Laboratory #1

Frequency Domain Representation of Signals

by

Adrian D. C. Chan, Geoffrey C. Green and Peter X. Liu

SYSC3501 Communication Theory

Department of Systems and Computer Engineering
Faculty of Engineering and Design
Carleton University

1 Purpose

The purpose of this laboratory is to ensure that the student understands that time-domain signals can be represented in an alternate form (in the frequency domain) through the use of Fourier analysis by means of illustrative examples in Matlab Simulink.

2 Laboratory

2.1 Sinusoidal input

The simplest signal to analyze in the frequency domain is a pure sinusoid. The expression for a pure sinusoidal signal with a frequency of f_0 Hz and peak amplitude of A is:

$$x(t) = A\cos 2\pi f_0 t$$

This signal is shown in Figure 1(below).

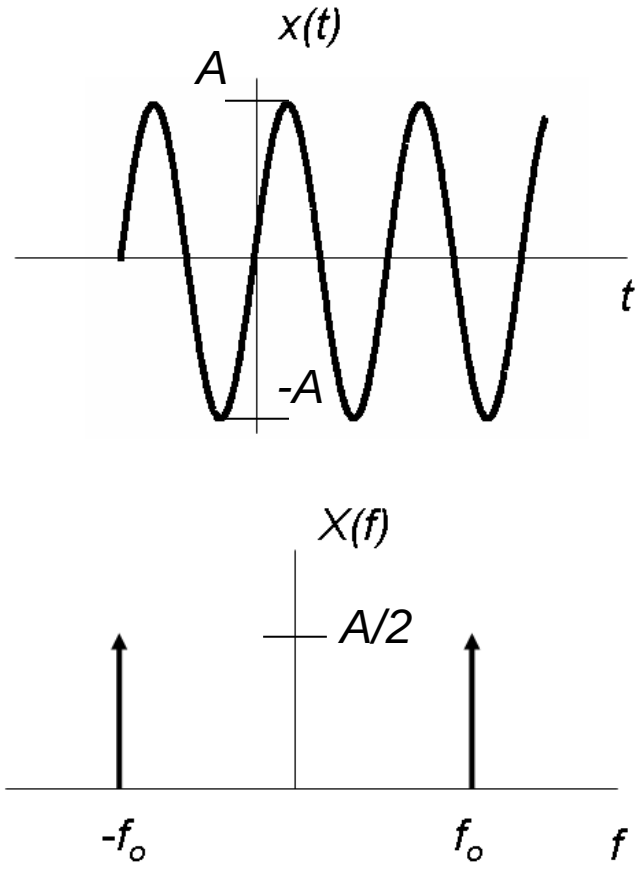

Figure 1

The Fourier transform of $x(t)$, called $X(f)$, consists of a pair of delta functions at $\pm f_0$:

$$X(f) = \frac{A}{2}[\delta(f - f_o) + \delta(f + f_o)]$$

A plot of $X(f)$ is shown in Figure 1.

It is important to realize that these expressions are simply two different ways of describing the same signal (in this example, a pure sinusoid).

- In the case of $x(t)$, the time domain representation shows the oscillatory behaviour of a sinusoid at frequency f_0 for all time.
- In the case of $X(f)$, the frequency domain representation illustrates that the $x(t)$ signal has its frequency information entirely localized at a single frequency, $\pm f_0$

For $x(t)$ as given above, an alternate frequency domain representation is the power spectral density (PSD), $p_x(f)$, a quantity that gives the density of a power signal's total power as function of frequency. The PSD expression for $x(t)$ is:

$$p_x(f) = \frac{A^2}{4}[\delta(f - f_o) + \delta(f + f_o)]$$

We can use a software program running on a digital computer to show the frequency representation of various signals. When we do this, we run into practical considerations which make a simple computation of the Fourier transform (or PSD) impossible. Recall the definition of the Fourier transform of a signal $x(t)$:

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

Immediately we see a large problem – any representation of a signal stored in a computer will have to have a finite duration (due to memory limits), so it is impossible to store $x(t)$ for all values of time (from $-\infty$ to ∞), as the integral above requires. Moreover, a digital computer can only store discrete samples of $x(t)$ over this finite range, not a continuous waveform. Thus, a strict evaluation of the Fourier transform of $x(t)$ in software is impossible. The same argument holds for $p_x(f)$.

