Digital Signal Processing Lab

Experiment 1 - Sampling

Utkarsh Patel (18EC30048)

Objective

- Sampling a sinusoidal waveform above Nyquist rate and observing the effect of changing window length while computing DFT
- Sampling a sinusoidal waveform below Nyquist rate and observing effect of aliasing
- Analysing spectrum of a square wave
- Implementing zero-interpolation or up-sampling for resampling a sequence

Theory

- To compute the spectrum of an analog signal numerically, the sampled waveform has to be truncated to apply DFT. This truncation or rectangular windowing in time domain causes spectrum spreading. The more the width of the window is chosen the less is spreading. Note also that the DFT involves a sampling in frequency domain.
- To check the sampling theorem, we sampled the wave at different frequencies and observed that for all frequencies <= 2 x Fmax the spectrum is not proper. Either we find aliasing in the spectrum or we find some component missing.
- Theoretically a square wave has infinite frequency components with amplitudes decreasing with increasing frequency.
- Upsampling is the process of inserting zero-valued samples between
- original samples to increase the sampling rate. (This is called “zero stuffing”). Upsampling adds to the original signal undesired spectral images which are centered on multiples of the original sampling rate.
- The primary reason to interpolate is simply to increase the sampling rate at the output of one system so that another system operating at a higher sampling rate can input the signal.
- If an analog signal is sampled at a frequency higher than the Nyquist rate it is possible to interpolate the intermediate L-1 samples or in other words obtain samples at Fs2=LFs1 frequency. This can be done by passing the sampled signals through an ideal lowpass filter of cutoff frequency Fmax and sampling it again at higher rate but in discrete domain we can do this by inserting zeroes between the samples and then passing it through a lowpass filter.

Procedure

PART A

- Choose an analog waveform
$$x(t) = 10\cos(2\pi \times 10^3 t) + 6\cos(2\pi \times 2 \times 10^3 t) + 2\cos(2\pi \times 4 \times 10^3 t)$$
- Sample it at Fs = 12kHz
- Obtain DFT of x(t) with N= 64, 128, 256 points

PART B

- Repeat the experiment with sampling frequencies: 4kHz,5kHz and 8kHz.
- Find out from the spectrum, what are the aliases of the original frequencies present in x(t) when the sampling rate is below the Nyquist rate.

PART C

- Take a square wave with time period T=1ms (F=1kHz).
- Sample at 20kHz.

- Obtain DFT of sampled square wave with 256 points.

PART D

- Take a lowpass signal of bandwidth 6kHz.
- Sample it at Fs1=12kHz.
- Insert one zero in between every 2 samples.
- Pass it through a lowpass filter of cutoff frequency 6kHz.
- Plot the output of the lowpass filter and compare it with the original signal sampled at Fs=24kHz.

