D Verification of Sampling Theorem in Time Domain

Objectives

1. To understand the concept of sampling in digital signal processing.
2. To explore the effects of under-sampling, Nyquist sampling, and over-sampling.
3. To reconstruct the original continuous signal from discrete samples using MATLAB.

Apparatus

Software: MATLAB R2022b

Hardware: Personal Computer

Theory

In digital signal processing (DSP), sampling refers to process of converting a continuous-time (analog) signal into a discrete-time signal by capturing the signal's amplitude values at regular intervals. Sampling is fundamental to DSP applications such as audio, image, and communication systems, where continuous signals are converted into discrete samples for efficient digital use. The sampling rate, also known as the sampling frequency (F_s), is the rate at which these samples are collected; it is commonly expressed in samples per second (Hz).

- Discrete-Time Signals:

A discrete-time signal $x[n]$, is obtained by taking samples of a continuous analog signal $x_a(T)$ at regular time intervals (sampling period) T. The relationship between the continuous-time and discrete-time signals is given by:

$$x[n] = x_a(nT) \text{ where } -\infty < n < \infty$$

Here, n represents the sample index. The reciprocal of T is known as the sampling rate or sampling frequency ($F_s = 1/T$).

- Sampling Theorem (Nyquist Theorem):

The Nyquist Sampling Theorem states that for a continuous signal to be perfectly reconstructed from its samples, the sampling frequency must be at least twice the highest frequency present in the signal. This minimum sampling frequency is called the Nyquist rate and is given by:

$$F_s \geq 2F_{max}$$

where F_{max} is the highest frequency component of the original signal. Sampling below this rate leads to aliasing, where different signals become indistinguishable from each other.

- Aliasing

  Aliasing occurs when a continuous signal is sampled at a frequency below the Nyquist rate. This effect causes higher frequencies in the signal to appear as lower frequencies in the sampled data, leading to distortions. Due to aliasing, different frequencies in the original signal become indistinguishable from each other in the sampled signal, which makes it impossible to accurately reconstruct the original signal. Understanding aliasing is crucial in DSP, as it directly impacts signal accuracy in applications like audio and image processing.

- Under-sampling and Over-sampling:

  Under-sampling occurs when the sampling frequency $F_s < 2F_{max}$, which leads to aliasing and distortion of the signal. In contrast, over-sampling takes place when $F_s > 2F_{max}$, resulting in a larger number of samples. This can enhance signal reconstruction and reduce noise, although it may increase computational costs.

MATLAB Code Overview

1. Continuous-Time Signal Generation
   A continuous-time signal can be represented in various forms. In this instance, it is modeled as a sine wave with defined amplitude (A) and frequency (F). The sine wave is chosen for its fundamental role in signal analysis and the ability to illustrate continuous-time behavior effectively.

   MATLAB code for generating the Continuous-Time Signal is,

```
% Amplitude and Frequency
A = 1; F = 2;

%Time vector for continuous signal
t = 0:0.001:1;

% Continuous-time sine wave
% x_a = A*sin(2*pi*F*t);
```

2. Discrete-Time Signal Generation (Nyquist Sampling)
   The discrete-time signal, sampled at the Nyquist rate, captures specific points from a continuous sine wave. It is represented by discrete values plotted at regular intervals, reflecting the sampled nature of the original continuous signal.

   MATLAB code for generating the Discrete-Time Signal is,

```
% Sampling at Nyquist rate
F_s = 2*F;

% Sampling points
n = 0:1/F_s:1;

% Discrete-time signal
x_s = A*sin(2*pi*F*n);
```

3. Reconstruction using Interpolation
   Signal reconstruction involves converting a sampled discrete-time signal back into a continuous-time analog signal. This process is important in DSP because, after processing a digital signal, we often need to convert it back to its original analog form for playback, transmission, or other applications. The original signal can be approximated and plotted using interpolation methods, such as linear and spline interpolation in MATLAB, to illustrate how well the discrete samples map back to a smooth waveform.

   MATLAB code for Reconstruction using Interpolation is,

```
% Time vector for reconstruction
t_r = linspace(0, 1, 1000);

% Reconstruct signal using linear interpolation
x_linear = interp1(n, x_s, t_r, 'linear');

% Reconstruct signal using spline interpolation
x_spline = interp1(n, x_s, t_r, 'spline');
```

Procedure

1. Open MATLAB and create a new script by navigating to **Home → New → Script**.

2. Implement the MATLAB code in the script to generate a continuous-time sin wave, setting amplitude (A = 1), frequency (F= 2 Hz), and time step (0 to 1 sec). Use the 'plot' function to visualize the subplot of continuous-time signal, as shown in Figure 1(a).