Fortunately, there is a way of *approximating* the frequency representation of $x(t)$ based on a finite-length discrete signal. We will use a method called the *periodogram estimate* in this lab. Bear in mind that this method provides only an *estimate* of the PSD, so the observed results will deviate somewhat from the ideal case.

2.1.1 Build the Simulink model – Single Sinusoid

In this lab, you will be using the Simulink graphical simulation tool (from within Matlab) to simulate your input signals / systems and observe their properties / behaviours.

1. Start Matlab, then type "simulink" to open the Simulink Library Browser. Select File -> New -> Model to open up a new (empty) model.

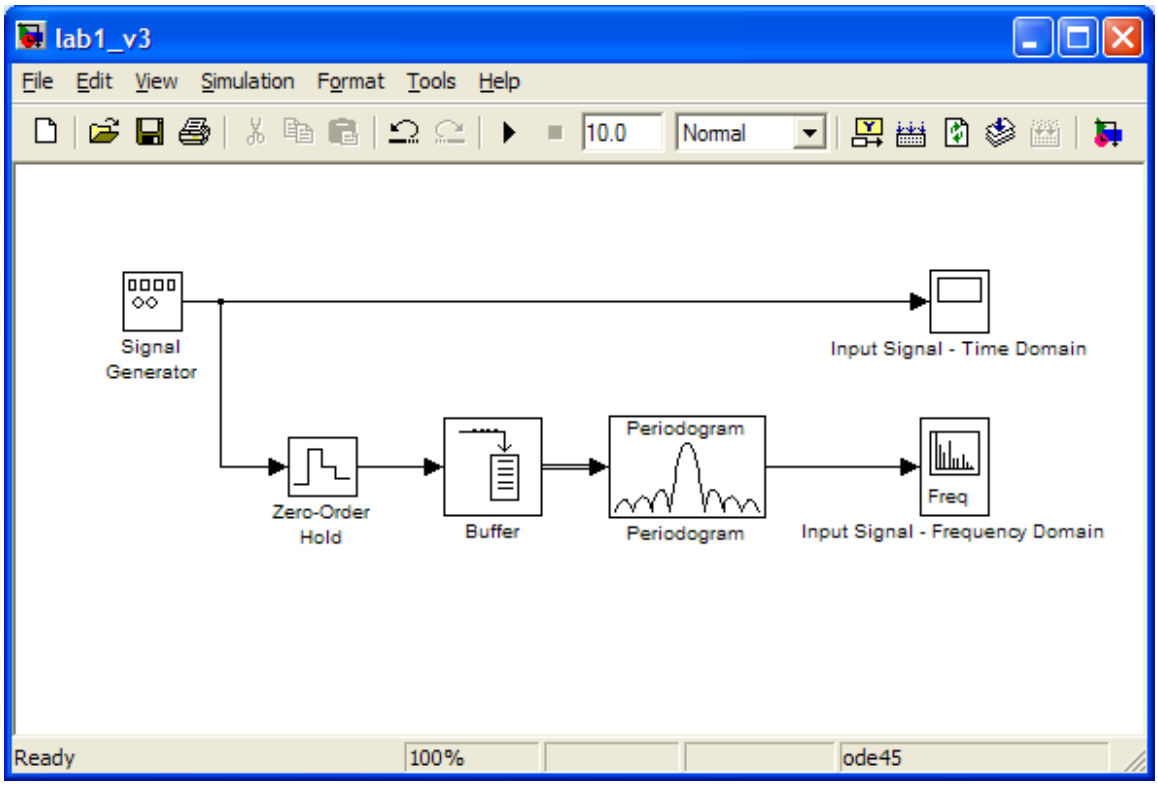

Figure 2

2. Create the software model shown in Figure 2 by dragging the appropriate components from the Simulink Library Browser into the new model window. Connect the blocks together by clicking with the left button and dragging the connector to the next block.

