Lab 7 – Sampling and Reconstruction

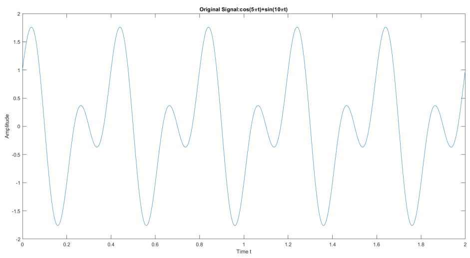
Name: K Sri Rama Rathan Reddy - 2022102072

Team Mate: B Karthikeya - 2022102042

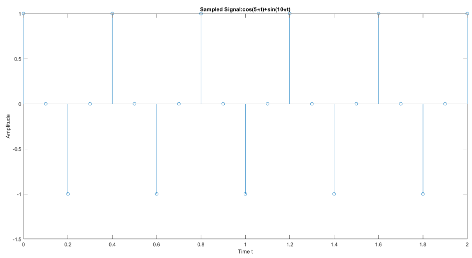
Team: Noicifiers

7.1 A signal and its samples

Given Signal:

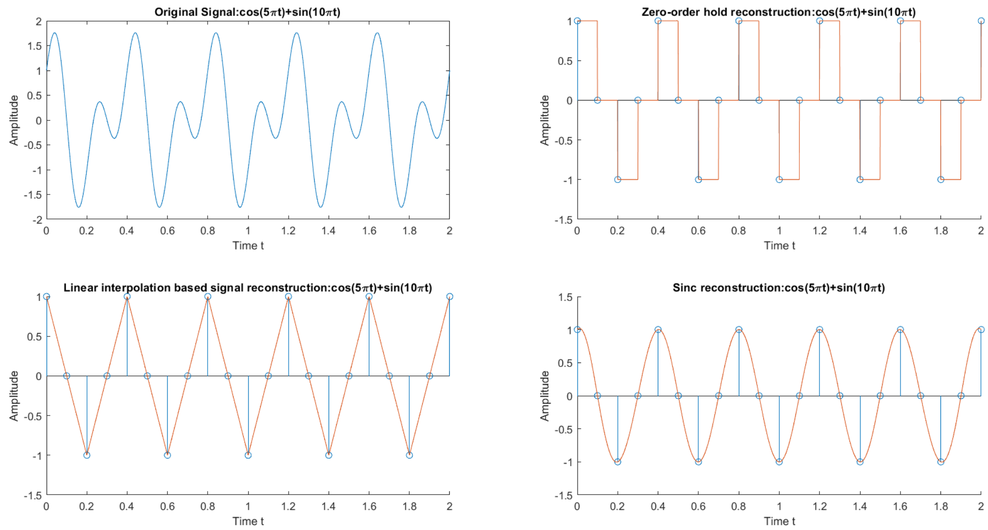


Samples of given Signal:

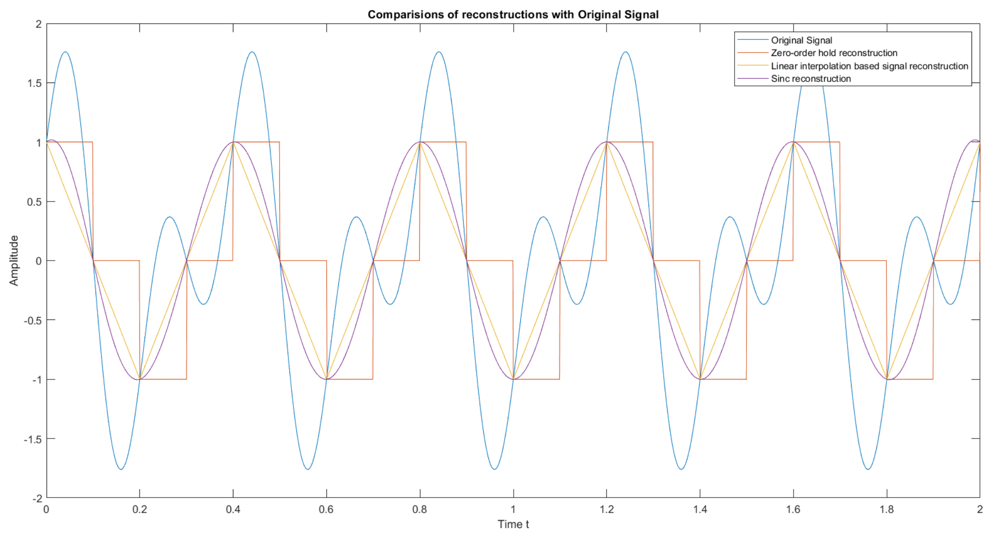


7.2 Reconstruction methods

Reconstruction of the given Signal in three different methods:



