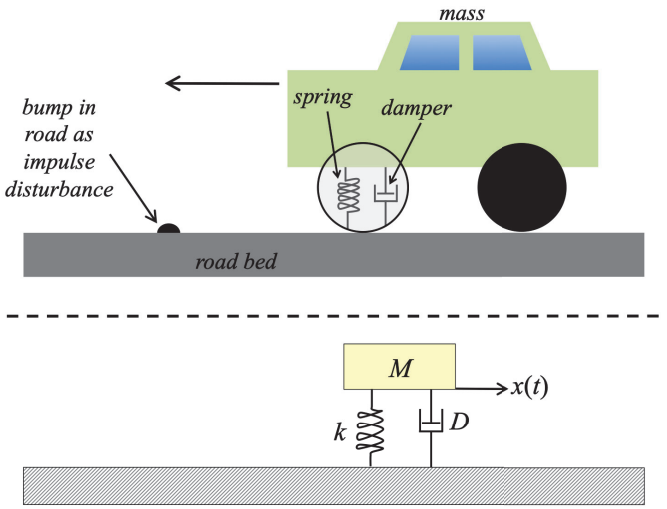

Figure 6.1. ***Mass-Spring-Damper Example:*** *The mass-spring-damper problem is a good simplified version of a car suspension. The car is the mass, the suspension spring is the spring, and the shock absorber is the damper. Position, $x(t)$, of these things relative to the fixed surface helps define the forces exerted.*

If we define positive $x(t)$ to be downward, then we can sum these forces as follows:

$$-F_{spring} - F_{damper} + F_{ext} = F_{mass}$$

$$\Downarrow$$

$$M\frac{d^2x(t)}{dt^2} + D\frac{dx(t)}{dt} + kx(t) = F_{ext}(t)$$

The interested student should review the Laplace Transform to truly appreciate this case study. We know that when a system is energized by an impulse function (the input is an impulse), the Laplace Transform of the resulting output is referred to as the *Transfer Function*. Denoting the operation of the Laplace transform using the notation $\mathcal{L}()$. Letting F_{ext} be the system input, and taking the Laplace transform of both sides of the force equation, we have

$$M\mathcal{L}\left(\frac{d^2x(t)}{dt^2}\right) + D\mathcal{L}\left(\frac{dx(t)}{dt}\right) + k\mathcal{L}(x(t)) = \mathcal{L}(F_{ext}(t))$$

If the Laplace transform of $x(t) \xrightarrow{\mathcal{L}} X(s)$, then we learn in a basic Signals and Systems course that the transform of a second derivative is given by $\mathcal{L}\left(\frac{d^2x(t)}{dt^2}\right) = s^2X(s)(ROC \subset R)$. Furthermore, the transform of a first-order derivative is given by $\mathcal{L}\left(\frac{dx(t)}{dt}\right) = sX(s)(ROC \subset R)$. Therefore, the Laplace transform of our Force equation becomes

$$Ms^2X(s) + DsX(s) + kX(s) = \mathcal{L}(F_{ext}(t))$$

Finally, a Laplace Transform table will show that the transform of an impulse function is given by $\mathcal{L}(\delta(t)) = 1$. Replacing the right hand term of the equation and recognizing the transfer function relates the input (external force) and output (position) we get

$$\begin{aligned}\left(Ms^2 + Ds + k\right)X(s) &= 1 \\ H(s) &= \frac{1}{X(s)} \\ &= \frac{1}{Ms^2 + Ds + k}\end{aligned}$$

When the denominator of this transfer function is set equal to zero, the expression is called the *characteristic equation*. This is an extremely important equation because the roots of the denominator tell us the underlying, unforced behavior baked into the system by design (in other words, ignoring external perturbations of the system). Recall that the general solution to a linear, constant coefficient differential equation is based on the natural exponent, $e^{\alpha t}$, where we let $\alpha \in \mathbb{C}$ for the most general case. Because $e^{\alpha t}$ is an *eigenfunction*, the roots of the characteristic equation determine the *eigenvalues* of the system. When the eigenvalues are real (repeated or distinct), the resulting impulse response in the time domain is damped so that after the impulse the output simply decays over time. This is shown in Figure 6.2. On the other hand, a second order system with complex roots has an oscillatory impulse response. It should be noted that complex roots *always* come as complex conjugate pairs. An odd-order system, then, must have at least one real root! The oscillatory nature of the response can be understood in relation to Euler's relationship. For example, the cosine function can be written using complex values as

$$\cos(\omega t) \quad = \quad \frac{e^{j\omega t} + e^{-j\omega t}}{2}.$$

It is critical to note that these roots, whether they are complex or real, occur in the Laplace Domain. We must still use a process called *partial fraction expansion* (PFE) to get the individual Laplace components that can be easily transformed back into the time domain. The point is that a pair of complex conjugate roots in the Laplace Domain will translate into a decaying sinusoid in the time domain.

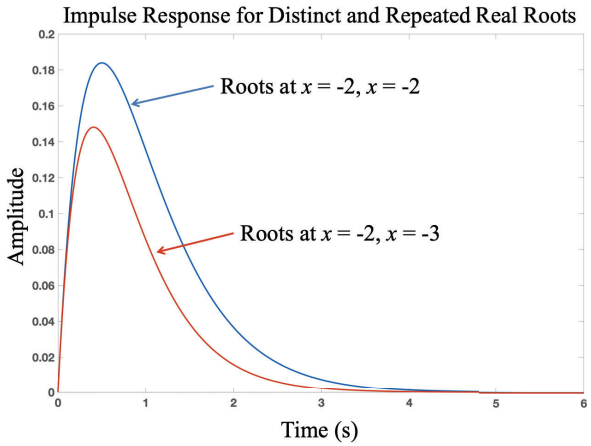

Figure 6.2. ***Impulse Response of Two Second-Order System With Real Roots:*** *The taller trace is due to a second order system with two real, repeated roots as shown. The slightly lower-amplitude trace represents the impulse response of a second order system with two real, distinct roots. In each case, after reaching a peak the signal monotonically decays to zero over time.*

