
# Digital Signal Processing Ⅰ

# 9th Week EXPERIMENT

# Report

(6th report of DSP1 course)

| | |
|---|---|
| **Subject** | Digital Signal Processing Ⅰ |
| **Professor** | Joon-Hyuk Chang |
| **Submission Date** | May 5th, 2021 |
| **University** | Hanyang University |
| **School** | College of Engineering |
| **Department** | Department of Computer Science & Engineering |
| **Student ID** | **Name** |
| 2019009261 | 최가온(CHOI GA ON) |

# -Contents-

# Ⅰ. Abstraction

## 1. The frequency domain representation of LTI systems

### 1-1. Frequency Response

The discrete-time Fourier transform of an impulse response is called the frequency response of an LTI system and is denoted by

$$H(e^{j\omega n}) = \sum_{-\infty}^{\infty} h(n)e^{-j\omega n}$$

In general, the frequency response $H(e^{j\omega})$ is a complex function of $\omega$. The magnitude $|H(e^{j\omega})|$ of $H(e^{j\omega})$ is called the magnitude (or gain) response function, and the angle $\angle H(e^{j\omega})$ is called the phase response function.

### 1-2. Response to sinusoidal sequences

Let $x(n) = \mathrm{A}\cos(\omega_0 n + \theta_0)$ be an input to an LTI system h(n).

Then,

$$y(n) = A|H(e^{jw_0})|\cos(w_0 n + \theta_0 + \angle H(e^{jw_0}))$$

This response is called the steady-state response, denoted by $y_{ss}(n)$.

### 1-3. Response to arbitary sequences

It can be generalized to arbitrary absolutely summable sequences.

$$Y(e^{jw}) = H(e^{jw})X(e^{jw})$$

The output y(n) is then computed from $Y(e^{jw})$ using the inverse discrete-time Fourier transform.

## 1-4. Frequency response function from difference equations

When an LTI system is represented by the difference equation

$$y(n) = \sum_{l=1}^{N} a_l y(n-l) = \sum_{m=0}^{M} b_m x(n-m)$$

then to evaluate its frequency response, we would need the impulse response h(n). However, we can easily obtain $H(e^{jw})$. We know that when $x(n) = e^{jwn}$, then y(n) must be $H(e^{jw})e^{jw}$.

$$H(e^{jw})e^{jwn} + \sum_{l=1}^{N} a_l H(e^{jw})e^{jw(n-l)} = \sum_{m=0}^{M} b_m e^{jw(n-m)}$$

# 2. Sampling and Reconstruction

## 2-1. Sampling

Let $x_a(t)$ be an analog signal. Its continuous-time Fourier transform(CTFT) is given by

$$X_a(j\Omega) = \int_{-\infty}^{\infty} x_a(t)e^{-j\Omega t}dt$$

where $\Omega$ is an analog frequency in radians/sec.

The inverse continuous-time Fourier transform is given by

$$x_a(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} X_a(j\Omega)e^{j\Omega t}d\Omega$$

We now sample $x_a(t)$ at sampling interval $T_s$ seconds apart to obtain the discrete-time signal x(n).

$$x(n) = x_a(nT_s)$$

Let $X(e^{jw})$ be the discrete-time Fourier transform of $x(n)$. Then it can be shown that $X(e^{jw})$ is a countable sum of amplitude-scaled, frequency-scaled, and translated versions of the Fourier transform $X_a(j\Omega)$.