Comparisons of the three methods:



Zero-Order Hold Reconstruction:

- For two samples at time t_i and t_{i+1}, it just assigns the value at t_i to all values of t form t_i to t_{i+1}
- hold indicates holding the value until the next sample is encountered.
- The reconstructed signal has a High Maximum Absolute Error from the original signal

Linear Interpolation-based Signal Reconstruction:

- For two samples at time t_i and t_{i+1}, it just assigns the value at $y(t),\ where\ y(t) = \frac{y(t_{i+1})-y(t_i)}{t_{i+1}-t_i} \cdot (t - t_i)$ to all values of t form t_i to t_{i+1}
- The Reconstructed signal is just the lines joining the samples in the sampled signal

Sinc Reconstruction:

- The formula for Sinc Reconstruction:

$$x_r(t) = \sum_{n=-\infty}^{\infty} T_s \cdot x(nT_s) \cdot \frac{sin(\omega_c(t - nT_s))}{\pi(\omega_c(t - nT_s))}$$

- Though this is an infinite sum and cannot be exactly implemented in a computer, we will approximately implement it by restricting it to a particular time interval and using only the samples $x[n]$ we have from that interval.
- This is the better form of reconstruction as the error between the original signal and the reconstructed signal is minimal for Sinc reconstruction of all three.

Sinc Reconstruction function:

```
function xr = sinc_recon(n,xn,Ts,t_fine)
    ws = 2*(pi/Ts);
    wc = ws/2;
    xr = zeros(size(t_fine));
    s = zeros(size(t_fine));
    for t = 1:1:length(t_fine)
        for k = 1:1:length(n)
            if (wc*(t_fine(t) - n(k)*Ts)) == 0
                s(t) = 1;
            else
                s(t) = (sin(wc*(t_fine(t) - n(k)*Ts))/(wc*(t_fine(t) - n(k)*Ts)));
            end
            % s(t) = sinc((wc/pi)*(t_fine(t) - n(k)*Ts));
            xr(t) = xr(t)+(Ts*xn(k)*(wc/pi)*s(t));
        end
    end
end
```

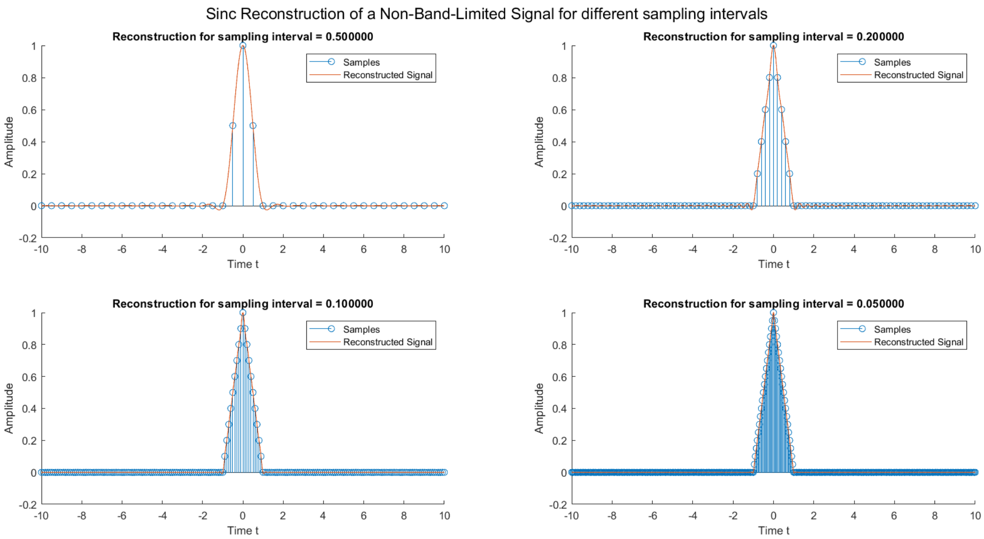
- **Inputs:**
    - **n -** The array containing the integer points where the sample lies
    - **xn** - The sampled signal which needs to be reconstructed
    - **Ts** - Sampling Interval
    - **t_fine** - The time span to which the signal needs to be reconstructed