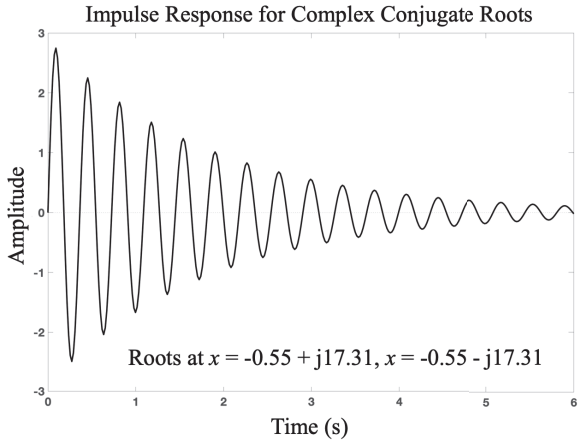

Figure 6.3. ***Impulse Response of a Second-Order System With Complex-Conjugate Roots:*** *The taller trace is due to a second order system with two real, repeated roots as shown. The slightly lower-amplitude trace represents the impulse response of a second order system with two real, distinct roots. In each case, after reaching a peak the signal monotonically decays to zero over time.*

The reader should recall that one of the seminal Laplace Transform relationships is given by

$$Ae^{\alpha t} \overset{\mathcal{L}}{\longleftrightarrow} = \frac{A}{s+\alpha}$$

However, we know that in the general case where $\alpha \in \mathbb{C}$ we get a complex exponential in the time domain. If we have two terms in the Laplace domain (due to the known fact that the roots of the characteristic equation come in complex conjugate pairs), then we will get two complex-conjugate exponentials in the time domain, resulting in a sinusoidal result. Digging a bit deeper, the numerator value carries across between the time and Laplace domains. If the numerator is complex (which it will be if the roots of the original transfer function are complex conjugates), the other numerator (in a second-order system) will be the conjugate pair. The residuals translate directly into the time domain. Separating the real and imaginary parts of the residuals (the numerator values in the Laplace domain after PFE), we can carry the imaginary part into the sinusoid in the time domain as a phase shift, leaving the real part of the exponent as our decaying term. Using our example, this is described mathematically as

$$H(s) = \frac{1}{Ms^2 + Ds + k}$$

$$\Downarrow \qquad \textit{taking the roots of the denominator}$$

$$H(s) = \frac{1}{\left(s + \frac{D + \sqrt{D^2 - 4Mk}}{2M}\right)\left(s + \frac{D - \sqrt{D^2 - 4Mk}}{2M}\right)}$$

$$\Downarrow \qquad \textit{letting each root be represented by } -\alpha \pm j\omega$$

$$H(s) = \frac{1}{(s + (\alpha - j\omega))\,(s + (\alpha + j\omega))}$$

$$\Downarrow \qquad \textit{partial fraction expansion to get two first-order terms}$$

$$H(s) = \frac{-j\frac{1}{2\omega}}{s + (\alpha - j\omega)} + \frac{j\frac{1}{2\omega}}{s + (\alpha + j\omega)}$$

$$\Downarrow \qquad \textit{using the inverse Laplace transform}$$

$$\begin{aligned} h(t) &= -j\frac{1}{2\omega}e^{(-\alpha + j\omega)t} + j\frac{1}{2\omega}e^{(-\alpha - j\omega)t} \\ &= \frac{1}{2\omega}e^{-j\frac{\pi}{2}}e^{(-\alpha + j\omega)t} + \frac{1}{2\omega}e^{j\frac{\pi}{2}}e^{(-\alpha - j\omega)t} \\ &= \frac{1}{2\omega}\left[e^{-\alpha t}e^{j\left(\omega t - \frac{\pi}{2}\right)} + e^{-\alpha t}e^{-j\left(\omega t - \frac{\pi}{2}\right)}\right] \\ &= \frac{1}{2\omega}e^{-\alpha t}\left[e^{j\left(\omega t - \frac{\pi}{2}\right)} + e^{-j\left(\omega t - \frac{\pi}{2}\right)}\right] \end{aligned}$$

$$\Downarrow \qquad \textit{using Euler's relationship}$$

$$h(t) = 2\left(\frac{1}{2\omega}\right)e^{-\alpha t}\cos\left(\omega t - \frac{\pi}{2}\right)$$

The final term uses cosine with a $\frac{\pi}{2}$ shift rather than using sine. Either expression would be correct. This author is an Electrical Engineer by trade, and most Circuits 2 textbooks, when discussing partial fraction expansion, provide the basic relationship for a second-order system with complex roots, that has a polynomial on s for both the numerator and denominator (we had a simple constant in the numerator for our derivation) such that

$$H(s) = \frac{N(s)}{D(s)} \quad = \quad \frac{K}{s + (\alpha - j\omega)} + \frac{K^*}{s + (\alpha + j\omega)}$$

$$\Downarrow \qquad \textit{Inverse Laplace transform}$$

$$h(t) = 2|K|e^{-\alpha t}\cos(\omega t + \varphi) \qquad \textit{where } \varphi = \arg(K)$$

noting that arg(K) simply means finding the *argument* of the complex number, K. The argument of a complex number is the angle (in the complex plane) that the complex number describes, taking into account the proper quadrant. The argument is found by taking the arctangent of the imaginary part divided by the real part of the complex number. The quadrant is important and cannot be ignored. The inverse tangent computation of $\frac{-\beta}{\rho}$ and $\frac{\beta}{-\rho}$ produce the same value, but are clearly not in the same quadrant.

Setting aside the details of this derivation, the fundamental message is that the *impulse response*–that is, the function that describes the result of an impulse in the time domain–has the form of a decaying exponential, where the decay rate is controlled by the variable α in the above equations, and the oscillation is determined by ω. IN the mass-spring-damper system it is possible to control the variables so that the roots of the system are real and the system is highly damped. In a car, this would provide an unacceptable stiff and uncomfortable ride. On the other hand, one does not want the car to bounce down the road, so auto designers have worked hard to find the appropriate amount of damping and spring action to provide a comfortable ride, but maintain good handling characteristics.