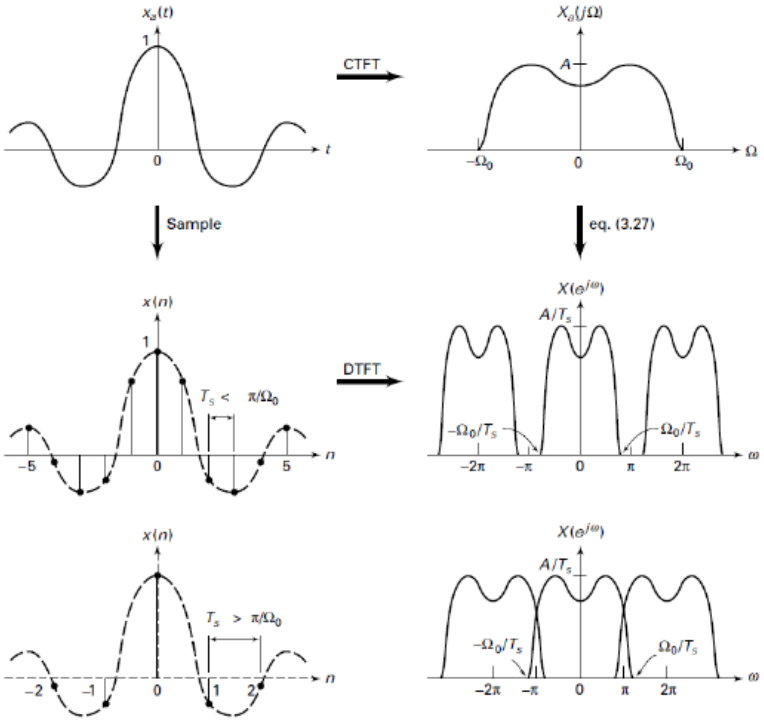
$$X(e^{jw}) = \frac{1}{T_s}\sum_{l=-\infty}^{\infty} X_a[j\left(\frac{w}{T_s} - \frac{2\pi}{T_s}l\right)]$$

This relation is known as the aliasing formula. The analog and digital frequencies are related through $T_s$

$$w = \Omega T_s$$

While the sampling frequency $F_s$ is given by

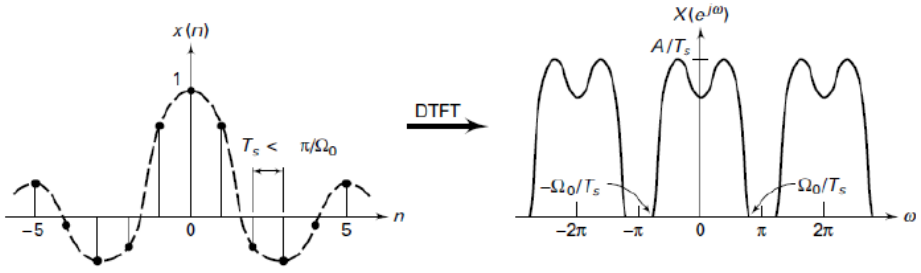
$$F_s = \frac{1}{T_s}, \quad sam/sec$$



**[Band-limited Signal]**

A signal is band-limited if there exists a finite radian frequency $\Omega_0$ such that $X_a(j\Omega)$ is zero for $|\Omega| > \Omega_0$. The frequency $F_0 = \frac{\Omega_0}{2\pi}$ is called the signal bandwidth in Hz.

$$X(e^{jw}) = \frac{1}{T_s} X\left(j\frac{w}{T_s}\right); \quad -\frac{\pi}{T_s} < \frac{w}{T_s} \leq \frac{\pi}{T_s}$$




**[Nyquist Sampling Theorem]**

A band-limited signal $x_a(t)$ with bandwidth $F_0$ can be reconstructed from its sample values $x(n) = x_a(nT_s)$ if the sampling frequency $F_s = \frac{1}{T_s}$ is greater than twice the bandwidth $F_0$ of $x_a(t)$.

$$F_s > 2F_0$$

Otherwise aliasing would result in x(n). The sampling rate of $2F_0$ for an analog band-limited signal is called the Nyquist rate.

## 2-2. Reconstruction

From the sampling theorem and the preceding examples, it is clear that if we sample band-limited $x_a(t)$ above its Nyquist rate, then we can reconstruct $x_a(t)$ above its Nyquist rate, then we can reconstruct $x_a(t)$ from its samples $x(n)$. This reconstruction can be thought of as a 2-step process:

1. First the samples are converted into a weighted impulse train.

$$\sum_{n=-\infty}^{\infty} x(n)\delta(t-nT_s) = \cdots + x(-1)\delta(n+T_s) + x(0)\delta(t) + x(1)\delta(n-T_s) + \cdots$$

2. Then the impulse train is filtered through an ideal analog lowpass filter band-limited to the $[-\frac{F_s}{2}, \frac{F_s}{2}]$ band.