Observations

PART A: Sampling a sinusoidal waveform

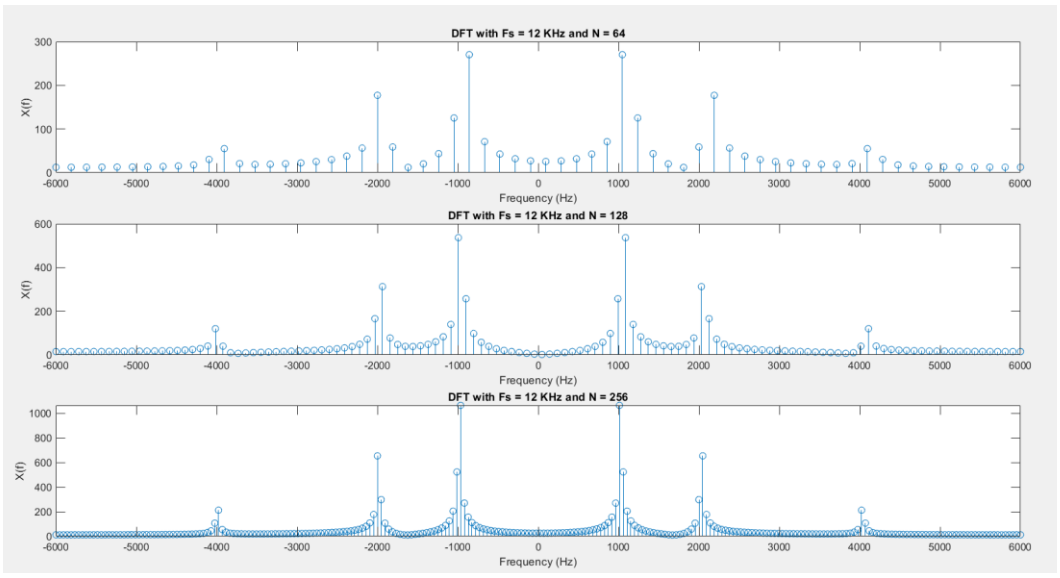

Fig. 1. Sampling the given signal at 12 KHz with different window length (above Nyquist rate)

PART B: Sampling at below Nyquist rate and effect of aliasing

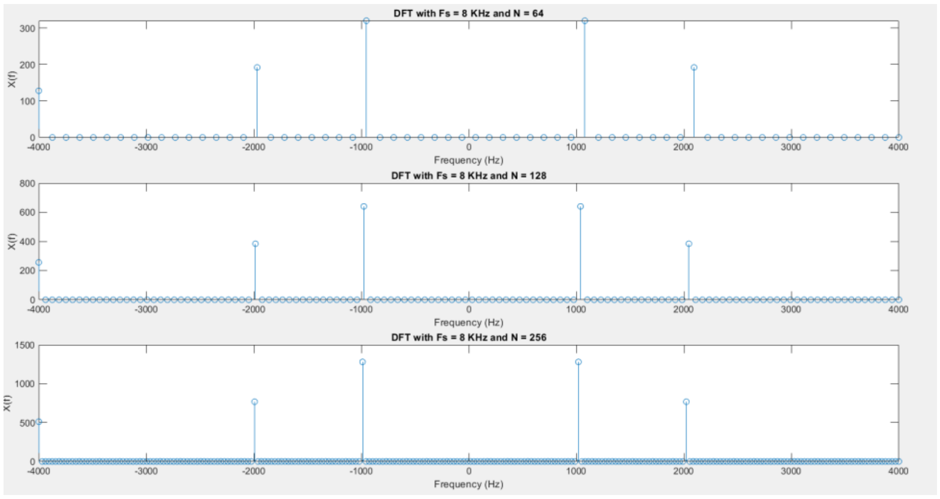

Fig. 2. Sampling the given signal at 8 KHz with different window length (below Nyquist rate)

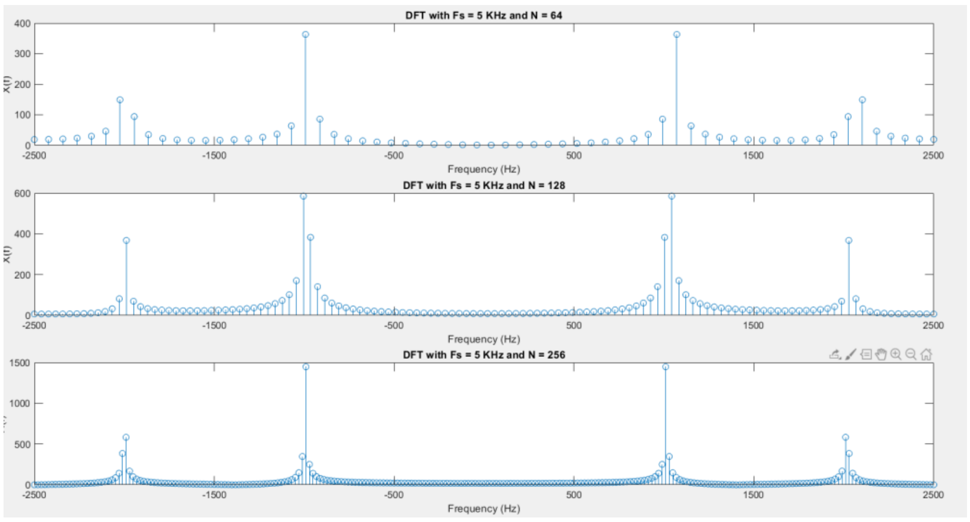

Fig. 3. Sampling the given signal at 5 KHz with different window length (below Nyquist rate)