6.2 Project

The physics and math of a second-order mass-spring-damper system also apply when the mass is suspended from a fixed surface (as described in Chapter 4 under accelerometers). Referring to Figure 4.10, we see the same mathematical derivation that provides a second-order differential equation for the position of the mass with respect to time. For this project, a mass-spring-damper system (a *bouncy toy*!) is suspended from the ceiling with a tape measure affixed from the ceiling to a table beneath the toy. The tape measure should be assumed to be parallel to the vertical motion of the toy. We do not know the mass of the toy, the spring constant for the spring suspending it, or the damping constant of its paddle blades. The toy is shown in Figure 6.4.

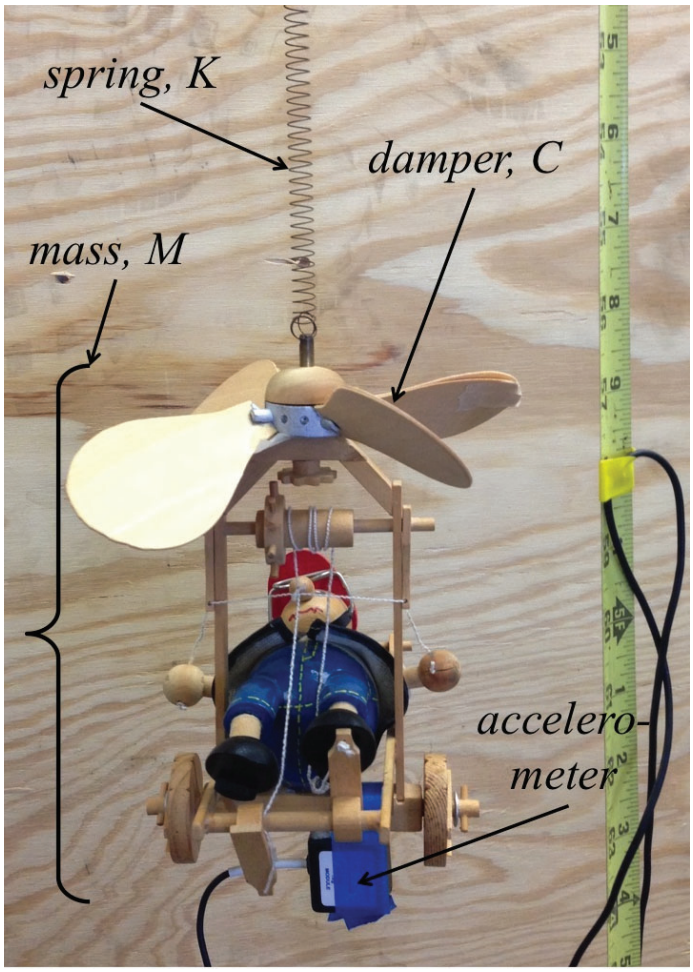

Figure 6.4. ***Bouncy Toy Example of Mass-Spring-Damper System:*** *The bouncy toy (including the accelerometer attached to the bottom) has some mass, and is suspended from a spring with spring constant k. The non-ideal nature of the spring and the big paddle-shaped, rotor-like blades provide the damping factor.*

The sensor that will provide some insight into the system is an accelerometer. We do not know the sensitivity of the accelerometer, but we do know that its output is a single-ended voltage that ranges between 0 volts and the power supply voltage of +5 volts. It is noteworthy that both negative and positive acceleration is represented in this voltage range, so the user should expect a DC offset when the system is completely idle. The tape measure is provided so that the user can measure the initial displacement for an impulse response.

The goal of this project is to determine the *motion equations* for the device. For our purposes, the term motion equations implies the equation for $x(t)$ (position), the equation for $v(t)$ (velocity), and the equation for $a(t)$ (acceleration). The available data to be collected are a series of voltages that represent acceleration of the device over time, but we do not

know the relationship between the recorded voltages and acceleration. In general, we might take the following steps in executing this project.

1. Choose the sensor. In this case, the user would consider what sort of g-load is expected, what kind of bandwidth is needed (how fast will it oscillate), and what type of electrical output it will provide.

2. Choose the data acquisition device: single-ended or differential capability, dynamic range capability, etc.

3. Assemble the measurement system, including the sensor wiring, the connection from the sensor to the data acquisition device, and the connection from the data acquisition device to the computer.

4. Produce the data acquisition software, accounting for dynamic range, sample rate, format (single-ended versus differential), duration of acquisition, etc.

5. Analyze the data (presumably software-based analysis)

For this project, steps 1 and 2 have already been completed. The accelerometer is chosen (not that it is guaranteed to be the ideal), and the data acquisition device is already present. The data acquisition device is a National Instruments USB-6211. The instructor will provide the channel number for the accelerometer signal (step 3). So, the student's main efforts will center on steps 4 and 5.

For step 5, the student must understand the fundamental concepts provided in Chapter 2. Specifically, the student must determine the sampling rate. How does one go about this? Also, the student must determine the dynamic range. How does one determine that? The connection configuration must also be correctly set. This project employs the LabVIEW programming environment to acquire and analyze data. Appendix (XX) provides a basic tutorial for LabVIEW. However, students should internalize the idea that engineering courses do not award ABET accredited engineering credits for learning a commercial software product. So, this book is not in the nature of a LabVIEW training mechanism. Engineering students all take a programming basics course, and the fundamental concepts of programming are no different in LabVIEW than they are in C, Java, Python, etc. A While

loop is a While loop, a For loop is a For loop, conditional statements (if-then) are just that, regardless of the programming language. So, the task in this case is to learn enough of another language to get the job done. It is a bit like a native English-speaking person, who also speaks French, finding herself in need of interacting in Spanish at a restaurant. With a little help from a vocabulary book, this can be done, because the so-called romance languages all have the same (or very similar) fundamental grammar constructs.