This 2-step procedure can be describe mathematically using an interpolating formula

$$x_a(t) = \sum_{n=-\infty}^{\infty} x(n)sinc[F_s(t-nT_s)]$$

Practical D/A converters

In practice we need a different approach. The 2-step procedure is still feasible, but now we replace the ideal lowpass filter by a practical analog lowpass filter.

1. Zero-order-hold(ZOH) interpolation

   In this interpolation a given sample value is held for the sample interval until the next sample is received.

$$\widehat{x_a}(t) = x(n), \quad nT_s \le n < (n+1)T_s$$

   which can be obtained by filtering the impulse train through an interpolating filter of the form

$$h_0(t) = \begin{cases} 1, & 0 \le t \le T_s \\ 0, & otherwise \end{cases}$$

2. 1st-order-hold(FOH) interpolation

   In this case the adjacent samples are joined by straight lines. This can be obtained by filtering the impulse train through

$$h_1(t) = \begin{cases} 1 + \dfrac{t}{T_s}, & 0 \le t \le T_s \\ 1 - \dfrac{t}{T_s}, & T_s \le t \le 2T_s \\ 0, & otherwise \end{cases}$$

Once again, an appropriately designed analog postfilter is required for accurate reconstruction. These interpolations can be extended to higher orders.

# Ⅱ. Exercises

In this part, there are four exercise questions. Each exercise consists of code and its result. All documents including Matlab code, result, and this report are uploaded in this website :

https://github.com/Gaon-Choi/ELE3076/tree/main/5_Property_of_DTFT

**Example 3.13**

Determine the frequency response $H(e^{jw})$ of a system characterized by $h(n) = (0.9)^n u(n)$. Plot the magnitude and the phase response.

First, the frequency response can be obtained by Fourier Transform of h(n).

$$H(e^{jw}) = \sum_{n=-\infty}^{\infty} h(n)e^{-j\omega n} = \sum_{n=0}^{\infty} (0.9)^n e^{-j\omega n} = \sum_{n=0}^{\infty} \left(0.9e^{-j\omega}\right)^n = \frac{1}{1-0.9e^{-j\omega}}$$

Now the magnitude can be obtained by making absolute value of $H(e^{jw})$.

$$\left|H(e^{j\omega})\right| = \sqrt{\frac{1}{(1-0.9cos\omega)^2 + (0.9sin\omega)^2}} = \frac{1}{\sqrt{1.81 - 1.8cos\omega}}$$

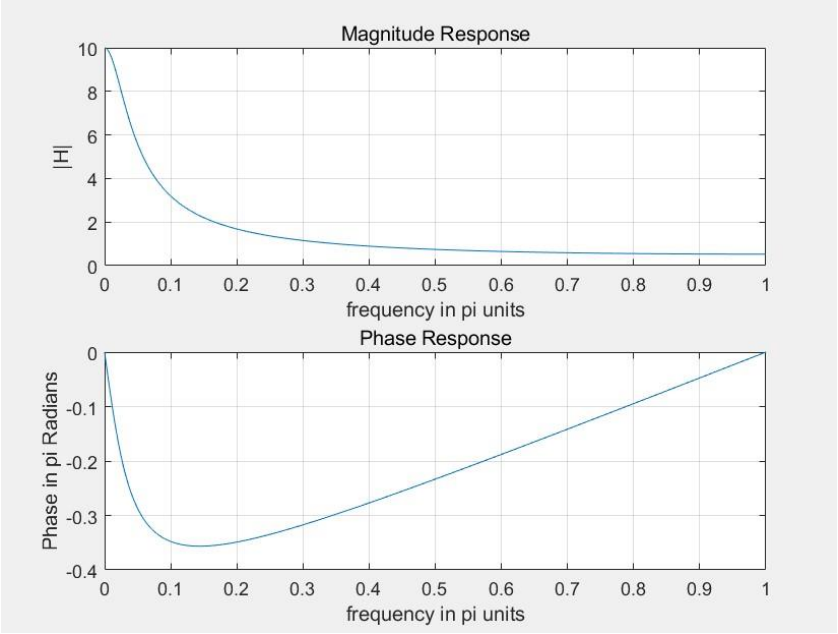
The phase of $H(e^{jw})$ can be obtained by arctan function.

