# Digital Signal Processing Ⅰ

# 3rd Week EXPERIMENT

# Report

(1st report of DSP1 course)

| | |
|---|---|
| **Subject** | Digital Signal Processing Ⅰ |
| **Professor** | Joon-Hyuk Chang |
| **Submission Date** | March 23rd, 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. Discrete-time signal

   1-1. Definition and its implementation with Matlab

   1-2. Types of discrete-time signal

   1-3. Operations on discrete-time signal

Ⅱ. Exercises

1. Matlab codes of each exercise

2. Results

# Ⅰ. Abstraction

## 1. Discrete-time signal

### 1-1. Definition and its implementation with Matlab

Discrete-time signal is a time series consisting of a sequence of quantities. It views values of variables as occuring at distinct, separate "points in time", or equivalently as being unchanged throughout each non-zero region of time("time period")—that is, time is viewd as a discrete variable.[1]

Its form can be regarded as a number sequence,

$$x(n) = \{\ x(n)\ \} = \{\dots \quad , x(-1), x(0), x(1), \quad \dots\ \}$$

where the highlighted part indicates the sample at n = 0.

Matlab is a mathematics tool, which can be used in Engineering Lab or experiment involving statistic knowledge. With Matlab, discrete-time signal can be implemented as a data structure called "list". Here is an example.

All signal can be represented with two components. One is "time" and the other is "information". Thus, a discrete-time signal can be implemented with two list data structure in CS field.

```
Example: x[n] = { 2, 1, 1, 0, 1, 4, 3, 7 }
>> n = [3, 2, 1, 0, 1, 2, 3, 4]; x = [2, 1, 1, 0, 1, 4, 3, 7];
```

[1] https://en.wikipedia.org/wiki/Discrete_time_and_continuous_time

## 1-2. Types of discrete-time signal

### 1. Unit sample sequence

$$\delta(n) = \begin{cases} 1, & n = 0 \\ 0, & n \neq 0 \end{cases} = \{\ \dots, 0, 0, 1, 0, 0, \dots\}$$

In Matlab, the function zeros(1, N) generates a row vector of N zeros, which can be used to implement $\delta(n)$ over a finite interval. However, the logical relation n == 0 is an elegant way of implementing $\delta(n)$. For example, to implement

$$\delta(n - n_0) = \begin{cases} 1, & n = n_0 \\ 0, & n \neq n_0 \end{cases}$$

over the $n_1 \leq n_0 \leq n_2$ interval, we will use the following MATLAB function.

7 lines (7 sloc) | 196 Bytes

```
function [x, n]=impseq(n0,n1,n2)
% Generates x(n) = delta(n - n0); n1 <= n <= n2
% -----------------------------------------------
% [x, n] = impseq(n0, n1, n2)
%
n = [n1:n2];
x = [(n-n0) == 0];
```

### 2. Unit step sequence

In MATLAB the function ones(1, N) generates a row vector of N ones. It can be used to generate u(n) over a finite interval. Once again an elegant approach is to use the logical relation n >= 0. To implement

$$u(n - n_0) = \begin{cases} 1, & n \geq n_0 \\ 0, & n < n_0 \end{cases}$$

over the $n_1 \leq n_0 \leq n_2$ interval, we will use the following MATLAB function.

6 lines (6 sloc) | 198 Bytes

```
function [x, n] = stepseq(n0, n1, n2)
% Generates x(n) = u(n-n0); n1 <= n <= n2
% ------------------------------------------------
% [x, n] = stepseq(n0, n1, n2)
%
n = [n1:n2]; x = [(n - n0) >= 0];
```