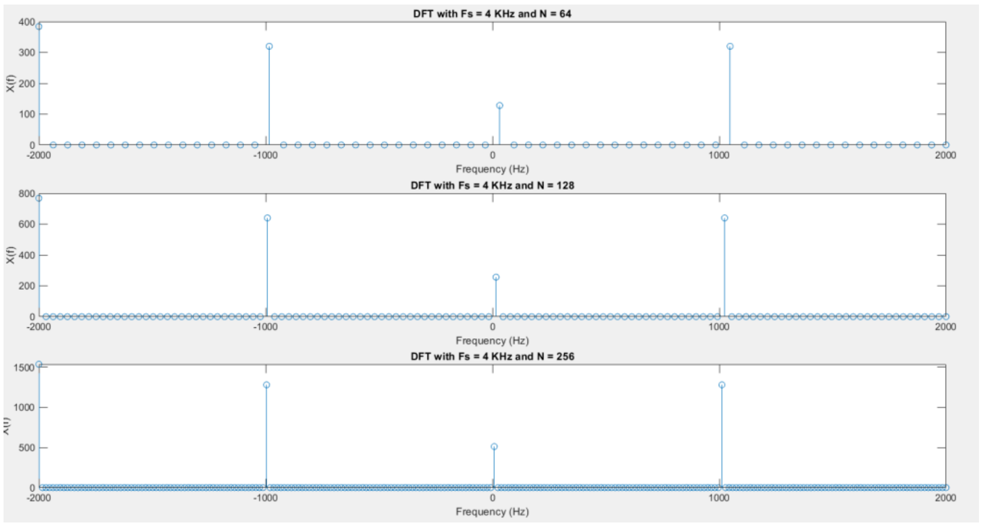

Fig. 4. Sampling the given signal at 4 KHz with different window length (below Nyquist rate)