    - **INPUT SIGNAL**
        - **Signal Generator** is found in Simulink/Sources
    - **SIGNALS – TIME DOMAIN REPRESENTATION**
        - **Scope** is found in Simulink/Sinks. Rename the Scope to "Input Signal – Time Domain" as shown in Figure 2, by double-clicking on the name labels, then editing them.
    - **SIGNALS – FREQUENCY DOMAIN REPRESENTATION**
        - **Zero-Order Hold** is found in Simulink/Discrete
        - **Buffer** is found in Signal Processing Blockset/Signal Management/Buffers

- **Periodogram** is found in Signal Processing Blockset/Estimation/Power Spectrum Estimation
- **Vector Scope** is found in Signal Processing Blockset/Signal Processing Sinks. Rename the Vector Scopes to "Input Signal – Frequency Domain" as shown in Figure 2, by double-clicking on the name labels, then editing them.

3. Configure the various blocks by double-clicking on each and setting the parameters as follows:

- **Signal Generator** - ensure that the Signal Generator gives a 0.5 Hz sinusoid with amplitude of 1.
- **Scope** – no changes required (defaults OK).
- **Zero-Order Hold** - ensure a sample time of 0.02
- **Buffer** - ensure an output buffer size of 512, buffer overlap of 511, and zero initial conditions
- **Periodogram** – ensure a Hamming window, symmetric window sampling, 512 FFT length, and 8 spectral averages
- **Vector Scope** – ensure that the Scope properties use "Input domain: Frequency", and the Axis properties are as shown in Figure 3. For Display and Line properties, keep the default settings. Please note that the output spectrum is displayed in dB.

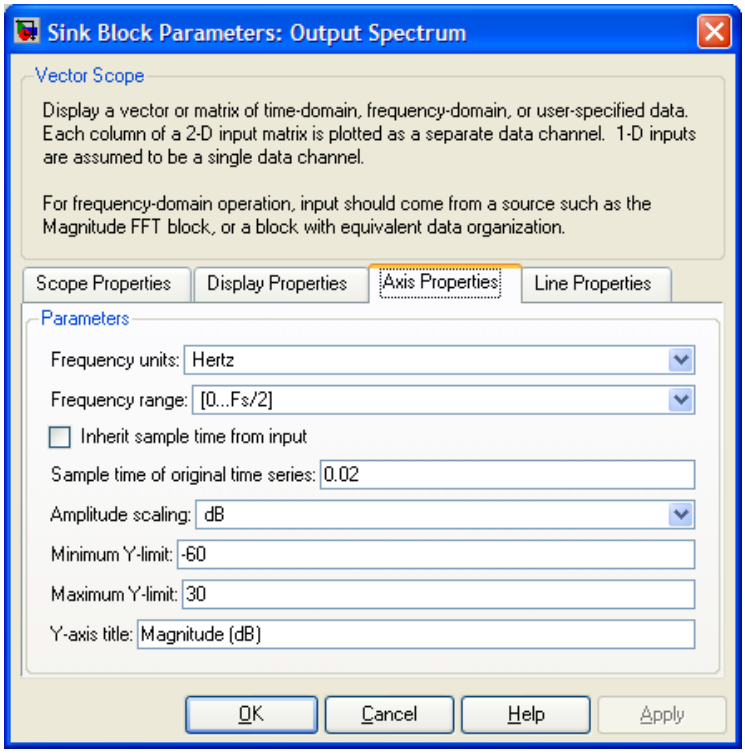

Figure 3

2.1.2 Running the Simulink Model

1. Run the Simulink model by pressing the ▶ toolbar button.
2. Observe and save the "Input Signal – Time Domain" (scope output) and "Input Signal – Frequency Domain" (vector scope output) in the accompanying "answer sheet" document. To save these plots, select the window with the plot you want to copy, press-and-hold the "Alt" key and then "Print Screen" key (this copies the window to the Clipboard), and then select the "Paste" option in the "answer sheet" document.
3. Repeat steps 1 and 2 using an input sinusoid frequency of 5Hz.
4. Repeat steps 1 and 2 using an input sinusoid frequency of 5Hz, with amplitude of 2 (instead of 1).
5. Save your model as "sysc3501_swlab1_1.mdl".