### 3. Real-valued exponential sequence

$$x(n) = a^n \ , \forall n; \ \ a \in R$$

Example: $x[n] \ = \ (0.9)^n$

```
>>> n = [0:10];     x = (0.9) .^ n;
```

### 4. Complex-valued exponential sequence

$$x(n) = e^{(\sigma + j\omega_0)n}, \ \ \forall n$$

Example: x[n] = exp([2+j3]n)

```
>>> n = [0:10];     x = exp((2+3j) * n);
```

## 1-3. Operations on discrete-time signal

### 1. Signal Addition

$$\{x_1(n)\} + \{x_2(n)\} \ = \ \{\, x_1(n) + x_2(n) \,\}$$

13 lines (13 sloc) | 672 Bytes

```
function [y, n] = sigadd(x1, n1, x2, n2)
% implements y(n) = x1(n) + x2(n)
% -------------------------------------------------
% [y, ] = sigadd(x1, n1, x2, n2)
%   y = sum sequence over n, which includes n1 and n2
%  x1 = first sequence over n1
%  x2 = second sequence over n2 (n2 can be different from n1)
%
n = min(min(n1), min(n2)):max(max(n1), max(n2));    % duration of y(n)
y1 = zeros(1, length(n)); y2 = y1;                  % initialization
y1(find((n >= min(n1))&(n <= max(n1)) == 1)) = x1;  % x1 with duration of y
y2(find((n >= min(n2))&(n <= max(n2)) == 1)) = x2;  % x2 with duration of y
y = y1 + y2;                                        % sequence addition
```

## 2. Signal Multiplication

$$\{x_1(n)\} \cdot \{x_2(n)\} = \{\, x_1(n)x_2(n) \,\}$$

13 lines (13 sloc) | 680 Bytes

```
function [y, n] = sigmult(x1, n1, x2, n2)
% implements y(n) = x1(n) * x2(n)
% -----------------------------------------------
% [y, ] = sigmult(x1, n1, x2, n2)
%   y = sum sequence over n, which includes n1 and n2
%  x1 = first sequence over n1
%  x2 = second sequence over n2 (n2 can be different from n1)
%
n = min(min(n1), min(n2)):max(max(n1), max(n2));    % duration of y(n)
y1 = zeros(1, length(n)); y2 = y1;                  % initialization
y1(find((n >= min(n1))&(n <= max(n1)) == 1)) = x1;  % x1 with duration of y
y2(find((n >= min(n2))&(n <= max(n2)) == 1)) = x2;  % x2 with duration of y
y = y1 .* y2;                                       % sequence multiplication
```

## 3. Scaling

$$\alpha\{x(n)\} = \{\, \alpha x(n) \,\}$$

## 4. Shifting

$$y(n) = \{\, x(n-k) \,\}$$

6 lines (6 sloc) | 165 Bytes

```
function [y, n] = sigshift(x, m, k)
% implements y(n) = x(n - k)
% -------------------------------------------------
% [y, n] = sigshift(x, m, k)
%
n = m + k; y = x;
```

## 5. Folding

$$y(n) = \{x(-n)\}$$

6 lines (6 sloc) | 167 Bytes

```
function [y, n] = sigfold(x, n)
% implements y(n) = x(-n)
% -----------------------------------------------
% [y, n] = sigfold(x, n)
%
y = fliplr(x); n = -fliplr(n);
```

# Ⅱ. Exercises

In this part, there are 7 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/1_discrete_time_signal

### Example 2.1-a

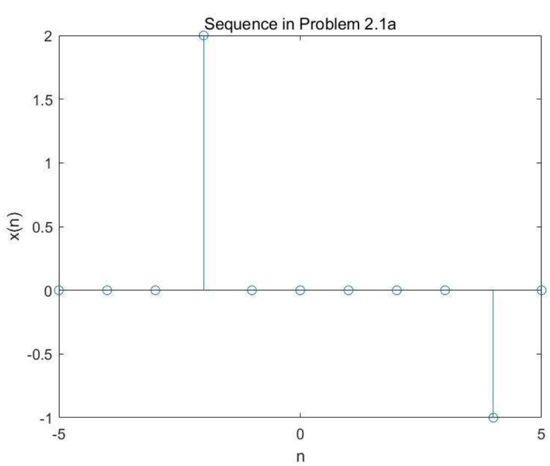
$$x(n) = 2\delta(n+2) - \delta(n-4), -5 \le n \le 5$$