- **Outputs:**
  - **xr** - The array which contains the reconstructed signal

Maximum Absolute Errors:

```
The maximum absolute error (MAE) between the original signal and the Zero-order hold reconstructed signal = 1.760074
The maximum absolute error (MAE) between the original signal and the Linear interpolation based signal reconstructed signal = 1.207633
The maximum absolute error (MAE) between the original signal and the Sinc reconstructed signal = 1.034058
The maximum absolute error (MAE) between the original signal and the SPLINE Interpolation reconstructed signal = 1.019607
```

7.3 Sampling non-band-limited signal

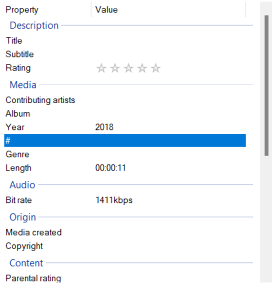


As the Sampling Interval(Ts) is changed,

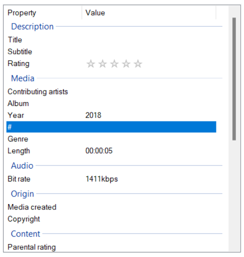
- As **Ts decreases**, No. of samples considered increases the signal reconstructed would be **closer to the original one**.
- If Ts is very large, Then there is a chance of losing the originality of the reconstructed signal.

7.4 Audio signals

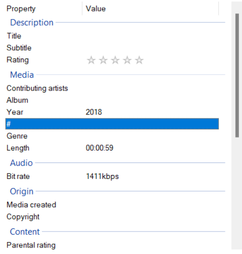
Properties of the Audio files:



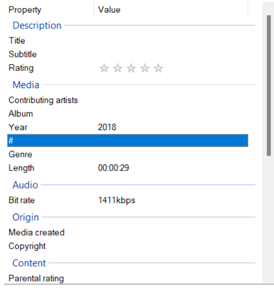
file_example_WAV_1MG



file_example_WAV_2MG



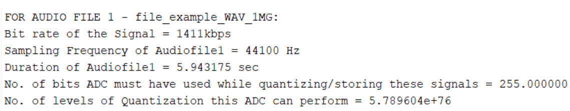
file_example_WAV_5MG



file_example_WAV_10M

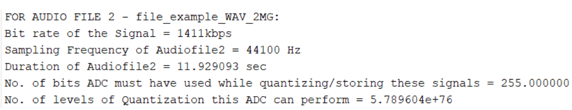
```
G1 = [0.9 0.8 0.7];
G2 = [1.2 1.4 1.6];

Bit_rate = 1411;

disp("FOR AUDIO FILE 2 - file_example_WAV_2MG:")
disp("Bit rate of the Signal = " + Bit_rate + "kbps");
[y2,fs2] = audioread("Audio files\file_example_WAV_2MG.wav");
fprintf("Sampling Frequency of Audiofile2 = %d Hz\n",fs2);
d2 = length(y2)/fs2;
fprintf("Duration of Audiofile2 = %f sec\n",d2);
num_bits2 = floor(Bit_rate*8*1000*(1/fs2));
fprintf("No. of bits ADC must have used while quantizing/storing these signals = %f\n",num
_bits2);
num_quantization_levels2 = 2^num_bits2;
fprintf("No. of levels of Quantization this ADC can perform = %d\n",num_quantization_level
s2);
sound(y2,fs2);
pause(12);
for k = 1:length(G1)
    sound(y2,G1(k)*fs2);
    pause(16);
end
for k = 1:length(G2)
    sound(y2,G2(k)*fs2);
    pause(12);
end
fprintf("\n");
```

Outputs of the code for all four Audio Files:



```
FOR AUDIO FILE 1 - file_example_WAV_1MG:
Bit rate of the Signal = 1411kbps
Sampling Frequency of Audiofile1 = 44100 Hz
Duration of Audiofile1 = 5.943175 sec
No. of bits ADC must have used while quantizing/storing these signals = 255.000000
No. of levels of Quantization this ADC can perform = 5.789604e+76
```

for audio file 1



```
FOR AUDIO FILE 2 - file_example_WAV_2MG:
Bit rate of the Signal = 1411kbps
Sampling Frequency of Audiofile2 = 44100 Hz
Duration of Audiofile2 = 11.929093 sec
No. of bits ADC must have used while quantizing/storing these signals = 255.000000
No. of levels of Quantization this ADC can perform = 5.789604e+76
```

for audio file 2

```
FOR AUDIO FILE 3 - file_example_WAV_5MG:
Bit rate of the Signal = 1411kbps
Sampling Frequency of Audiofile3 = 44100 Hz
Duration of Audiofile3 = 29.983061 sec
No. of bits ADC must have used while quantizing/storing these signals = 255.000000
No. of levels of Quantization this ADC can perform = 5.789604e+76
```