2.1.3 Build the Simulink model – Two Sinusoids

In this section, we consider an input signal consisting of a sum of pure sinusoids (of differing frequency):

$$x(t) = A_0 \cos 2\pi f_0 t + A_1 \cos 2\pi f_1 t$$

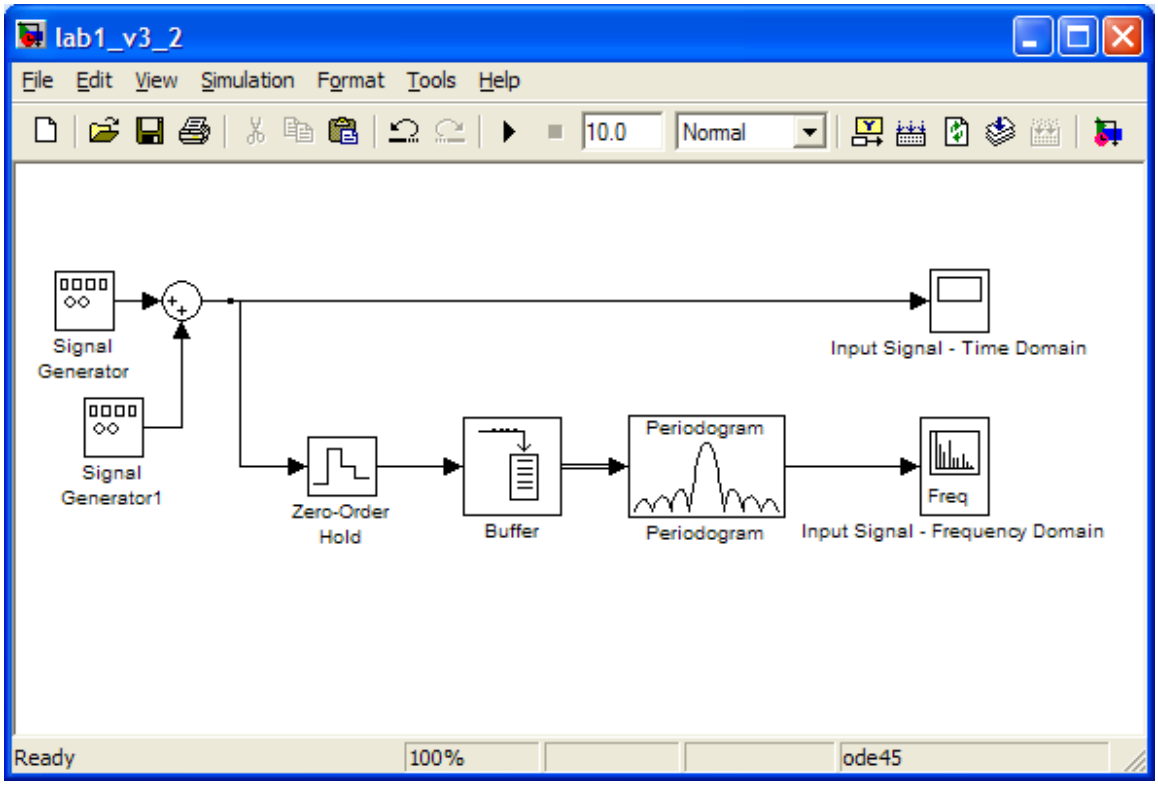

Figure 4

1. Create the software model shown in Figure 4 by dragging in:
    - Another **Signal Generator** block (in Simulink/Sources)
    - A **Sum** block (in Simulink/Math Operations)
2. Configure the Signal Generator blocks as 0.5 Hz and 5Hz sinusoids (with amplitudes of 2 and 2, respectively), i.e. $A_0 = A_1 = 2$, $f_0 = 0.5\,\text{Hz}$, $f_1 = 5\,\text{Hz}$.
3. Run the Simulink model by pressing the ▶ toolbar button. Observe and save the "Input Signal – Time Domain" (scope output) and "Input Signal – Frequency Domain" (vector scope output).
4. Change f_1 to 3Hz, re-run the simulation, and save the "Input Signal – Time Domain" (scope output) and "Input Signal – Frequency Domain" (vector scope output) plots.
5. Change A_0 to 5, re-run the simulation, and save the "Input Signal – Time Domain" (scope output) and "Input Signal – Frequency Domain" (vector scope output) plots.
6. Configure the Signal Generator blocks as 5 Hz and 5.5Hz sinusoids (each with amplitude of 1) and re-run the simulation. Observe and save the "Input Signal – Frequency Domain" (vector scope output) plot. Repeat for frequencies of 5Hz and 5.1 Hz, then for frequencies 5Hz and 5.01Hz.
7. Save your model as "sysc3501_swlab1_2.mdl".