(Matlab code)

```
편집기 - C:₩Users₩choig₩Documents₩MATLAB₩example2_1_a.m
example2_1_a.m

n = [-5: 5];
x = 2 * impseq(-2, -5, 5) - impseq(4, -5, 5);
stem(n, x); title('Sequence in Problem 2.1a')
xlabel('n'); ylabel('x(n)');
```

(Result)



## Example2-1.b

$$x(n) = n[u(n) - u(n-10) + 10e^{-0.3(n-10)}[u(n-10) - u(n-20)], 0 \leq n \leq 20$$

(Matlab Code)

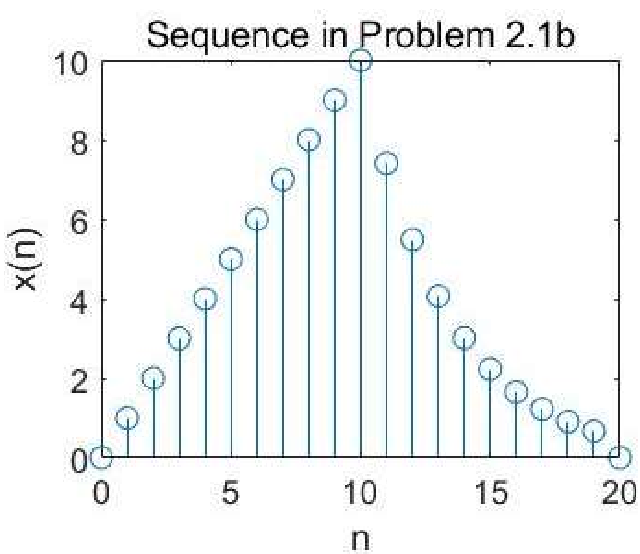
편집기 - C:₩Users₩choig₩Documents₩MATLAB₩example2_2_b.m

sigadd.m | example2_2_b.m

```
n = -2:10; x = [1:7, 6:-1:1];

[x21, n21] = sigfold(x, n); [x21, n21] = sigshift(x21, n21, 3);         % x(3-n)
[x22, n22] = sigshift(x, n, 2); [x22, n22] = sigmult(x, n, x22, n22);   % x(n)x(n-2)
[x2, n2] = sigadd(x21, n21, x22, n22);
subplot(2, 1, 2); stem(n2, x2); title("Sequence in Example 2.2b")
xlabel("n"); ylabel("x2(n)");
```

(Result)



## Example2-1.c

$$x(n) = \cos(0.04\pi n) = 0.2\omega(n), 0 \leq n \leq 50$$

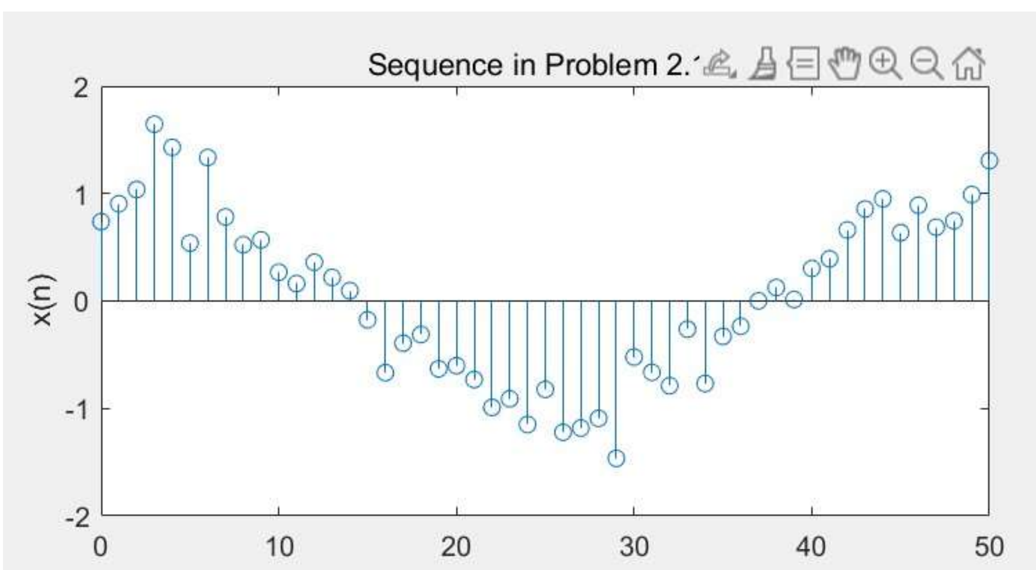
(Matlab Code)