PART C: Spectrum of square wave

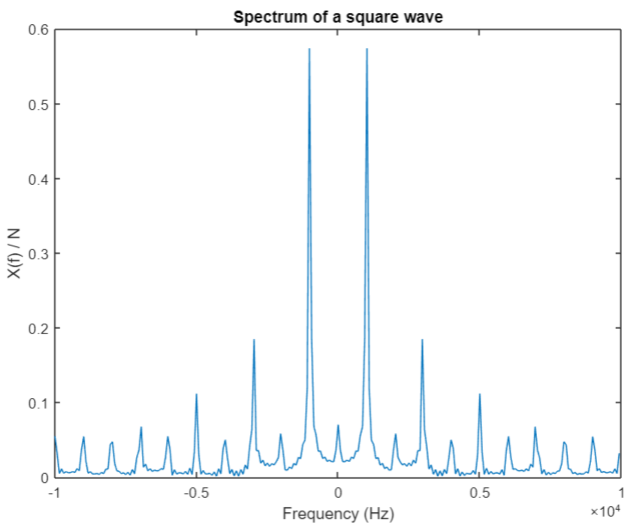

Fig. 5. Spectrum of a square wave

PART D: Up-sampling

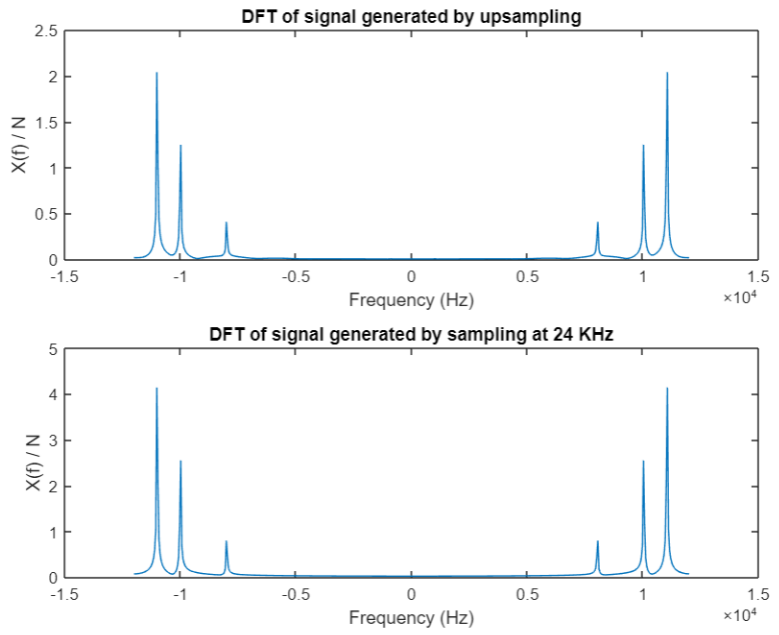

Fig. 6. Frequency-domain representation of signal

Discussion

- In PART A, the signal used is:
$$x(t) = 10\cos(2\pi \times 10^3 t) + 6\cos(2\pi \times 2 \times 10^3 t) + 2\cos(2\pi \times 4 \times 10^3 t)$$
Therefore, the maximum frequency component present in the signal corresponds to 4 KHz. As per sampling theorem, this signal must be sampled at sampling rate greater than 8 KHz. For this experiment, it was sampled at 12 KHz. The window of convolution while calculating DFT is varied as N = 64, 128, and 256. As per the theory, rectangular windowing in time domain causes spectrum spreading. The more the width of the window is chosen the less is the spreading. The same is observed while simulation. For N = 64, the spectrum is heavily spread across the frequencies present in the signal, while for N = 256, this spread is minimum.

- In PART B, the above signal is sampled below the Nyquist rate. The sampling frequency is varied as 8 KHz, 5 KHz and 4 KHz. For any given sampling frequency, in case of lowpass signals, the frequencies greater than half of the sampling frequency are lost while sampling. It can be seen that for sampling frequency of 8 KHz, the 4 KHz component present in the signal is lost and for sampling frequency, the 2 KHz component is lost as well. However, the spread of the spectrum reduces as the width of window (for calculating convolution) increases as observed in PART A.

- PART C deals with spectrum analysis of a square wave. It is known that any square wave with period T can be decomposed as sum of sinusoidal as:
$$x(t) = \frac{4}{\pi} \sum_{n=1,3,5\ldots} \frac{1}{n} \sin\left(\frac{2n\pi t}{T}\right)$$
Therefore, as per this theory, if the spectrum of any square wave is analysed, it will contain only odd integral frequencies and the contribution of each frequency decreases as frequency is increased. The same is observed in simulation. Amplitude of frequency-domain representation decreases as we move further away from origin and only odd integral frequencies are present.

- PART D deals with resampling an already sampled signal with higher harmonics of the previously used sampling frequency. This means that if we have already sampled any lowpass signal at, say, 12 KHz, then this can be resampled at the sampling rate of 24 KHz without actually sampling it at 24 KHz. For this, zero embedding is done on the sampled sequence as per the order of harmonic and this modified sequence is then low-passed at 24 KHz. It can be proved that the sequence obtained after low-passing and the sequence obtained after actually sampling at 24 KHz are one and the same. From Figure 6, it can be observed that frequency spectrum of these two sequences is identical.

APPENDIX

MATLAB code

Code for Figure 1

```
% Author: Utkarsh Patel (18EC30048)

% Experiment - 1 Part A
% DFT of the given signal with sampling frequency of 12 KHz

Fs = 12000; % sampling frequency

% for N = 64
[f, y] = foo(Fs, 64);
subplot(3, 1, 1);
stem(f, y);
xticks(-Fs / 2 : 1000 : Fs / 2);
title('DFT with Fs = 12 KHz and N = 64');
xlabel('Frequency (Hz)');
ylabel('X(f)')

% for N = 128
[f, y] = foo(Fs, 128);
subplot(3, 1, 2);
stem(f, y);
xticks(-Fs / 2 : 1000 : Fs / 2);
title('DFT with Fs = 12 KHz and N = 128');
xlabel('Frequency (Hz)');
ylabel('X(f)')

% for N = 256
[f, y] = foo(Fs, 256);
subplot(3, 1, 3);
stem(f, y);
xticks(-Fs / 2 : 1000 : Fs / 2);
title('DFT with Fs = 12 KHz and N = 256');
xlabel('Frequency (Hz)');
ylabel('X(f)')

function [f, y] = foo(Fs, N)
    % Function to generate DFT of the given signal
    % Inputs:
    % Fs : Sampling frequency
    % N : Dimension of DFT
    % Outputs:
    % f : Frequency range of DFT
    % y : DFT itself
    t = 0 : 1 / Fs : 0.1;
    x = 10 * cos(2 * pi * 1000 * t) + 6 * cos(2 * pi * 2000 * t) + 2 * cos(2 * pi * 4000*
t);
    f = -Fs / 2 : Fs / (N - 1) : Fs / 2;
    y = fftshift(abs(fft(x, N)));
end
```