2.2 Square Wave input

As we have seen, frequency domain representations of sinusoidal inputs are particularly simple. It is possible to analyze more complicated inputs, and in this part, we consider a square wave, as shown in Figure 5.

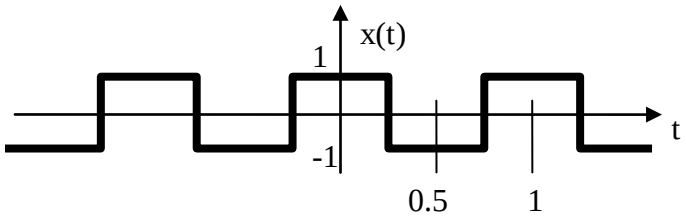

Figure 5

Like the sinusoid considered previously, the square wave $x(t)$ is a periodic input with a fundamental frequency f_0. (Given the sketch of the square wave in Figure 5, what is the value of f_0?)

The key difference between a sinusoid and a square wave (from a frequency domain perspective), is that the square wave does not only consist of a single frequency f_0. Instead, the square wave consists of various harmonics, located at integer multiples of f_0.

The Fourier transform of the square wave $x(t)$ is:

$$X(f) = \sum_{odd\,k} \frac{4}{k\pi} \sin\frac{k\pi}{2} [\delta(f - kf_o)]$$

Notice that in the case of a square wave input, it is the frequencies at the ODD harmonics (*i.e.* $f = f_0, 3\,f_0, 5\,f_0, 7\,f_0$, etc.) that appear in the frequency spectrum.