3. Sample the continuous-time signal at the Nyquist sampling rate ($F_s = 2F$) by creating a discrete-time signal. Use the '`stem`' function to plot the sampled signal in the second subplot, as shown in Figure 1(b).

4. Reconstruct the continuous-time signal from the discrete samples using spline interpolation. Visualize the reconstructed signal using the '`plot`' function as the third subplot, as shown in Figure 1(c).

5. Save the generated figure in JPEG format by selecting **File → Save As** in the Figure window.

6. Modify the sampling frequency to represent under-sampling and oversampling operations to verify the sampling theorem, as illustrated in Figures 2 and 3.

Results

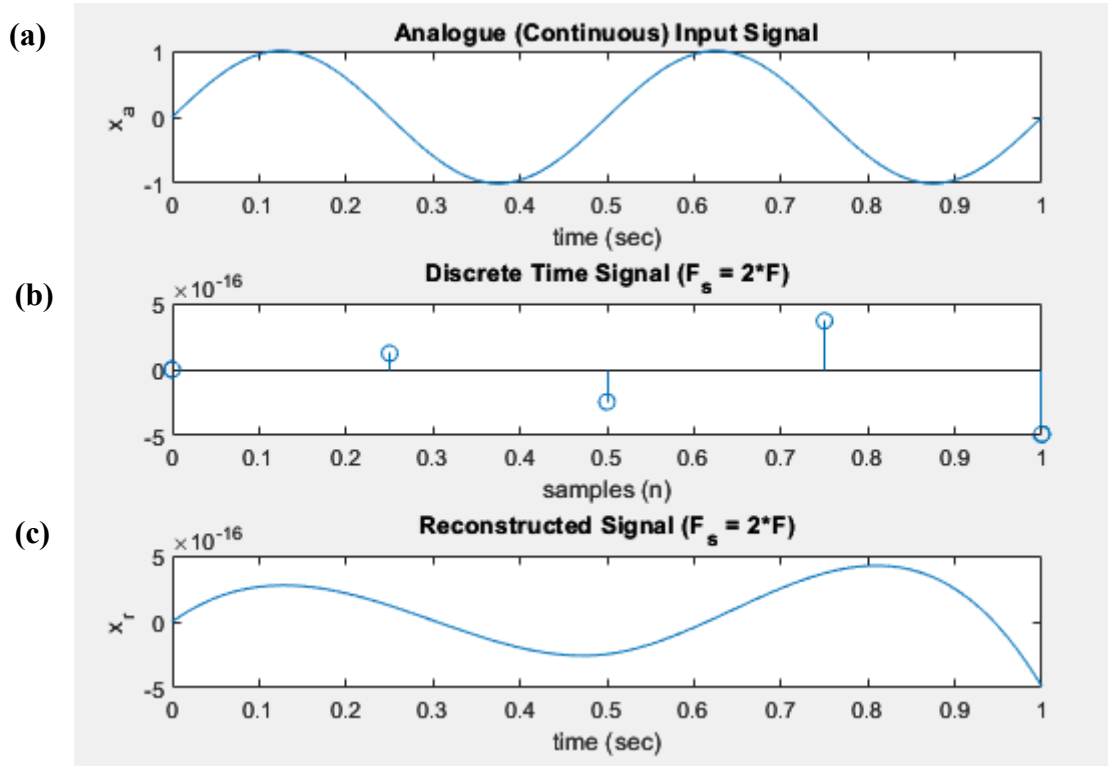

Figure 1. Sampling of the continuous-time signal at the Nyquist sampling rate.