편집기 - C:₩Users₩choig₩Documents₩MATLAB₩example2_1_c.m

example2_1_c.m

```
n = [0:50]; x = cos(0.04 * pi * n) + 0.2 * randn(size(n));
subplot(2, 2, 2); stem(n, x); title("Sequence in Problem 2.1c")
xlabel("n"); ylabel("x(n)");
```

(Result)



## Example2-1.d

$$\tilde{x}(n) = \{\ldots, 5, 4, 3, 2, 1, 5, 4, 3, 2, 1, 5, 4, 3, 2, 1, \ldots\}; \ \ -10 \leq n \leq 9.$$

Note that over the given interval, the sequence $\tilde{x}$(n) has four periods.

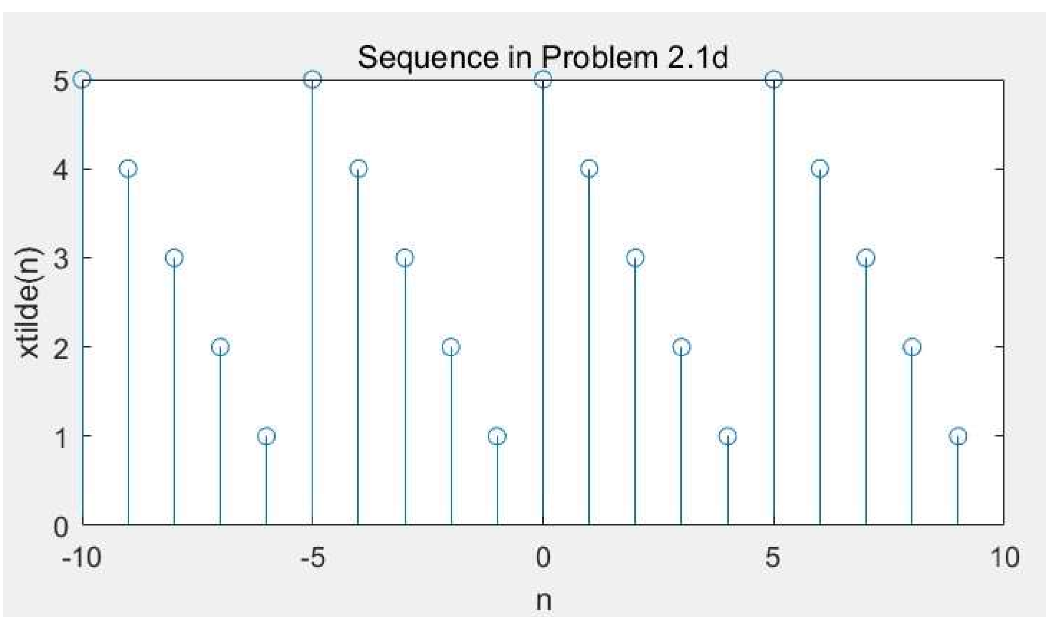
(Matlab Code)

편집기 - C:₩Users₩choig₩Documents₩MATLAB₩example2_1_d.m

example2_1_d.m

```
n = [-10:9]; x = [5, 4, 3, 2, 1];
xtilde = x' * ones(1, 4); xtilde = (xtilde(:))';
subplot(2, 2, 4); stem(n, xtilde); title("Sequence in Problem 2.1d")
xlabel("n"); ylabel("xtilde(n)");
```

(Result)



## Example2-2.a

$$x_1(n) = 2x(n-5) - 3x(n+4)$$

The first part is obtained by shifting x(n) by 5 and the second part by shifting x(n) by -4. This shifting and the addition can be easily done using the **sigshift** and the **sigadd** functions.