6.2.1 Data Acquisition Programming

Once the hardware setup is complete, the next step is to write a small LabVIEW program to acquire the data. The first step in this process is to program the DAQ device itself. Fortunately, LabVIEW has a number of Express Wizards to help the user begin data acquisition in short order. Assuming the student knows how to open a LabVIEW VI session, we first make the *Block Diagram* the active Window. This is done with the key combination CTRL-e, or by going to the Window sub-menu at the top of the Main Diagram window and selecting Block Diagram. Once the Block Diagram is the active window, the user should *right-click* to bring up the *Functions Palette* (see Figure 6.5).

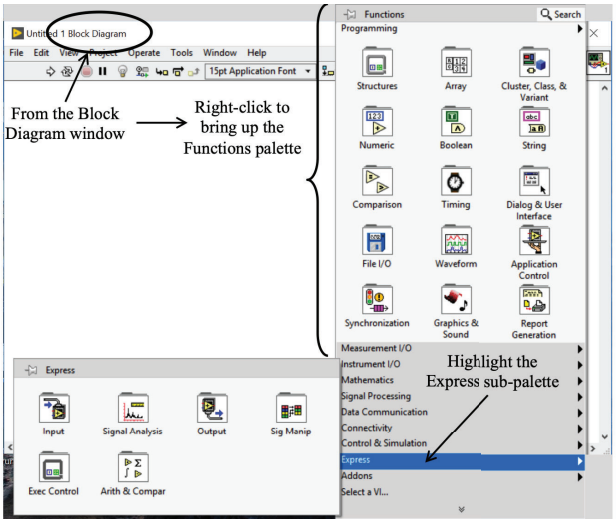

Figure 6.5. ***Block Diagram of the VI with the Functions Palette Activated:*** *To get to the functions that we program with in the Block Diagram, right-click to get the Functions Palette.*

When the Functions Palette is visible, highlighting (but not clicking) on the Express sub-palette brings up anther small window (see Figure 6.5).

Once the Express sub-palette is open, hovering the mouse over the Input icon brings up yet another sub-palette. In this final sub-palette, the user should click on the DAQ Assist icon (see Figure 6.6). The DAQ Assist is a Wizard that walks the user through the data acquisition steps of choosing a device (it must be actively seen by the system before this step), selecting a channel (or channels–note that one can select multiple sequential channels by selecting the first desired channel, holding down the SHIFT key, and selecting the last desired channel; or multiple non-sequential channels by holding down the CTRL key and selecting each individual desired channel), and the programming the acquisition parameters associated with the selected channels.

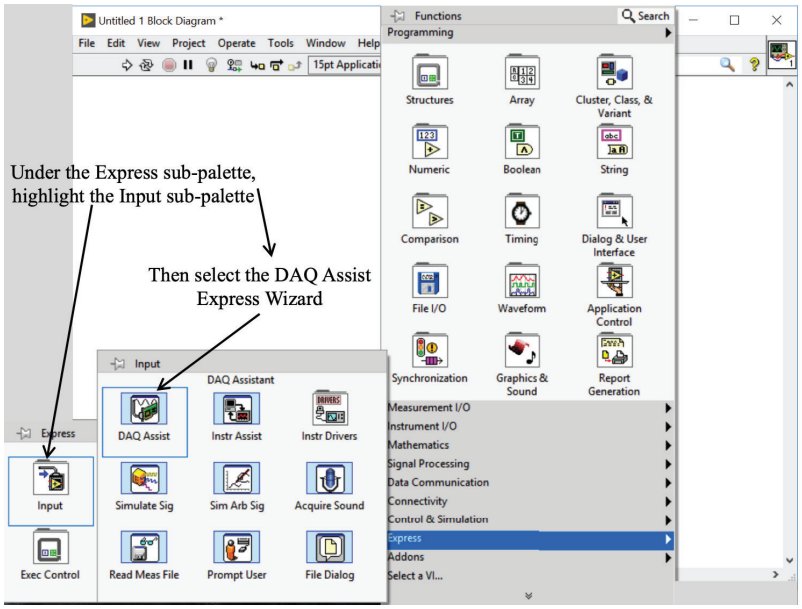

Figure 6.6. ***Getting to the DAQ Express Wizard:*** *Going through several sub-palettes we finally arrive at the palette containing the DAQ Express icon. Selecting this icon activates a Wizard that will walk the user through the setup process.*

Once the DAQ Assist window opens, the user is guided through the process of selecting a device, channels, etc. The first step is to determine if the user is acquiring or generating signals. For this project the user only needs to acquire a single channel of data, so the Acquire Signals option is expanded (see Figure 6.7).

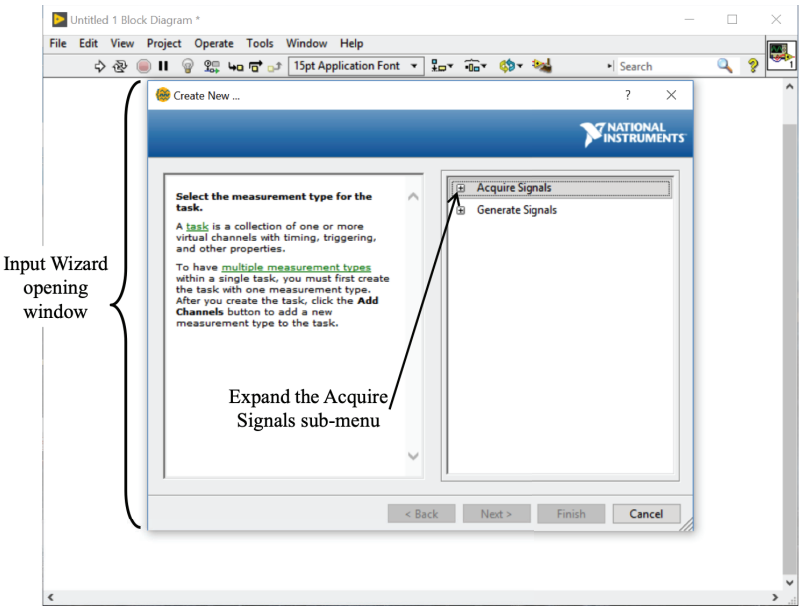

Figure 6.7. ***Acquisition Wizard–Expanding Acquire Signals:*** *Once the initial acquisition window opens, the user selects Acquire Signals to find a device and select channels for data acquisition.*

After expanding the Acquire Signals sub-menu, the user should expand the Analog Input sub-menu (see Figure 6.8).