$$\angle H(e^{j\omega}) = -\arctan\left(\frac{0.9sin\omega}{1-0.9cos\omega}\right)$$

(Matlab code)

```
w = [0:1:500] * pi / 500;   % [0, pi] axis divided into 501 points
H = exp(1j * w) ./ (exp(1j * w) - 0.9 * ones(1, 501));
magH = abs(H); angH = angle(H);
subplot(2, 1, 1); plot(w / pi, magH); grid;
xlabel("frequency in pi units"); ylabel("|H|");
title("Magnitude Response");
subplot(2, 1, 2); plot(w/pi, angH/pi); grid
xlabel("frequency in pi units"); ylabel("Phase in pi Radians");
title("Phase Response");
```

(Result)


Magnitude Response
|H|
10
8
6
4
2
0
0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1
frequency in pi units
Phase Response
Phase in pi Radians
0
-0.1
-0.2
-0.3
-0.4
0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1
frequency in pi units

**Example 3.15**

An LTI system is specified by the difference equation

$$y(n) = 0.8y(n-1) + x(n)$$

a. Determine $H(e^{j\omega})$

b. Calculate and plot the steady-state response $y_{ss}(n)$ to

$$x(n) = \cos(0.05\pi n)\, u(n)$$

The frequency response is

$$H(e^{j\omega}) = \frac{1}{1 - 0.8e^{-j\omega}}$$

In the steady state the input x(n) with frequency $\omega_0 = 0.05\pi$ and $\theta_0 = 0°$.

The response of the system is

$$H(e^{j0.05\pi}) = \frac{1}{1 - 0.8e^{-j0.05\pi}} = 4.0928e^{-j0.5377}$$

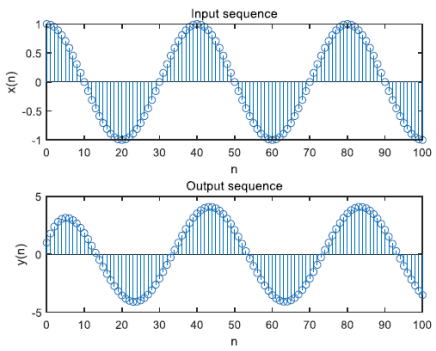
Therefore

$$y_{ss}(n) = 4.0928\cos(0.05\pi n - 0.5377) = 4.0928\cos[0.05\pi(n - 3.42)]$$

(Matlab code)

```
subplot(1, 1, 1)
b = 1; a = [1, -0.8];
n = [0:100]; x = cos(0.05 * pi * n);
y = filter(b, a, x);
subplot(2, 1, 1); stem(n, x);
xlabel("n"); ylabel("x(n)"); title("Input sequence")
subplot(2, 1, 2); stem(n, y);
xlabel("n"); ylabel("y(n)"); title("Output sequence")
```

(Result)



**Example 3.16**

The 3rd-order lowpass filter is described by the difference equation

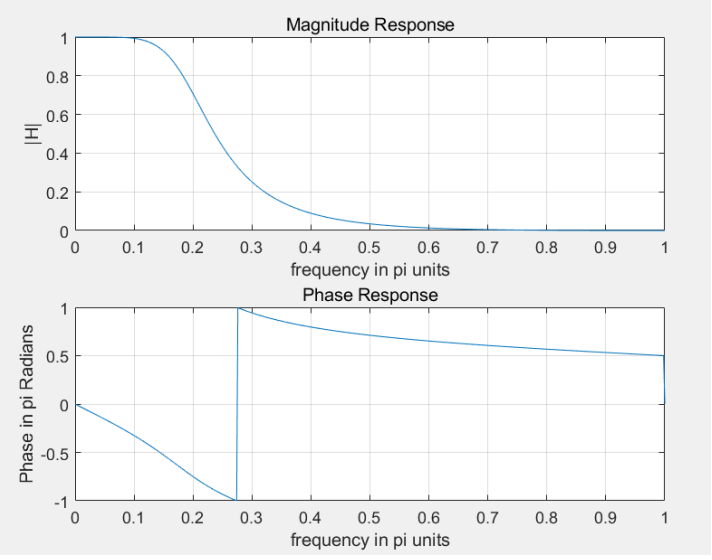
$$y(n) = 0.0181x(n) + 0.0543x(n-1) + 0.0543x(n-2) + 0.0181x(n-3) + 1.76y(n-1) - 1.1829y(n-2) + 0.2781y(n-3)$$

Plot the magnitude and the phase response of this filter, and verify that it is a lowpass filter.