(Matlab Code)

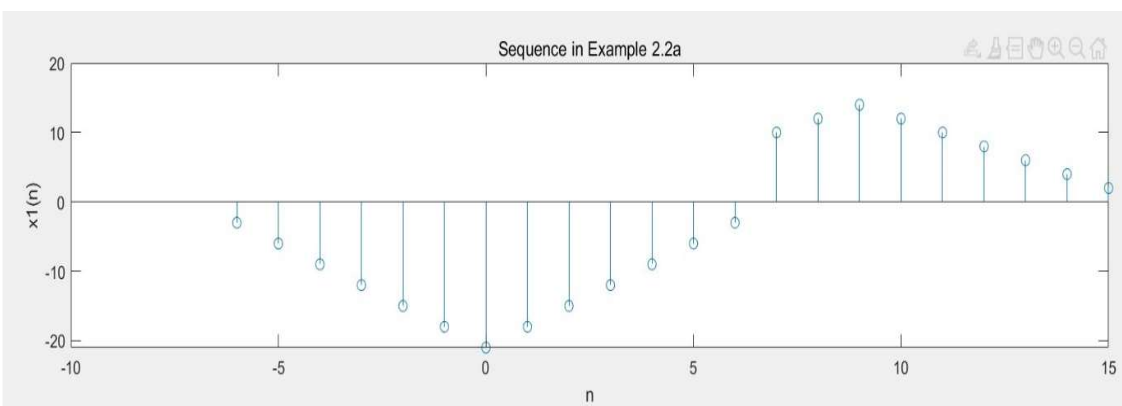
편집기 - C:₩Users₩choig₩Documents₩MATLAB₩example2_2_a.m

example2_2_a.m

```
n = -2:10; x = [1:7, 6:-1:1];

[x11, n11] = sigshift(x, n, 5); [x12, n12] = sigshift(x, n, -4);
[x1, n1] = sigadd(2 * x11, n11, -3 * x12, n12);
subplot(2, 1, 1); stem(n1, x1); title("Sequence in Example 2.2a")
xlabel("n"); ylabel("x1(n)");
```

(Result)



## Example2-2.b

$$x_2(n) = x(3-n) + x(n)x(n-2)$$

The first term can be writeen as x(-(n-3)). Hence it is obtained by first folding x(n) and then shifting the result by 3. The second part is a multiplication of x(n) and x(n-2), both of which have the same length but different support (or sample positions). These operations can be easily done using the **sigfold** and the **sigmult** functions.

(Matlab Code)

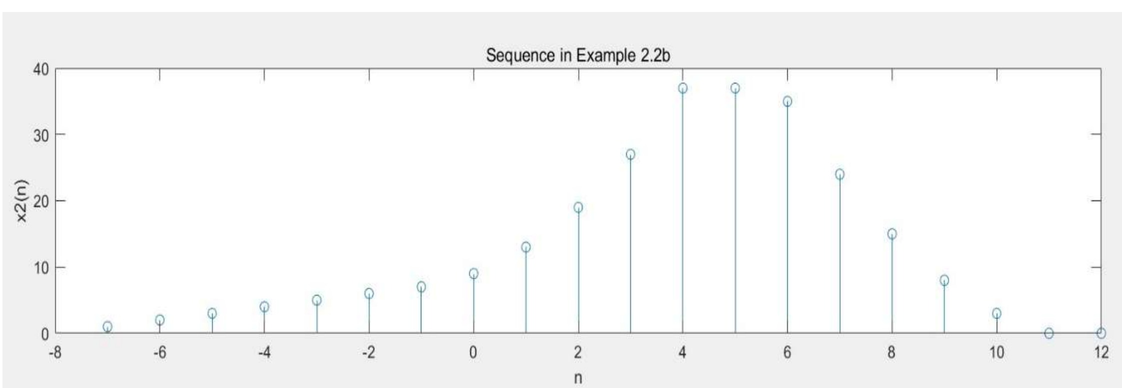
편집기 - C:₩Users₩choig₩Documents₩MATLAB₩example2_2_b.m

example2_2_b.m

```
n = -2:10; x = [1:7, 6:-1:1];

[x21, n21] = sigfold(x, n); [x21, n21] = sigshift(x21, n21, 3);        % x(3-n)
[x22, n22] = sigshift(x, n, 2); [x22, n22] = sigmult(x, n, x22, n22);   % x(n)x(n-2)
[x2, n2] = sigadd(x21, n21, x22, n22);
subplot(2, 1, 2); stem(n2, x2); title("Sequence in Example 2.2b")
xlabel("n"); ylabel("x2(n)");
```

