Lab 6

Sampling an Analog Signal

6.1 Introduction

The sampling of an analog signal is done in order to convert the analog signal into a digital representation of the signal. The sampling of analog signal is the first step in performing any digital processes to the signal.

Sampling is performed by multiplying an analog signal with a comb function, which ideally consists of impulses at specific intervals. The impulses are separated by time T_s and thus the frequency of the impulses is $f_s = 1/T_s$. The output of a sampler is a digitized signal consisting of individual samples of the analog signal at discrete time intervals. The question that arises is what are the limits of f_s? The answer is given by the Nyquist criterion which denotes that the sampling frequency must be at least twice the maximum frequency present in the analog signal. A value of f_s lower than this will contribute to aliasing, where we cannot get the original signal back from the digitized signal.

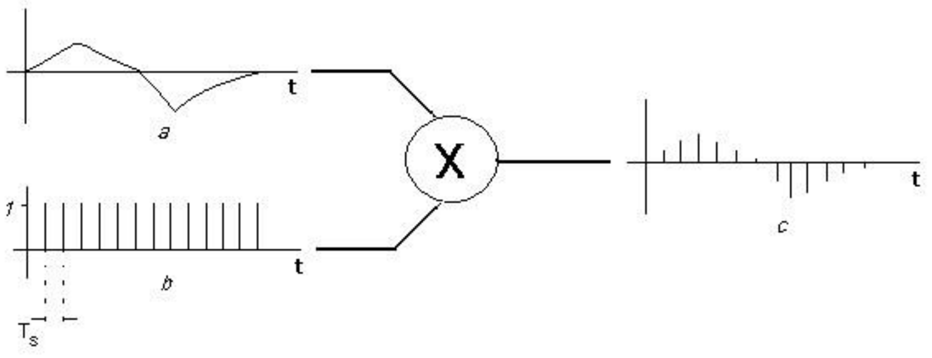


Figure 6.1: This figure shows how sampling is done. (a) An analog signal is multiplied with a comb signal (b) of magnitude one to obtain its sampled (digitized) version (c)

Unfortunately, MATLAB cannot work with analog signals, because after all, MATLAB is run by a digital machine! But we can represent analog and digital signals separately.

In MATLAB, we can generate a random sequence between 0 and 1 using the function *rand*(). To generate a random sequence between 0 and 1 with mean 0, variance 1 and standard deviation 1, we use the *randn*() function. This sequence is now called normally distributed random numbers.

In MATLAB, we can view a signal and its properties using the sptool. For this purpose, type sptool in MATLAB. The Fig. 6.2 will appear.