for audio file 3

```
FOR AUDIO FILE 4 - file_example_WAV_10MG:
Bit rate of the Signal = 1411kbps
Sampling Frequency of Audiofile4 = 44100 Hz
Duration of Audiofile4 = 59.772971 sec
No. of bits ADC must have used while quantizing/storing these signals = 255.000000
No. of levels of Quantization this ADC can perform = 5.789604e+76
```

for audio file 4

- **Bit Rate**:
  - Number of Bits used to represent one second of an Audio.
  - The bit rate for all the given Audio signals is 1411 kbps.
- **Sampling Frequency**:
  - No. of samples of the audio signal occurs per second.
  - The Sampling frequency for all the given Audio signals is 44100 Hz.
- **Duration**:
  - Time taken by all the samples of the Audio Signal
  - = (length of the signal/sampling frequency)
  - = (length of first component / second component) where the components are returned by the audioread() function.
- **No.of bits used by ADC while quantizing or storing the signal**:
  - It is the number of bits per sample of the Audio signal
  - = (Bit rate/Sampling frequency)
  - = [((1411*1000*8)/44100)] = 255 bits
- **No. of levels of quantization this ADC can perform** :
  - While converting from Analog to Digital, each bit is assigned 0 or 1 randomly.
  - Therefore one bit has 2 levels of Quantization.
  - So, N bits would have 2^N levels of Quantization.
  - i.e., $5.7896 \cdot 10^{76}$ levels of quantization per sample for the given audio signals.

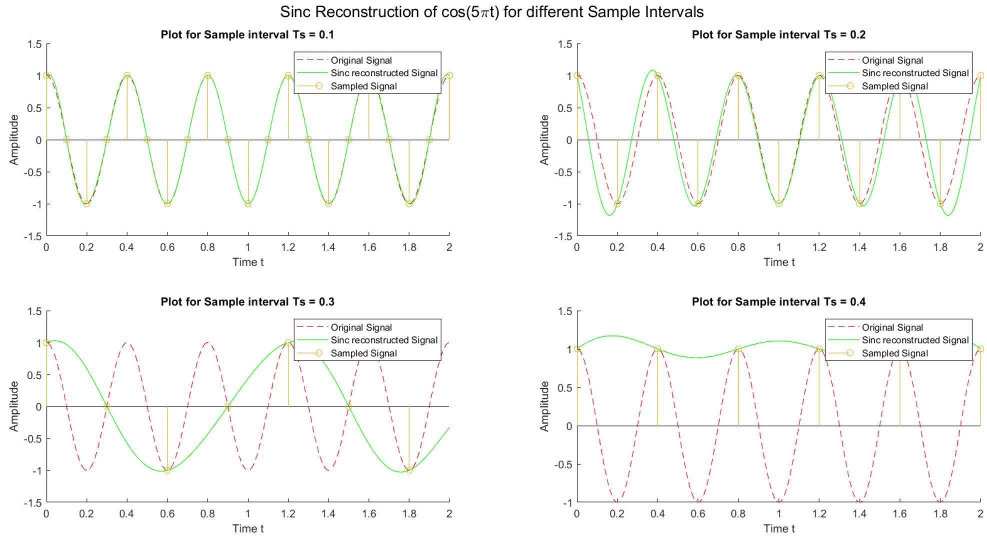
Changing the sampling Frequency:

- When the Sampling frequency is changed, the duration of the audio signal changes.
- The sampling frequency is increased $\Rightarrow$ Duration decreases
- The sampling frequency is decreased $\Rightarrow$ Duration increases

- i.e., the Signal is scaled by changing the Sampling Frequency
- The time scaling Property of Fourier transformation is observed here.
  - $if\ FT\{f(t)\} = F(\omega)$
    $then\ FT\{f(a \cdot t)\} = \frac{a}{a} F(\frac{\omega}{|a|})$ is the time scaling property of Fourier Transform.

7.5 Aliasing

- Nyquist Rate for $cos(5\pi t) = 2 \cdot 5\pi = 10\pi$



- As the sampling Interval is changed,
  - As Ts increases, No. of Samples considered for the reconstruction of the signal, decreases
  - So, the signal may not be ideally reconstructed
  - Therefore Originality of the signal may be lost.
  - Here, Ts ≤ 0.2 for ideal reconstruction (Nyquist rate i.e., fs ≥ 10***π*** / 2***π*** = 5 or Ts ≤ 0.2)