Code for Figure 2

```
% Author: Utkarsh Patel (18EC30048)
% Experiment - 1 Part B-1
% DFT of the given signal with sampling frequency of 8 KHz

Fs = 8000; % sampling frequency

% for N = 64
[f, y] = foo(Fs, 64);
subplot(3, 1, 1);
stem(f, y);
xticks(-Fs / 2 : 1000 : Fs / 2);
title('DFT with Fs = 8 KHz and N = 64');
xlabel('Frequency (Hz)');
ylabel('X(f)')

% for N = 128
[f, y] = foo(Fs, 128);
subplot(3, 1, 2);
stem(f, y);
xticks(-Fs / 2 : 1000 : Fs / 2);
title('DFT with Fs = 8 KHz and N = 128');
xlabel('Frequency (Hz)');
ylabel('X(f)')

% for N = 256
[f, y] = foo(Fs, 256);
subplot(3, 1, 3);
stem(f, y);
xticks(-Fs / 2 : 1000 : Fs / 2);
title('DFT with Fs = 8 KHz and N = 256');
xlabel('Frequency (Hz)');
ylabel('X(f)')

function [f, y] = foo(Fs, N)
    % Function to generate DFT of the given signal
    % Inputs:
    % Fs : Sampling frequency
    % N : Dimension of DFT
    % Outputs:
    % f : Frequency range of DFT
    % y : DFT itself
    t = 0 : 1 / Fs : 0.1;
    x = 10 * cos(2 * pi * 1000 * t) + 6 * cos(2 * pi * 2000 * t) + 2 * cos(2 * pi * 4000*
t);
    f = -Fs / 2 : Fs / (N - 1) : Fs / 2;
    y = fftshift(abs(fft(x, N)));
end
```

Code for Figure 3

```
% Author: Utkarsh Patel (18EC30048)
% Experiment - 1 Part B-2
% DFT of the given signal with sampling frequency of 5 KHz

Fs = 5000; % sampling frequency

% for N = 64
[f, y] = foo(Fs, 64);
subplot(3, 1, 1);
stem(f, y);
xticks(-Fs / 2 : 1000 : Fs / 2);
title('DFT with Fs = 5 KHz and N = 64');
xlabel('Frequency (Hz)');
ylabel('X(f)')

% for N = 128
[f, y] = foo(Fs, 128);
subplot(3, 1, 2);
stem(f, y);
xticks(-Fs / 2 : 1000 : Fs / 2);
title('DFT with Fs = 5 KHz and N = 128');
xlabel('Frequency (Hz)');
ylabel('X(f)')

% for N = 256
[f, y] = foo(Fs, 256);
subplot(3, 1, 3);
stem(f, y);
xticks(-Fs / 2 : 1000 : Fs / 2);
title('DFT with Fs = 5 KHz and N = 256');
xlabel('Frequency (Hz)');
ylabel('X(f)')

function [f, y] = foo(Fs, N)
    % Function to generate DFT of the given signal
    % Inputs:
    % Fs : Sampling frequency
    % N : Dimension of DFT
    % Outputs:
    % f : Frequency range of DFT
    % y : DFT itself
    t = 0 : 1 / Fs : 0.1;
    x = 10 * cos(2 * pi * 1000 * t) + 6 * cos(2 * pi * 2000 * t) + 2 * cos(2 * pi * 4000*
t);
    f = -Fs / 2 : Fs / (N - 1) : Fs / 2;
    y = fftshift(abs(fft(x, N)));
end
```