(Matlab Code)

```
b = [0.0181, 0.0543, 0.0543, 0.0181];   % filter coefficient
array b
a = [1.0000, -1.7600, 1.1829, -0.2781]; % filter coefficient
array a
m = 0:length(b) - 1; l = 0:length(a) - 1;   % index arrays m
and l
K = 500; k = 0:1:K;
w = pi*k / K;
num = b * exp(-1j * m'*w);
den = a * exp(-1j * l'*w);
H = num ./ den;
magH = abs(H); angH = angle(H);
subplot(2, 1, 1); plot(w/pi, magH); grid; axis([0, 1, 0, 1])
xlabel("frequency in pi units"); ylabel("|H|");
title("Magnitude Response");
subplot(2, 1, 2); plot(w/pi, angH/pi); grid
xlabel("frequency in pi units"); ylabel("Phase in pi
Radians");
title("Phase Response");
```

(Result)



**Example 3.18**

Let $x_a(t) = e^{-1000|t|}$. Determine and plot its Fourier transform.

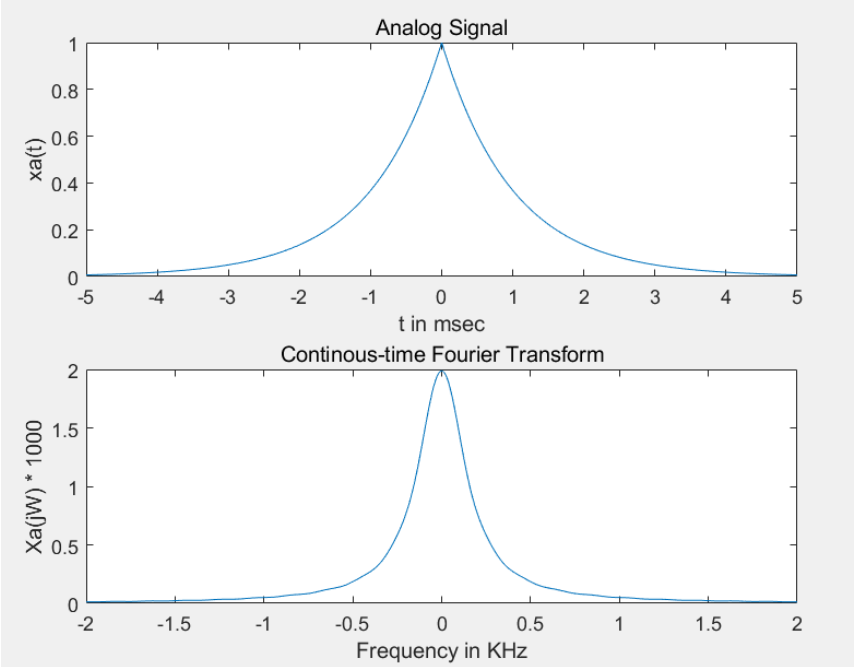
$$X_a(j\Omega) = \int_{-\infty}^{\infty} x_a(t) e^{-j\Omega t} dt = \frac{0.002}{1 + \left(\frac{\Omega}{1000}\right)^2}$$

$$\Delta t = 5 \times 10^{-5} \ll \frac{1}{2(2000)} = 25 \times 10^{-5}$$

(Matlab Code)

```
% Analog Signal
Dt = 0.00005; t = -0.005:Dt:0.005; xa = exp(-1000 * abs(t));
% Continuous-time Fourier Transformation
Wmax = 2 * pi * 2000; K = 500; k = 0:1:K; W = k * Wmax / K;
Xa = xa * exp(-1j * t'*W) * Dt; Xa = real(Xa);
W = [-fliplr(W), W(2:501)]; % Omega from -Wmax to Wmax
Xa = [fliplr(Xa), Xa(2:501)]; % Xa over -Wmax to Wmax
interval
subplot(2, 1, 1); plot(t * 1000, xa);
xlabel("t in msec"); ylabel("xa(t)");
title("Analog Signal");
subplot(2, 1, 2); plot(W/(2*pi*1000), Xa*1000);
xlabel("Frequency in KHz"); ylabel("Xa(jW) * 1000")
title("Continous-time Fourier Transform")
```

(Result)



**Example 3.19**

To study the effect of sampling on the frequency-domain quantities, we will sample $x_a(t)$ in Example 3.18 at 2 different sampling frequencies.