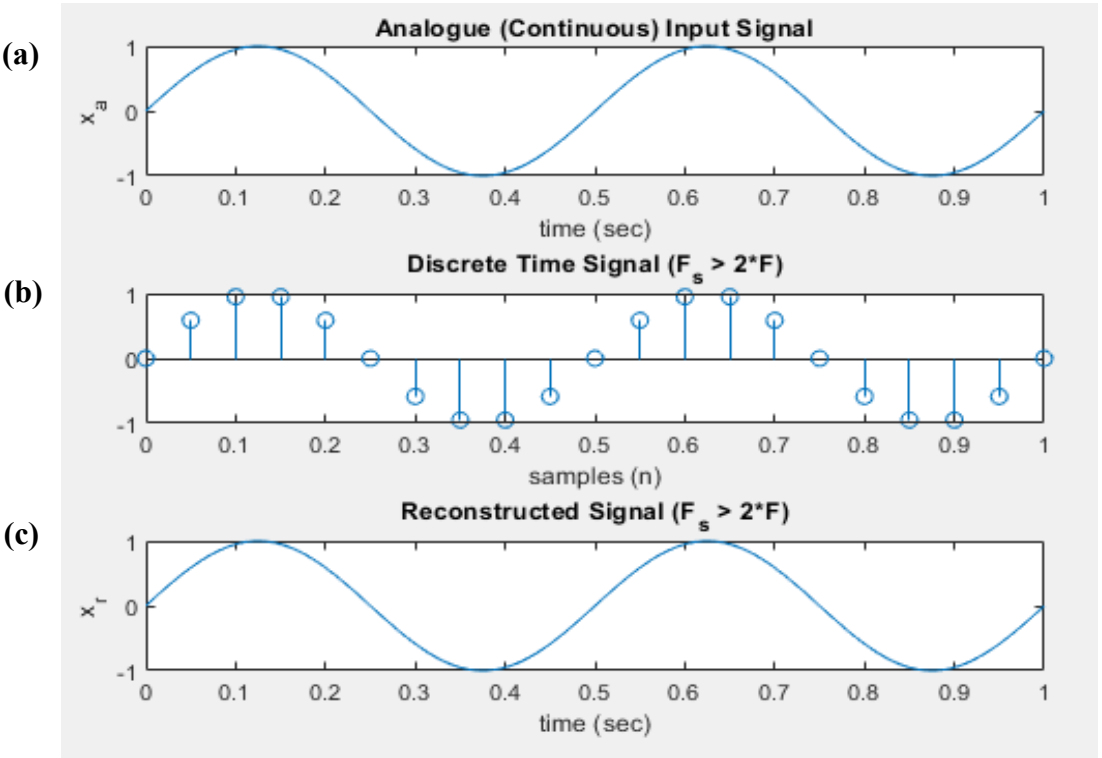

Figure 2. Sampling of the continuous-time signal above the Nyquist sampling rate.