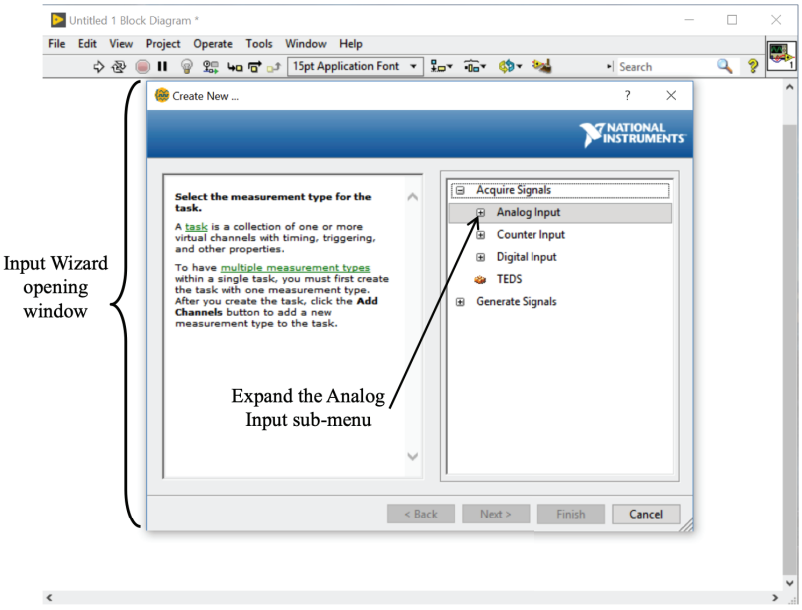

Figure 6.8. ***Getting to the DAQ Express Wizard:*** *Going through several sub-palettes we finally arrive at the palette containing the DAQ Express icon. Selecting this icon activates a Wizard that will walk the user through the setup process.*

Although not always the signal option for any given measurement project, the projects presented in this text are all voltage based, so the user should select Voltage (the first option) under the Analog Input submenu (see Figure 6.9). Indeed, the user can see from Figure 6.9 that there are a number of other options for analog input. Some of these are also voltages, but they are conditioned voltages, such as the Strain option, or the Temperature option. The LabVIEW software environment is designed to accommodate a number of industry sensor types, and helps the user accommodate those sensor types in terms of bridge constants, or thermocouple type.

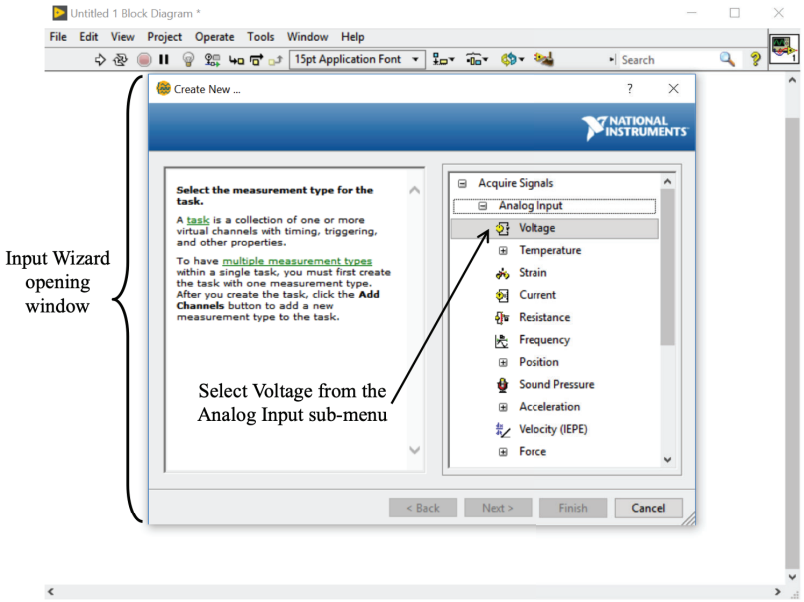

Figure 6.9. ***Choosing Voltage as the Analog Input Type:*** *Although some projects may require a thermistor input, or some other type of input, the accelerometer was chosen to have a simple voltage output that is proportional to the g-load, so the user should choose Voltage as the analog input type.*

The user has finally arrived at the point of selecting the desired input channel(s). For this specific project there is only one input channel and the instructor will inform the student which channel to use. In general, once can select multiple channels in two separate ways. If the multiple channels are sequential, the user can select the first desired channel, then hold the SHIFT key and select the last desired channel. This will highlight all channels in between these two selected ones. If the multiple channels are not sequential, the user can hold down the CTRL key and then individually select each desired channel (see Figure 6.10).

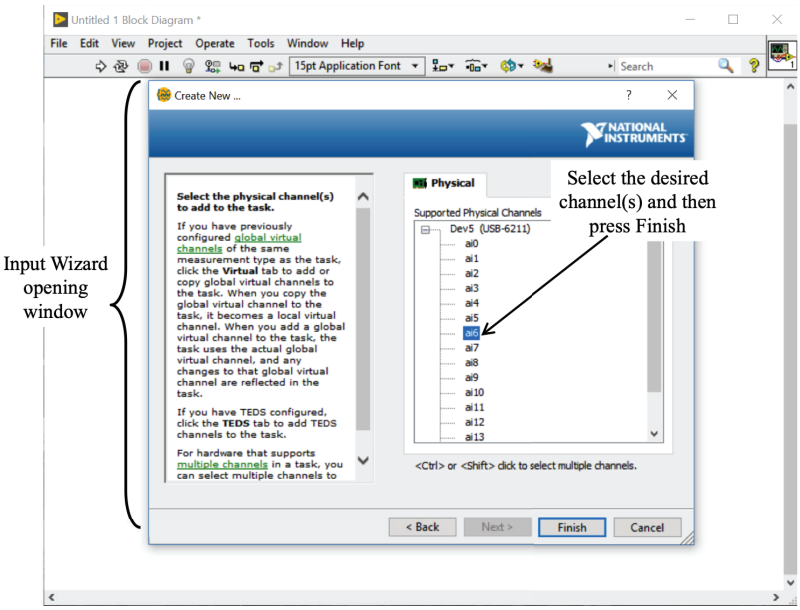

Figure 6.10. ***Choosing Individual Voltage Input Channels:*** *The user selects each channel with a mouse click. A single channel can be chosen (as in this project), or multiple channels can be chosen (as described in the text).*