2.2.1 Build the Simulink model – Square Wave Input

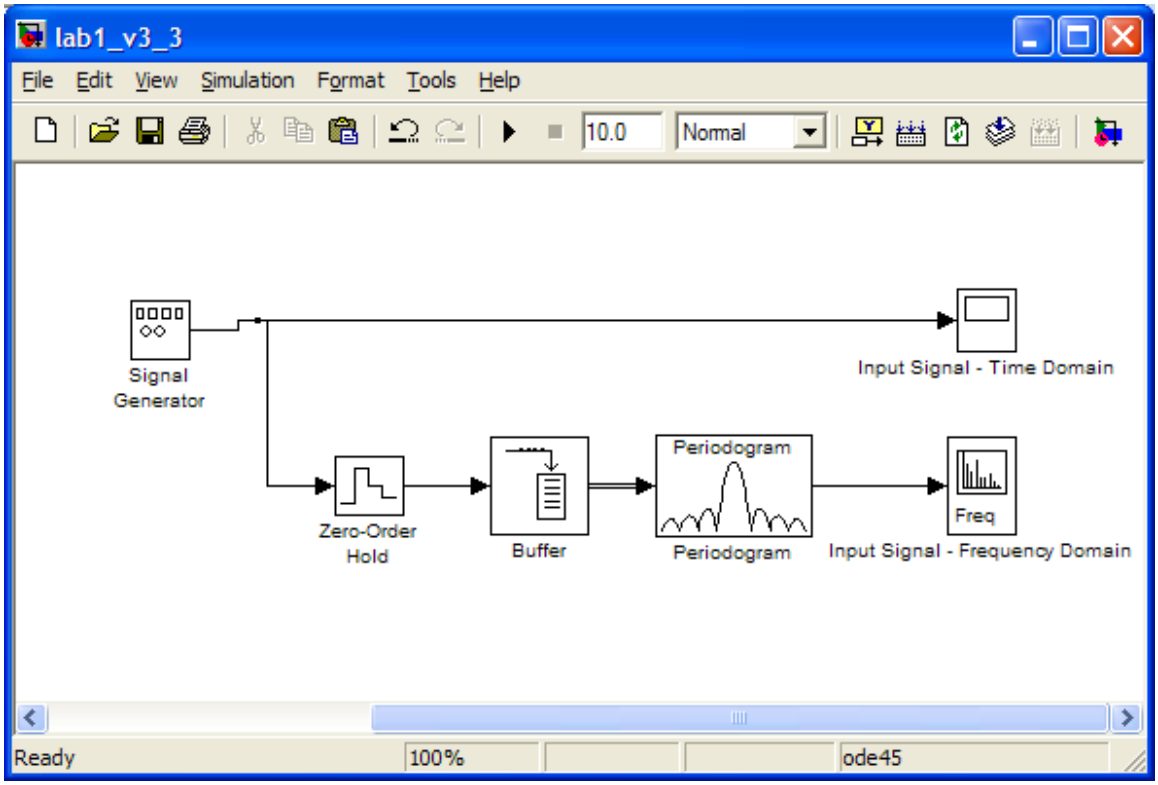

Figure 6

1. Create the software model shown in Figure 6. The square wave input is specified in the Signal Generator block properties (choose Wave Form= "Square" instead of "Sine". Choose a frequency of 5 Hz, and amplitude of 1.
2. Run the Simulink model by pressing the ▶ toolbar button. Observe and save the "Input Signal – Time Domain" (scope output) and "Input Signal – Frequency Domain" (vector scope output).

3. Repeat the above steps for a sawtooth input signal of the same frequency and amplitude (choose Wave Form= "Sawtooth" in the Signal Generator block) and re-run the simulation.
4. Save your model as "sysc3501_swlab1_3.mdl".

The examples above demonstrate that the frequency representation of periodic time domain signals with the same fundamental frequency (in the cases above, f_0=5Hz) can be very different, based on their shape (sine, square, sawtooth,etc.).

This observation can be related to the field of music – an orchestra, for example, tunes to a specific note (called "concert A") which has a fundamental frequency of f_0=440Hz. Several different musical instruments (e.g. trumpet, flute) can generate this same note, but obviously, each instrument will have its own characteristic sound. What makes the notes from different instruments sound unique, if they have the same fundamental frequency (f_0=440Hz)?

If you recorded those sounds with a microphone and analyzed the resulting signal in the frequency domain, you would discover that it is the relative magnitudes of the *harmonics* of f_0 that give each instrument its character. The sine, square, and sawtooth waves that you used above can be thought of as different "instruments" with the same fundamental frequency, but very different harmonic signatures. If we could listen to them, the sine wave would sound very pure, while the other two would sound much harsher. (Of course, any frequency less than about 20Hz (such as the 5Hz we used) can not be heard by humans).

2.3 Non-periodic input

Thus far, we have only considered periodic input signals. The tools we have used, though, are not limited to these types of inputs. In this section, we will consider a non-periodic, time-limited signal x(t) of the form shown in Figure 7:

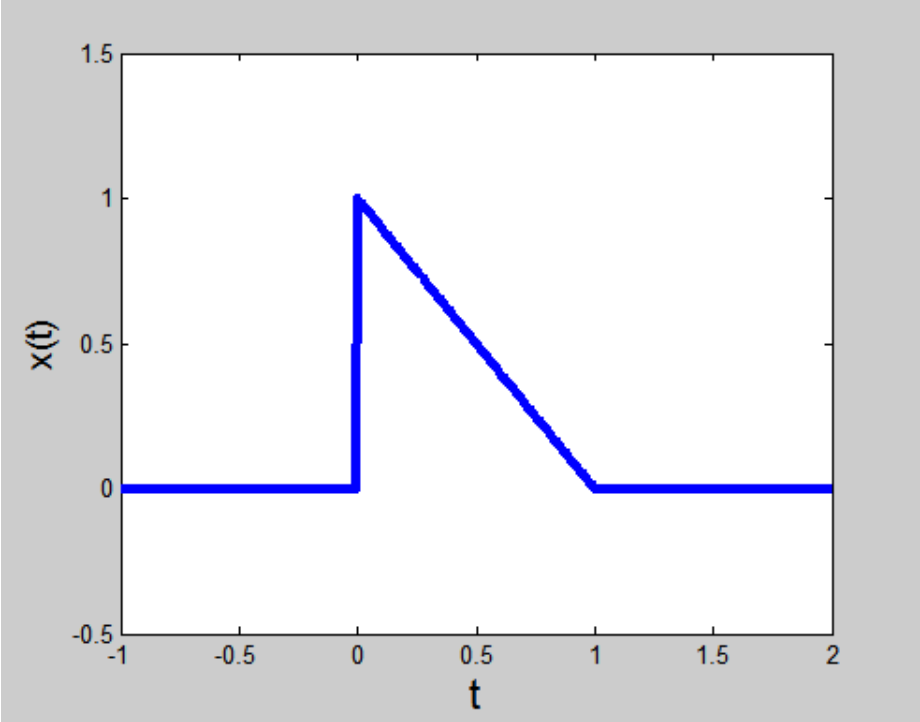

Figure 7

2.3.1 Build the Simulink model – Non-periodic Input

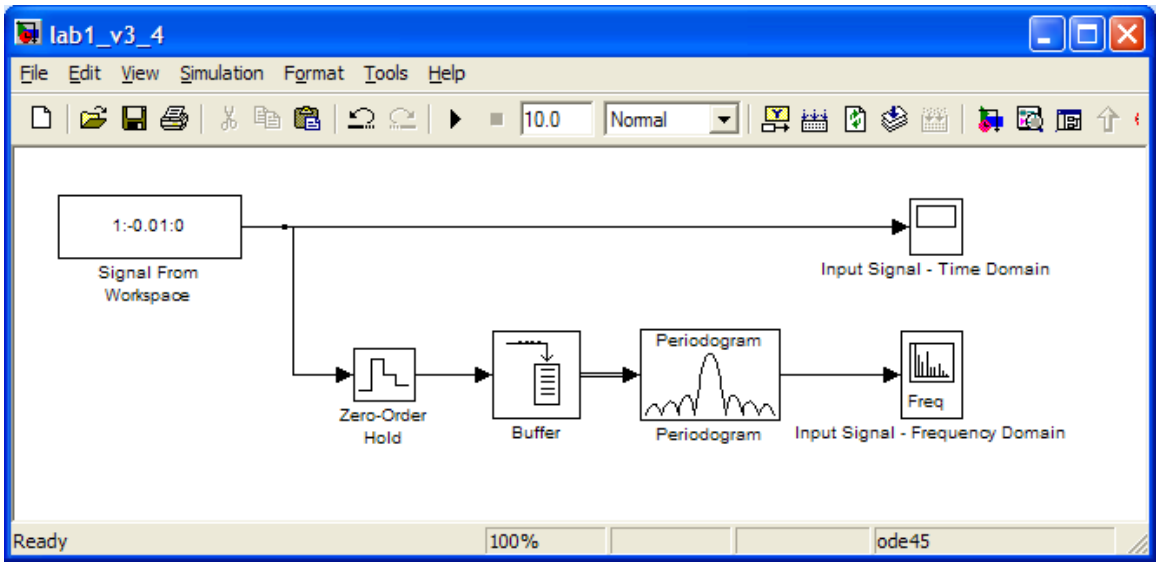

Figure 8

1. Create the software model shown in Figure 8. You will need the following block:

    a. **Signal from Workspace** is found in Simulink/Signal Processing Blockset/ Signal Processing Sources.

    Using this block, the input signal is defined as a simple Matlab vector in the block properties.

    Double-click on the block, then set Signal to "1:-0.01:0" and Sample Time to "0.01".

2. Run the Simulink model by pressing the ▶ toolbar button. Observe and save the "Input Signal – Time Domain" (scope output) and "Input Signal – Frequency Domain" (vector scope output).
3. Save your model as "sysc3501_swlab1_4.mdl".

You will remember that periodic signals will give Fourier spectra that consist of delta functions (occurring at multiples of the fundamental frequency). The sharp peaks we saw in the previous periodic signals can be thought of as discrete approximations to these impulses. In the case of the non-periodic triangle-shaped waveform above, the concept of fundamental frequency does not hold. In the frequency representation of this signal, the energy is "smeared" over a wide range of frequencies.