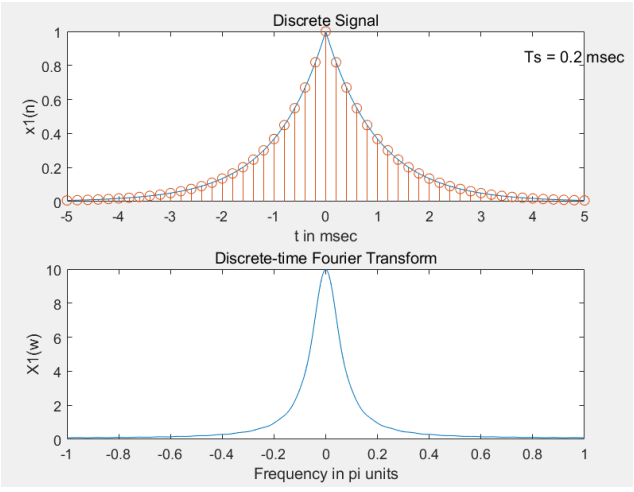
a. Sample $x_a(t)$ at $F_s = 5000$ sam/sec to obtain $x_1(n)$. Determine and plot $X_1(e^{jw})$.

b. Sample $x_a(t)$ at $F_s = 1000$ sam/sec to obtain $x_2(n)$. Determine and plot $X_2(e^{jw})$.

(Source Code)

```
% Analog Signal
Dt = 0.00005; t = -0.005:Dt:0.005; xa = exp(-1000 * abs(t));
% Discrete-time Signal
Ts = 0.0002; n = -25:1:25; x = exp(-1000 * abs(n * Ts));
% Discrete-time Fourier transform
K = 500; k = 0:1:K; w = pi*k/K;
X = x * exp(-1j * n'*w); X = real(X);
w = [-fliplr(w), w(2:K+1)]; X = [fliplr(X), X(2:K+1)];
subplot(2, 1, 1); plot(t*1000, xa);
xlabel("t in msec"); ylabel("x1(n)")
title("Discrete Signal"); hold on
stem(n * Ts * 1000, x); gtext("Ts = 0.2 msec"); hold off
subplot(2, 1, 2); plot(w/pi, X);
xlabel("Frequency in pi units"); ylabel("X1(w)");
title("Discrete-time Fourier Transform")
```

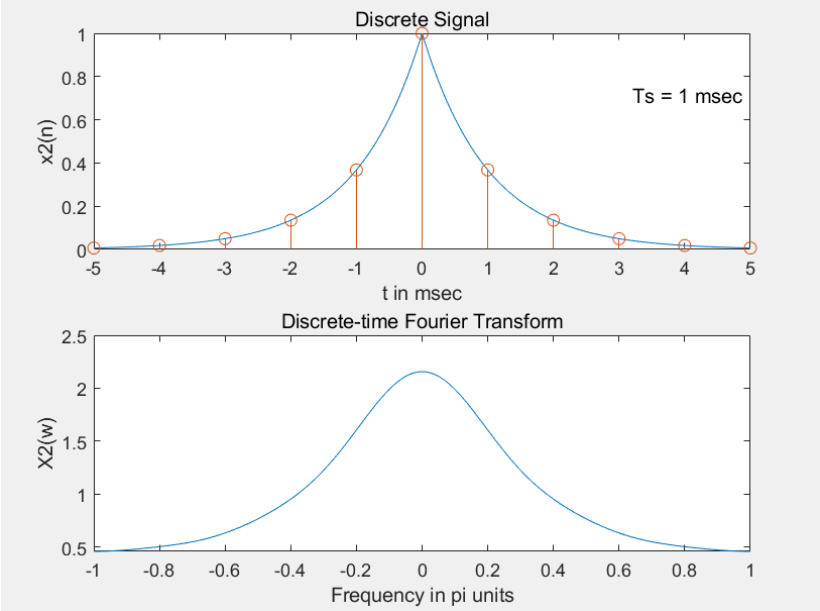
(Result)