Once the channel(s) have been chosen, the channel configuration window appears and the user must supply values to configure the channel(s) appropriately (see Figure 6.11). These parameters include

- Dynamic range: both upper and lower voltage limits expected for the incoming signal.
- Connection configuration: Differential or Reference Single-ended (RSE) are the two most common.
- Sampling rate: This is the sampling frequency the user has determined to meet the Nyquist criterion.
- Bin Size: this is a buffer associated with the actual A/D hardware; if the bin size is set to k samples, then A/D converter converts k samples, collecting them in the buffer (bin), and then empties the buffer to the processor and collects another k samples. The conversion rate is *always* the sampling rate, but the bin size determines how often the buffer of sampled values is sent to the processor. A good starting point is 10-40% of the sample rate. The update rate of the loop has an effect on how plots update from one cycle to the next. Longer update times with fast moving data can make then plots look jerky.

- Acquisition Mode: The user must decide to acquire a *single sample*, *N samples*, or *continuous samples*. Continuous samples require the process to be placed in a WHILE loop; the option to instantiate the WHILE loop properly is part of the Wizard and the user should ALWAYS accept this option.

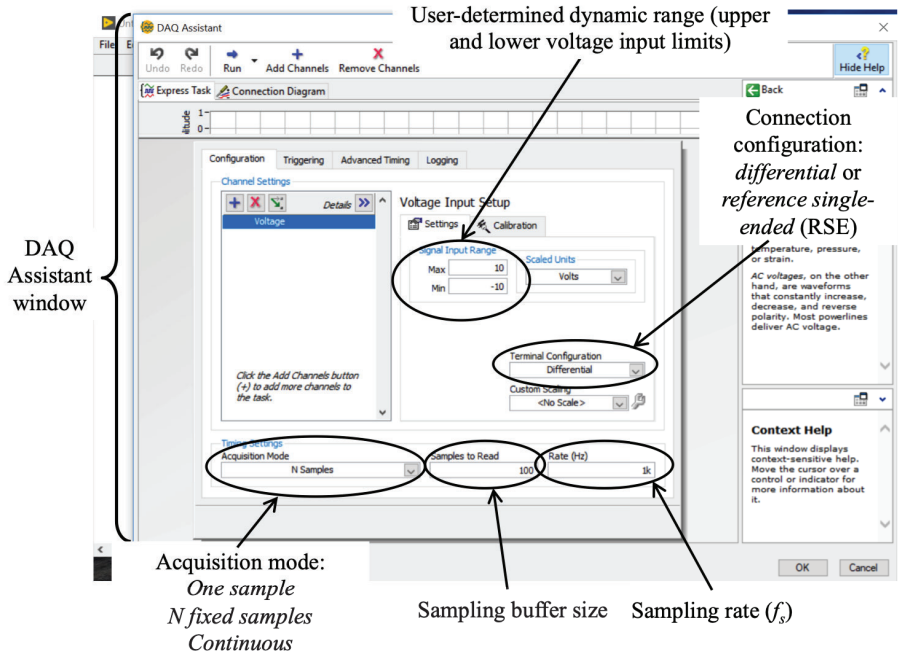

Figure 6.11. ***Setting the Acquisition Parameters:*** *The acquisition parameters must be set correctly to ensure data values that are representative of the actual process being measured.*

There are several subtle issues in setting these parameters that can have a significant effect on the results. The bin size, relative to the sample rate, has been discussed, but it mostly affects the visual aspect of the user interface. A more fundamental issue occurs with ongoing data collection and the mode of acquisition. Specifically, if we collect data over many cycles of a WHILE loop, we want to choose *continuous* acquisition. The reader might wonder why we cannot just choose N samples each time through the loop. There are two problems with this

1. Timing: The A/D converter converts analog values to digital samples at some sampling rate, f_s. In continuous mode, the bin (buffer) collects k of the digital conversions into the bin, and then passes those samples in bulk to the computer processor (over a USB connection in this case). Because the system is set to a continuous acquisition mode, the A/D converter continues to convert samples at the specified sample rate.

Therefore, the acquisition device must ensure that it can empty its buffer (bin) and return for the next converted sample within the sample period. Note that the sample period is simply the inverse of the sample rate, so if we set our sample rate to 100 Hz and our bin size to 20 samples, then the A/D converter converts and places the 20^{th} sample into the bin, and from that moment the acquisition device has 0.01 seconds to empty the buffer (i.e. transfer those 20 samples from the bin to the computer's processor) before the A/D converter will try to put a new converted sample into the bin. Furthermore, the remaining code must execute before the buffer gets full again, which in this example is 20/100 = 0.2 seconds. Of course, 0.2 seconds is a *long* time in computer processing! But, the continuous mode guarantees that the data are just that, continuous (and evenly sampled).

If the N samples acquisition mode is chosen, the buffer timing problem does not occur, because the A/D converter stops converting until the next loop cycle begins and the computer initiates another data acquisition of N samples. Once the data are collected the A/D converter is flushed and shut down and the data are transferred to the computer's processor. Therefore, the transfer time and the duration of the remainder of the code are irrelevant. There are no samples being queued up by the converter because it was told to collect N samples and stop. But this means that the interval between the last sample of cycle n and the first sample of cycle $n + 1$ is not necessarily the same as the regular interval between the N samples of each loop cycle. Indeed, from a programming perspective, we can't even guarantee that the loop cycle intervals are consistent because the central processor may have other operating system children processes running that can take precedence for several cycles at random times. This mode of data acquisition is not good for most ongoing data collection processes.

