Sampling and Quantization

Reconstruction of Sampled Signal

Prepared by:
ENG-219-004/2021

Lecturer:
Mr Martin Wafula

Multimedia Universirty of Kenya
Electrical Engineering

November 27, 2024

Reconstruction of Sampled Signal

1 Introduction

In digital signal processing, the process of reconstructing a continuous-time signal from its sampled versions is a critical operation. The reconstruction process ensures that the information lost during sampling is retrieved, provided that the sampling rate satisfies the conditions of the Sampling Theorem. According to the Nyquist-Shannon Sampling Theorem, a band-limited signal can be perfectly reconstructed from its samples if the sampling rate exceeds twice the maximum frequency present in the signal (the Nyquist rate). This report focuses on reconstructing a sampled signal using MATLAB.

2 Objective

The primary objective of this experiment is to:

- Demonstrate the process of reconstructing a continuous signal from its discrete samples.
- Use MATLAB to simulate and visualize the reconstruction of a sampled signal.
- Analyze how different sampling rates affect the accuracy of the reconstructed signal.

3 Theoretical Background

The reconstruction of a sampled signal relies on the ideal interpolation formula:

$$x(t) = \sum_{n=-\infty}^{\infty} x[n] \cdot \operatorname{sinc}\left(\frac{t - nT}{T}\right)$$

Where:

- $x(t)$ is the reconstructed continuous-time signal.
- $x[n]$ are the discrete samples of the signal.
- T is the sampling period, related to the sampling frequency by $f_s = \frac{1}{T}$.
- $\operatorname{sinc}(t) = \frac{\sin(\pi t)}{\pi t}$ is the sinc function, which acts as an ideal low-pass filter.

4 Methodology

1. Define Parameters and Generate Signal: Define the message signal and sampling parameters as in Experiment 1.
2. Up sample and Zero-fill the Sampled Signal: Increase the rate of the sampled signal by inserting zeros between samples.
3. Frequency Spectrum of the Sampled Signal: Calculate the FFT of the upsampled signal to observe its frequency spectrum.
4. Design a Low Pass Filter (LPF): Define the transfer function of an LPF to retain frequencies from -10 Hz to 10 Hz.

MATLAB Code

Below is the MATLAB code used for verifying the Sampling Theorem:

```
%% Copyright @ Dr Sudip Mandal
% Digital Communication Lab

%% Reconstruction from Sampled Signal

clear all;
close all;
clc;

%Define Parameters and Generate Signal
tot=1;
td=0.002;
t=0:td:tot;
L=length(t);
x=sin(2*pi*t)-sin(6*pi*t);

ts=0.02;

%Upsample and zero fill the sampled signal
Nfactor=round(ts/td);

xsm=downsample(x,Nfactor);

xsmu=upsample(xsm, Nfactor);

%Frequency Spectrum of Sampled Siganl
Lfu=length(xsmu);
Lffu=2^ceil(log2(Lfu)+1);
fmaxu=1/(2*td);
```

```
Faxisu=linspace(-fmaxu,fmaxu,Lffu);
xfftu=fftshift(fft(xsmu,Lffu));

%Plot the spectrum of the Sampled Signal
figure (1);
plot(Faxisu,abs(xfftu));
xlabel('Frequency');ylabel('Amplitude');
axis([-120 120 0 300/Nfactor]);
title('Spectrum of Sampled Signal');
grid;

%Design a Low Pass Filter
BW=10;
H_lpf=zeros(1,Lffu);
H_lpf(Lffu/2-BW:Lffu/2+BW-1)=1;

figure(2);
plot(Faxisu,H_lpf);
xlabel('Frequency');ylabel('Amplitude');
title('Transfer function of LPF');
grid;

%Filter the Sampled Signal
x_recv=Nfactor*((xfftu)).*H_lpf;

figure(3);
plot(Faxisu,abs(x_recv));
xlabel('Frequency');ylabel('Amlpitude');
axis([-120 120 0 300]);
title('Spectrum of LPF output');
grid;

%Inverse FFT for Time domain representation
x_recv1=real(ifft(fftshift(x_recv)));
x_recv2=x_recv1(1:L);

figure (4);
plot(t,x,'r',t,x_recv2,'b--','linewidth',2);
xlabel('Time');ylabel('Amplitude');
title('Original vs.Reconstructed Message Signal');
grid;
```

5 Results and Analysis

a. Signal Definition and Sampling

The code begins by defining the message signal as a combination of two sinusoids:

$$x(t) = \sin(2\pi t) - \sin(6\pi t)$$

where:

- The first component, $\sin(2\pi t)$, has a frequency of 1 Hz.
- The second component, $\sin(6\pi t)$, has a frequency of 3 Hz.

The signal is sampled with a time step $t_d = 0.002$ seconds, which means the signal is generated for a time interval of 1 second, resulting in a sampled vector x.

Next, the signal is downsampled by a factor of N_{factor}, which is determined by the ratio of the sampling period $t_s = 0.02$ seconds to the original time step t_d. This effectively reduces the sampling rate.

b. Upsampling and Zero-Filling

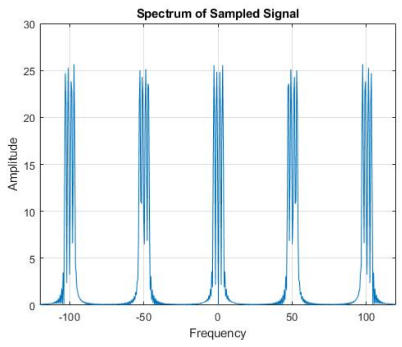
To simulate the reconstruction process, the code uses the `upsample` function to add zeros between the samples. This upsampling step does not actually recover the original signal but prepares the signal for filtering, which will help in the reconstruction process. The zero padding expands the signal to match the original sampling rate, making it suitable for filtering.