(Result)



## Example2-3

Generate the complex-valued signal

$$x(n) = e^{(-0.1+j0.3)n}, \qquad -10 \leq n \leq 10$$

and plot its magnitude, phase, the real part, and the imaginary part in four separate subplots.

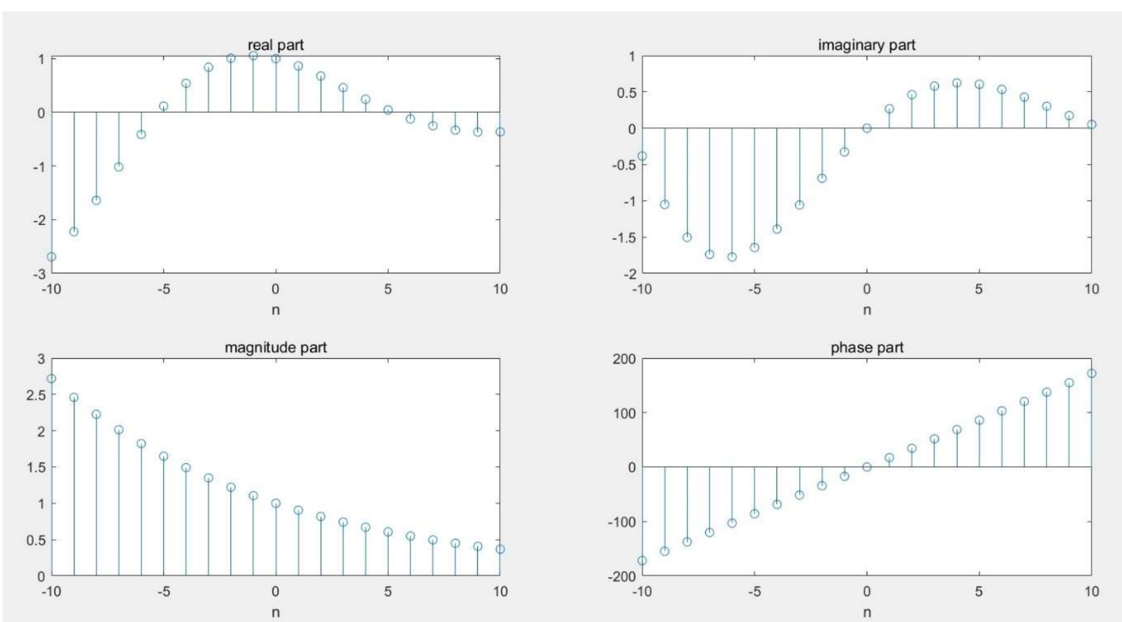
(Matlab Code)

편집기 - C:₩Users₩choig₩Documents₩MATLAB₩example2_3.m

example2_3.m

```
n = [-10:1:10]; alpha = -0.1 + 0.3j;
x = exp(alpha * n);
subplot(2, 2, 1); stem(n, real(x)); title("real part"); xlabel("n")
subplot(2, 2, 2); stem(n, imag(x)); title("imaginary part"); xlabel("n")
subplot(2, 2, 3); stem(n, abs(x)); title("magnitude part"); xlabel("n")
subplot(2, 2, 4); stem(n, (180/pi) * angle(x)); title("phase part"); xlabel("n");
```

(Result)



# References

Wikipedia(n.d.), Discrete time and continuous time,

https://en.wikipedia.org/wiki/Discrete_time_and_continuous_time

ASML Lab., Digital Signal Processing 1 Week 3 PPT, p.2 ~ p. 29.