(Source Code)

```
% Analog Signal
Dt = 0.00005; t = -0.005:Dt:0.005; xa = exp(-1000 * abs(t));
% Discrete-time Signal
Ts = 0.001; n = -5:1:5; x = exp(-1000 * abs(n * Ts));
% Discrete-time Fourier transform
K = 500; k = 0:1:K; w = pi*k/K;
X = x * exp(-1j * n'*w); X = real(X);
w = [-fliplr(w), w(2:K+1)]; X = [fliplr(X), X(2:K+1)];
subplot(2, 1, 1); plot(t*1000, xa);
xlabel("t in msec"); ylabel("x2(n)")
title("Discrete Signal"); hold on
stem(n * Ts * 1000, x); gtext("Ts = 1 msec"); hold off
subplot(2, 1, 2); plot(w/pi, X);
xlabel("Frequency in pi units"); ylabel("X2(w)");
title("Discrete-time Fourier Transform")
```

(Result)



**Example 3.21**

From the samples $x_1(n)$ in Example 3.19a, reconstruct $x_a(t)$ and comment on the results.

Note that $x_1(n)$ was obtained by sampling $x_a(t)$ at $T_s = \frac{1}{F_s} = 0.0002\,sec$. We will use the grid spacing of 0.00005 sec over $-0.005 \le t \le 0.005$, which gives x(n) over $-25 \le n \le 25$.

(Matlab Code)

```
% Discrete-time Signal x2(n)
Ts = 0.0002; n = -25:1:25; nTs = n * Ts; x = exp(-1000 * abs(nTs));
Fs = 1/Ts;
% Analog Signal reconstruction
Dt = 0.00005; t = -0.005:Dt:0.005;
xa = x * sinc(Fs*(ones(length(n), 1)*t-nTs'*ones(1, length(t))));
% check
error = max(abs(xa - exp(-1000 * abs(t))))
```

(Result)

**Example 3.22**

From the samples $x_2(n)$ in Example 3.17b reconstruct $x_a(t)$ and comment on the results.

In this case $x_2(n)$ was obtained by sampling $x_a(t)$ at $T_s = \frac{1}{F_s} = 0.001\ sec$. We will again use the grid spacing of 0.00005 sec over -0.005 ≤ t ≤ 0.005, which gives x(n) over -5 ≤ n ≤ 5.

(Matlab Code)

```
% Discrete-time Signal x2(n)
Ts = 0.001; n = -5:1:5; nTs = n * Ts; x = exp(-1000 * abs(nTs));
Fs = 1/Ts;
% Analog Signal reconstruction
Dt = 0.00005; t = -0.005:Dt:0.005;
xa = x * sinc(Fs*(ones(length(n), 1)*t-nTs'*ones(1, length(t))));
% check
```

(Result)

**Example 3.23**

Plot the reconstructed signal from the samples $x_1(n)$ in Example 3.19 using the ZOH and the FOH interpolations. Comment on the plots.

(Matlab Code)

```
% Discrete-time Signal x1(n) : Ts = 0.0002
Ts = 0.0002; n = -25:1:25; nTs = n * Ts; x = exp(-1000 * abs(nTs));
% Plots
subplot(2, 1, 1); stairs(nTs * 1000, x);
xlabel("t in msec."); ylabel("xa(t)");
title("Reconstructed Signal from x1(n) using zero-order-hold");
hold on
stem(n * Ts * 1000, x); hold off
%
% Discrete-time Signal x2(n) : Ts = 0.0001
Ts = 0.0001; n = -5:1:5; nTs = n * Ts; x = exp(-1000 * abs(nTs));
% Plots
subplot(2, 1, 2); plot(nTs * 1000, x);
xlabel("t in msec"); ylabel("xa(t)");
title("Reconstructed Signal from x2(n) using zero-order-hold");
hold on
stem(n * Ts * 1000, x); hold off
```

(Result)