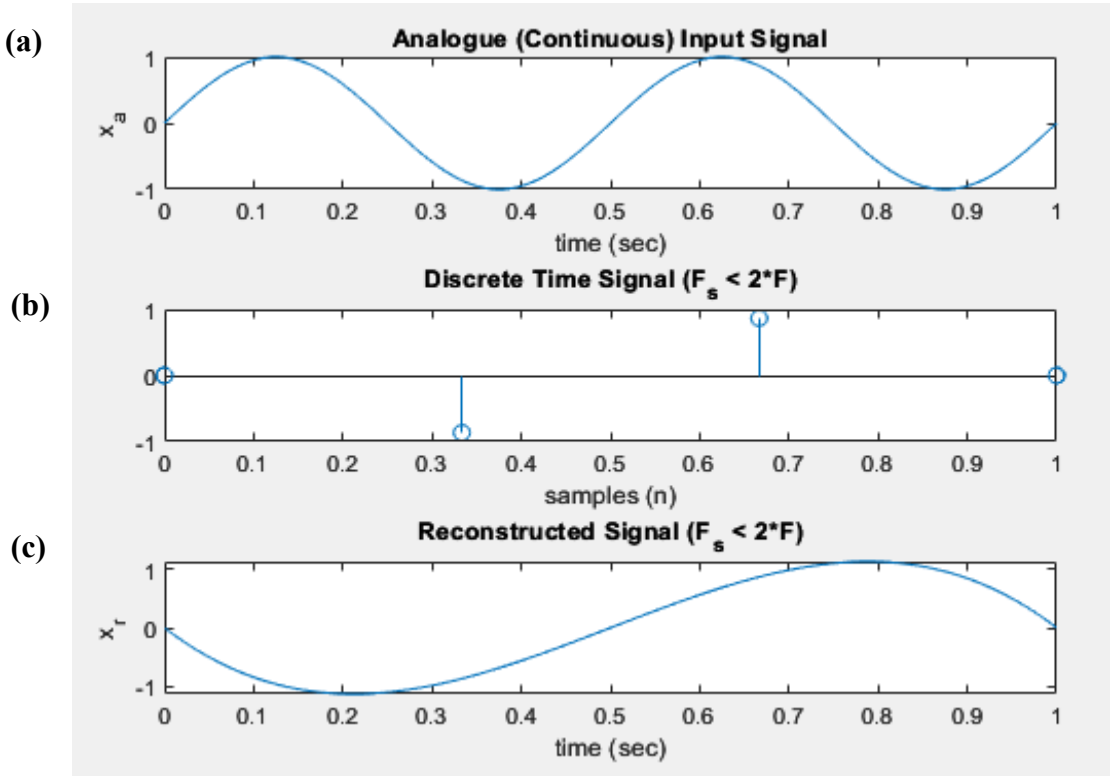

Figure 3. Sampling of the continuous-time signal below the Nyquist sampling rate.

Exercise

1. (a) Write a MATLAB program to plot two continuous-time signals, $x1(t) = \sin(25\pi t)$ and $x2(t) = \sin(25\pi(t - 0.2))$, at a sample rate of Fs = 20 Hz over the range 0 to 1 second. In the first subplot, visualize the continuous-time signals, using a solid magenta line for $x1(t)$ and a dashed green line for $x2(t)$. In the second and third subplots, plot the sampled discrete-time points of $x1(t)$ and $x2(t)$ as stem plots. Here lines should be in blue for $x1(t)$ and red for $x2(t)$, where points should be filled in yellow and blue respectfully. Observe whether the time shift between $x1(t)$ and $x2(t)$ can still be seen in the sampled signals, or if it has been obscured by aliasing and comment on your findings.

2. (a) Write a MATLAB program to verify aliasing in the sampled signals, specifically, showing that the sampled version of $x_1(t) = \sin(45\pi t)$ is aliased to $x_2(t) = \sin(05\pi t)$ when both signals are sampled at a rate of $F_s = 20$ Hz.

   Note - In your program, define the continuous-time signals, sample them at the specified rate, and create two subplots. The first subplot should visualize $x_1(t)$ using a solid green line and overlaid with the sampled discrete-time points as a magenta stem plot.
   Meanwhile the second subplot should illustrate $x_2(t)$ in a similar representation. Linewidth should be 1.5 for all.

(b) In third subplot, compare the sampled versions of $x_1(t)$ and $x_2(t)$to observe the aliasing effect. Here lines should be in green and points should be filled for $x_1(t)$ and lines should be in red for $x_2(t)$. Linewidth should be 1.5 for all. Comment on your findings regarding how the sampled signal $x_1(t)$ may appear similar to $x_2(t)$) due to the aliasing phenomenon.

3. (a) Write a MATLAB program to generate and plot the continuous-time signal $x(t) = \sin(10\pi t) + \sin(50\pi t)$ for *t* ranging from 0 to 1 second. In the first subplot, visualize this continuous-time signal using a solid magenta line. Additionally, this signal should be sampled at a frequency of 120 Hz, which satisfies the Nyquist criterion. Visualize the sampled
discrete-time points as a blue stem plot, where data points filled in yellow in the second subplot. Linewidth should be 1.5 for all.

(b) Reconstruct the continuous-time signal from the sampled points using both linear and spline interpolation. In the third subplot, plot the original continuous-time signal using a solid green line alongside the reconstructed signal obtained through linear interpolation represented with a blue dashed line. In the fourth subplot, plot the original continuous-time signal with a solid magenta line and the reconstructed signal using spline interpolation with a yellow dashed line. Linewidth should be 1.5 for all.

4. (a) Write a MATLAB program to plot the continuous-time signal $x(t) = \sin(50\pi t) + \sin(90\pi t)$ for *t* ranging from 0 to 1 second. In the first subplot, plot this signal with a solid green line. Next, determine the Nyquist sampling rate for this signal, which is the minimum rate needed for accurate reconstruction. Then, sample the signal at a rate higher than the Nyquist rate to ensure proper reconstruction. In the second subplot, plot both the continuous-time signal as a solid green line and the sampled discrete-time points as a blue stem plot for a clear comparison. Linewidth should be 1.5 for all.

(b) Reconstruct the continuous signal from the oversampled points using spline interpolation. In the third subplot, plot the original continuous signal with a solid green line and the reconstructed signal with a magenta dashed line for clear comparison between the original and reconstructed signals. Linewidth should be 1.5 for all.