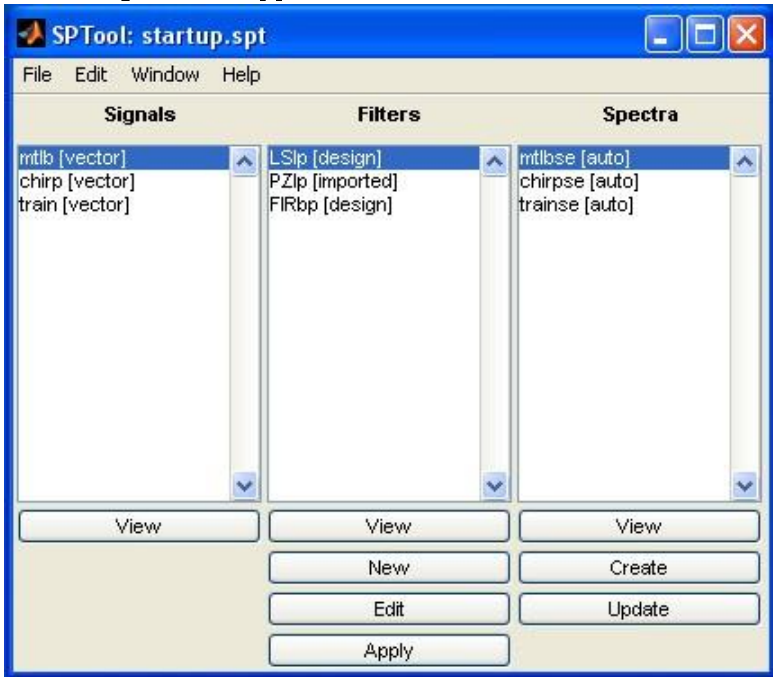


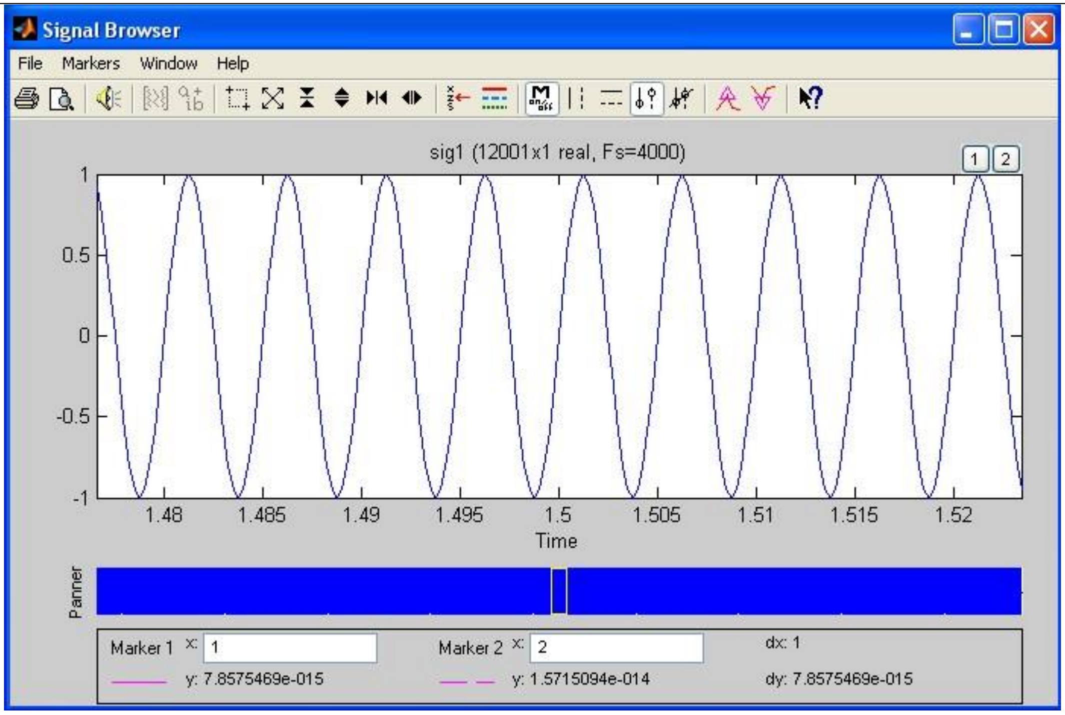
Figure 6.2: The sptool GUI

We see that this GUI has many capabilities. But for this lab, we will be focusing on only viewing the signals. The signals can be imported to the sptool via clicking File and then Import. Then we can view the signal by clicking the view button at the end of Signal window. A window as shown in Fig. 6.3 will appear.

We can zoom in and out and perform other functions using this Signal Browser.

6.2 Practical

Generate various waveforms like sinusoidal, triangular, damped oscillations and unit step using different sampling frequencies. Plot the waveforms on time axis. Increase and decrease the sampling frequency and note the effect on the representation of waveforms in digital domain.



Marker 1 x: 1
y: 7.8575469e-015
Marker 2 x: 2
y: 1.5715094e-014
dx: 1
dy: 7.8575469e-015

Figure 6.3: The figure obtained via sptool

6.3 Hints

Represent the analog waveform using the plot command and the digitized waveform using stem command. Remember that total number of samples N equals total time of the analog signal T_o multiplied with the sampling frequency f_s. i.e.

$$N = T_o f_s \tag{6.1}$$

6.4 Pseudocode

This section discusses the coding technique that can be utilized for performing the above lab.

Algorithm 6.1: Sample Generation

```
// Generate the signal, either sinusoid, square or rectangular
for $T_s$ = $NyquistSamplingRate$ to $min$ do
    $t = 0 : T_s : NT_s$
    $f(t)$
    $Plot(f(t))$
end for
```

6.5 Questions

1. Does the increase in sampling frequency fs makes the digitized waveform a better representation of the analog waveform? If so, why in practice we keep fs just above twice the maximum frequency?
2. Obtain the above results using your own script file. Generate analog and comb functions separately. Multiply the generated analog signal with the comb function, and display the results.
3. Using MATLAB, generate a sinusoid and then simulate the effect of noise by adding normally distributed random numbers. Save the sinusoid and the noisy output to *.mat files. View the sinusoid and the noisy output via sptool. Hear the two signals and get a grasp of the difference.
4. Using a sinusoid of 1KHz, sampled at 48KHz, generate 10KHz sinusoid form this signal in MATLAB. Hint: you need to ignore some of the samples!