2. Filtering: The discrete filters in LabVIEW are quite intelligent functions. However, if one chooses N samples as the acquisition mode, the filter treats each new set of N samples (each time through the loop) as a new set of data with no knowledge of previous data states. This problem is overtly demonstrated in higher-order filters that have transient responses for the first several samples. For example, a lowpass filter may take 50 out of 500 samples to initialize and produce a valid output. This would cause a cyclic transient in the long-term output signal that might be confused for real data. In continuous acquisition mode, the filters are designed to maintain the end state from the previous loop so that the filters carry over with no transients.

In all of the projects associated with this text, the continuous mode should be selected. After completing all the required fields, the user can press OK and leave the DAQ Assistant. As LabVIEW compiles the code for execution, it will discover that the Continuous Acquisition mode has been selected and pop up a window telling the user that this mode requires a WHILE loop. It asks the user if LabVIEW should go ahead and set up the WHILE loop. You should always accept (click Yes) this option (see Figure 6.12). LabVIEW sets up the stop control on the user interface, and wires it so that pressing the stop control shuts down the A/D converter before terminating the loop and ending the program see Figure 6.13). This is essential, because it is undesirable to have the A/D converter aborted during operation. This leaves it in an unknown state, with a partially filled buffer, and it may not communicate properly with the computer moving forward.

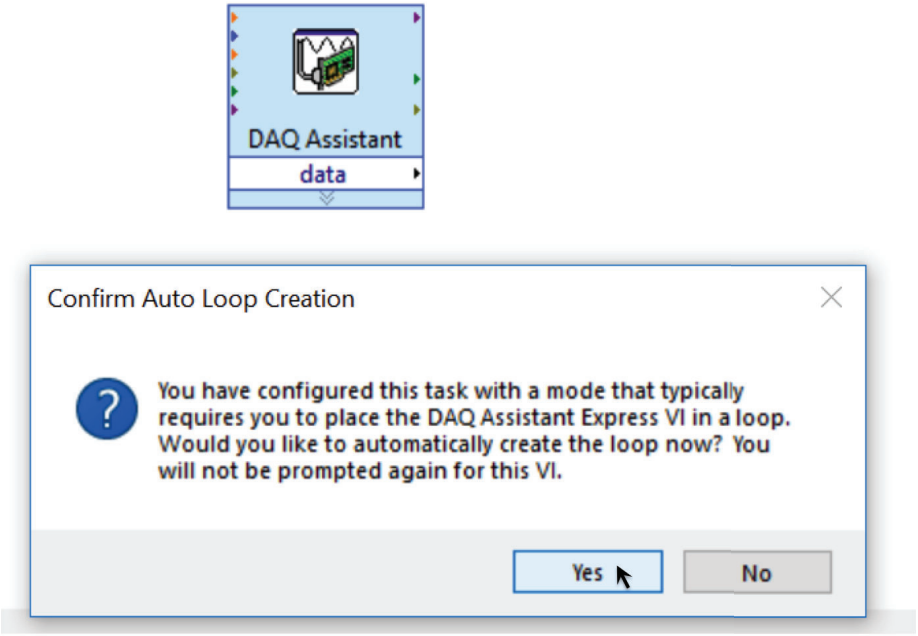

Figure 6.12. ***User Prompt for WHILE Loop Configuration:*** *When Continuous Acquisition mode is chosen, the user prompt appears asking if the user wishes LabVIEW to go ahead and create the WHILE loop correctly for proper program termination.*

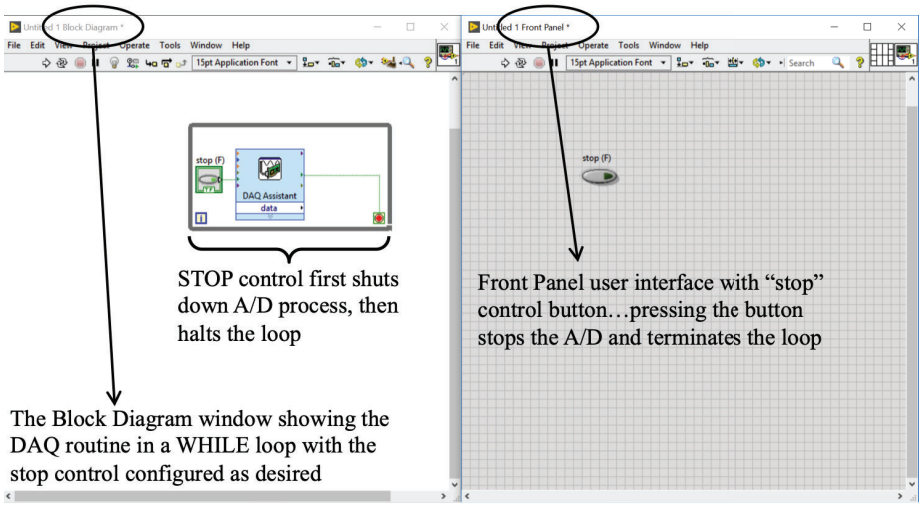

Figure 6.13. ***Correctly Configured WHILE Loop for Continuous Acquisition:*** *Once properly configured, the stop control button on the user interface, when pressed, will first terminate the A/D process in a clean manner–ensuring that the converter is idled and buffer(s) are flushed), before halting the loop itself.*

Once the actual acquisition process is programmed, the user must still make some decision about how to collect the measurements. What do we do with the k measurements in the buffer each cycle? For this project, we are going to need to collect data over some time period–it is left for the user to make this engineering decision–and we want that data to be sequential. We might use a FIFO (first-in-first-out) buffer, as this collects the data and places it in the buffer in time-order. In LabVIEW, a FIFO buffer is called a *Collector*. As with other processes, you can find this by typing it into the Search window (Collector Express VI is what you want). The collector can be set to hold m samples, so this must be determined based on how many seconds of data one collects, and the sample rate (samples per second). For example, if we set our sample rate to 100 Hz (100 samples per second), our bin size to 50 samples, and our Collector to 3000 samples, then the Collector will store 30 seconds of data at any given time. After the first 30 seconds of data collection, once the buffer (Collector) is initially filled, each new cycle of the loop brings another 50 samples into the buffer. Since the buffer size is set to 3000 samples, each cycle after 30 seconds first drops the 50 *oldest* samples from the Collector (they were the first ones *into* the buffer, so they are then first ones *out of* the buffer), shifts the remaining 2950 samples down, and finally places the newest 50 samples into the empty buffer slots. The buffer will always contain the latest 30 seconds of measurements. Figure 6.14 shows the collector in place, as well as a *graph indicator* so the process can be observed by the user during acquisition.