Code for Figure 4

```
% Author: Utkarsh Patel (18EC30048)
% Experiment - 1 Part B-3
% DFT of the given signal with sampling frequency of 4 KHz

Fs = 4000; % sampling frequency

% for N = 64
[f, y] = foo(Fs, 64);
subplot(3, 1, 1);
stem(f, y);
xticks(-Fs / 2 : 1000 : Fs / 2);
title('DFT with Fs = 4 KHz and N = 64');
xlabel('Frequency (Hz)');
ylabel('X(f)')

% for N = 128
[f, y] = foo(Fs, 128);
subplot(3, 1, 2);
stem(f, y);
xticks(-Fs / 2 : 1000 : Fs / 2);
title('DFT with Fs = 4 KHz and N = 128');
xlabel('Frequency (Hz)');
ylabel('X(f)')

% for N = 256
[f, y] = foo(Fs, 256);
subplot(3, 1, 3);
stem(f, y);
xticks(-Fs / 2 : 1000 : Fs / 2);
title('DFT with Fs = 4 KHz and N = 256');
xlabel('Frequency (Hz)');
ylabel('X(f)')

function [f, y] = foo(Fs, N)
    % Function to generate DFT of the given signal
    % Inputs:
    % Fs : Sampling frequency
    % N : Dimension of DFT
    % Outputs:
    % f : Frequency range of DFT
    % y : DFT itself
    t = 0 : 1 / Fs : 0.1;
    x = 10 * cos(2 * pi * 1000 * t) + 6 * cos(2 * pi * 2000 * t) + 2 * cos(2 * pi * 4000*
t);
    f = -Fs / 2 : Fs / (N - 1) : Fs / 2;
    y = fftshift(abs(fft(x, N)));
end
```

Code for Figure 5

```
% Author: Utkarsh Patel (18EC30048)
% Experiment - 1 Part C (Run on MATLAB online)
% Analyzing spectrum of square wave

Fs = 20000;               % Sampling Frequency
f = 1000;                 % Frequency of square wave
N = 256;                  % Number of samples
t = 0 : 1/Fs : (N - 1)/Fs; % Time range
x = square(2*pi*f*t);     % Square wave
y = fft(x);               % DFT of square signal
z = fftshift(y);
frange = (-N/2 : N/2-1) * (Fs/N); % Frequency range for DFT
plot(frange, abs(z)/N);
title('Spectrum of a square wave');
xlabel('Frequency (Hz)');
ylabel('X(f) / N');
```

Code for Figure 6

```
% Author: Utkarsh Patel (18EC30048)
% Experiment - 1 Part D (Run on MATLAB online)
% Interpolation and upsampling

Fs = 12000;  % Sampling frequency
N = 256;     % Number of samples for FFT
Nf = 2*N;    % Samples required after zero embedding
t = 0 : 1/Fs : (N-1)/Fs; % time range
s = 10 * cos(2 * pi * 1000 * t) + 6 * cos(2 * pi * 2000 * t) + 2 * cos(2 * pi * 4000 * t);
% signal used for this experiment
y = upsample(s,2); % upsampling singal 's' by zero embedding

% Filter design
d = designfilt('lowpassfir', 'FilterOrder', 20, 'CutoffFrequency', 6000, 'SampleRate',
24000); % Specification of filter
y_f = filter(d, y);              % filtering the 'upsampled' signal
y_fft = abs(fft(y_f))/(Nf);      % FFT of filtered signal
t1 = 0:1/(2*Fs):(Nf-1)/(2*Fs);
s1 = 10*cos(2*pi*1000*t1) + 6*cos(2*pi*2000*t1) + 2*cos(2*pi*4000*t1); % Sampling @ 24 KHz
s1_fft = abs(fft(s1)) / (Nf);
f = -Fs : 2 * Fs / (Nf - 1) : Fs; % frequency range for DFT

subplot(2, 1, 1);
plot(f, y_fft);
title('DFT of signal generated by upsampling');
xlabel('Frequency (Hz)');
ylabel('X(f) / N');
subplot(2, 1, 2);
plot(f, s1_fft);
title('DFT of signal generated by sampling at 24 KHz');
xlabel('Frequency (Hz)');
ylabel('X(f) / N');
```