The resulting upsampled signal is stored in x_{smu} and is of a higher resolution than the downsampled signal, though the zero-padding still keeps the signal in a discrete form.

c. Frequency Spectrum of Sampled Signal

The Fast Fourier Transform (FFT) is applied to the upsampled signal x_{smu} to analyze its frequency content. The frequency resolution is improved by increasing the number of FFT points, which is done by zero-padding the signal to the next power of two. This gives a finer frequency resolution for the signal's spectrum.

The frequency axis is calculated based on the sampling period t_d, and the spectrum is shifted using `fftshift` to center the frequencies around zero.

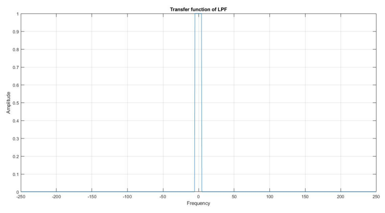


(Figure 1) shows the frequency components of the upsampled signal, where the peaks at frequencies corresponding to the original signal's components (1 Hz and 3 Hz) are visible, though **aliasing effects** may appear due to the downsampling.

d. Low Pass Filter Design

To reconstruct the original signal, a low-pass filter (LPF) is designed to remove high-frequency components (including aliasing effects) and retain the signal's desired frequency range. The bandwidth of the filter is set to BW = 10, which defines the passband of the LPF. This filter allows the low-frequency components, including the 1 Hz and 3 Hz signals, to pass through while attenuating higher frequencies.

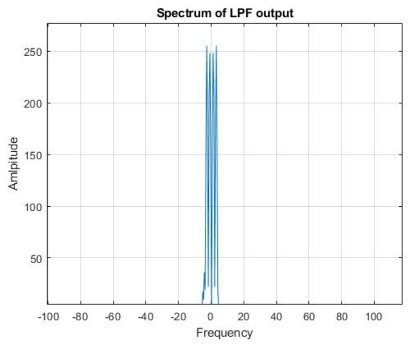
The low-pass filter is defined as a binary filter where frequencies within the passband are set to 1, and others are set to 0:



The transfer function of the LPF is plotted in Figure 2. The plot shows the frequency response of the filter, with a flat passband around zero frequency and a sharp cutoff at the edges of the passband.

e. Filtering the Sampled Signal

The upsampled signal is then filtered by multiplying the FFT of the upsampled signal by the low-pass filter's transfer function. This operation selectively passes the low-frequency components of the signal and attenuates the higher frequencies, effectively removing any **aliasing artifacts** that may have resulted from the downsampling.



The result is a filtered signal that should ideally resemble the original signal in the frequency domain. The spectrum of the filtered signal is plotted in Figure 3, showing the successful removal of high-frequency components and preserving the relevant frequency content within the passband of the LPF.

f. Inverse FFT for Time Domain Reconstruction

Finally, the inverse FFT (IFFT) is used to transform the filtered signal back into the time domain. The `ifft` function converts the frequency-domain signal back into the time-domain representation, and the result is stored in x_{recv1}. Since the signal was upsampled earlier, it is necessary to truncate the extra samples that were introduced during upsampling to match the original length of the signal. The reconstructed time-domain signal x_{recv2} is then compared with the original signal.

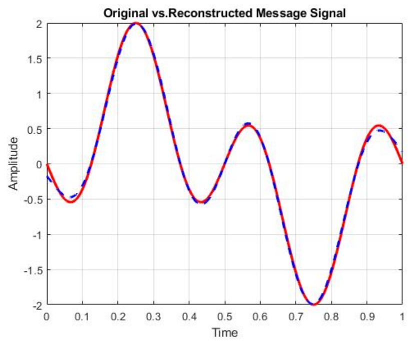


Figure 4 shows the comparison between the original and reconstructed signals. The red curve represents the original signal, while the blue dashed curve represents the reconstructed signal. The two curves closely match, demonstrating the success of the reconstruction process.

6 Conclusion

This experiment successfully demonstrates the process of reconstructing a signal from its sampled version using MATLAB. Key points to note include:

- **Sampling and Downsampling:** The original continuous signal is first sampled, and then downsampled to reduce the number of data points.
- **Upsampling and Zero Filling:** The signal is upsampled to prepare it for filtering. Zero padding between samples increases the resolution of the signal without introducing new information.
- **Low-Pass Filtering:** A low-pass filter is designed to remove high-frequency components and retain the desired frequency content. This step is crucial to mitigate aliasing and recover the original signal.
- **Reconstruction:** The inverse FFT is used to convert the filtered frequency-domain signal back into the time domain, producing a reconstructed signal that closely matches the original.

The results confirm that with proper sampling, upsampling, filtering, and inverse FFT, a sampled signal can be successfully reconstructed. This process is fundamental in **digital communications**, where accurate signal reconstruction is necessary for reliable transmission and processing of di

7 References

- Proakis, J.G., Salehi, M. (2007). Digital Communications. McGraw-Hill.
- Oppenheim, A.V., Schafer, R.W. (2010). Discrete-Time Signal Processing. Pearson.
- Shannon, C.E. (1949). Communication in the presence of noise. Proceedings of the IRE, 37(1), 10-21.