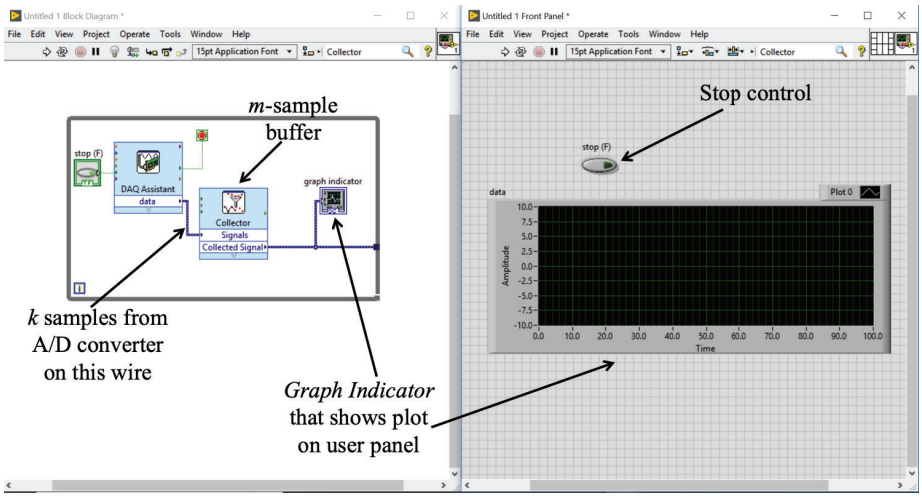

Figure 6.14. ***Data Acquisition Code for Bouncy Toy:*** *The data acquisition code is straightforward once the acquisition parameters have been set in the Wizard. The collector is a buffer and can store any desired length. It is useful to have a graph indicator attached to the collector output to see the overall data during acquisition.*

With the program running, the user can displace the toy by a known amount, and then watch until the buffer has a desired set of data points (i.e., the user ensures that there are no transient data points from the startup phase of the process) When the data look good, the user can press the Stop Control on the Front Panel to terminate the loop and pass the collector data buffer out of the loop to the analysis portion of the software.

The student is left the task of using the LabVIEW knowledge gained from the Appendix to create the analysis code. But the following things should be considered:

- Why does double integration not provide us the information we desire? Aside from the underlying problems with numerical integration, what would we need to know about our experimental setup to get a unit-correct position equation from the raw accelerometer data?
- By simply exercising the toy, it is obvious what form the position equation has (complex conjugate roots). Since the position equation is formed of two eigenfunctions (yes, cosine and sine are eigenfunctions, as well as $e^{\alpha t}$), what relationship between eigenfunctions and eigenvalues allows us to glean the position equation parameters from the accelerometer data?
- The student can use the search function in LabVIEW to find the VIs needed to analyze the data. For example, there is an Express VI under Signal Analysis for finding frequency content (Spectral for all frequencies, or Tone for a limited number

of frequencies). The student should consider what points on the decaying oscillation plot contribute to the decaying portion; what approach might be taken to identify these points?

- For each VI the student identifies, it is extremely important for the student to first read the Help information and understand precisely the required inputs and outputs; data types, value ranges, etc.

Figure 6.15 provides some hints into what the student is looking for in order to determine the position equation of the toy with respect to time. Hopefully the student has inferred from this exposition that the position equation is the first ting to find, followed by the velocity and acceleration equations. The reader should be considering why we don't just integrate the raw data twice to get velocity (first integration) and position (second integration) directly. The answer lies in our knowledge of the accelerometer and an understanding that numerical integration is fraught with pitfalls that take significant signal processing efforts to mitigate. However, by finding the position equation first (with correct units!)it is incumbent on the student to analytically compute the velocity (first derivative) and acceleration (second derivative), with proper units for those phenomena.

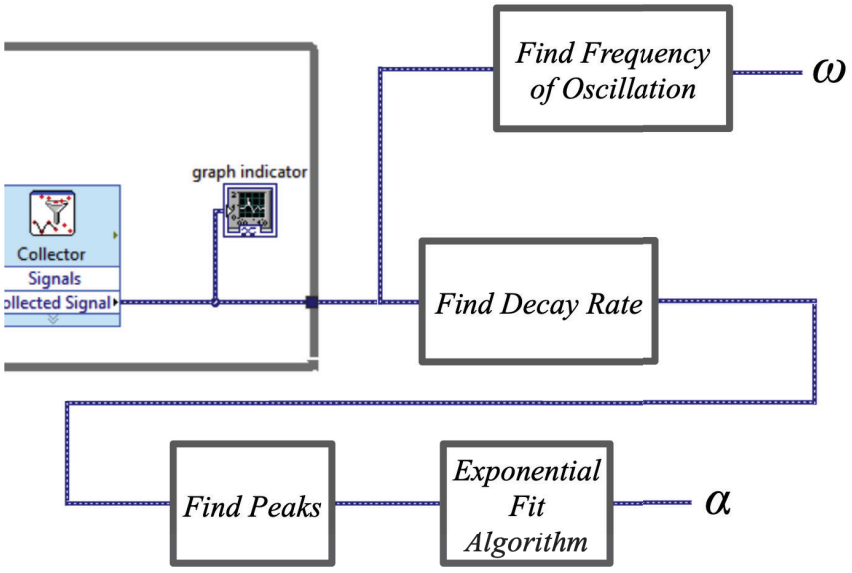

Figure 6.15. ***Data Analysis Goals:*** *After the WHILE loop is exited by the user, the analysis code must solve for oscillation frequency and decay rate in order to determine the position function (finding the decay rate may take several steps, as indicated.*