Digital Signal Processing Ⅰ

5th Week EXPERIMENT

Report

(3rd report of DSP1 course)

Subject	Digital Signal Processing Ⅰ
Professor	Joon-Hyuk Chang
Submission Date	April 7th, 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. Difference Equations

1-1. Representation of LTI discrete system with Difference Equation

An LTI discrete system can be described by a linear constant coefficient difference equation of the form

$$y(n) = \sum_{k=0}^{N} a_k y(n-k) = \sum_{m=0}^{N} b_m x(n-m)\,, \forall n$$

And it is denoted by:

$$y(n) = x(n) * h(n)$$

1-2. Matlab implementation of Difference Equations

To solve difference equation numerically, there is an adequate function, which is called filter. The input of the function is signal input and the difference equation coefficients.

y = filter(b, a, x)

where b = [b0, b1, b2, …, bM]; a = [a0, a1, a2, …, aN];

To compute and plot impulse response, Matlab proves the function impz.

h = impz(b, a, n)

where n: size

a: denominator coeffients

b: numerator coeffients

2. Moving-Average Filter

2-1. Definition of Moving-Average Filter

A moving-average filter is a common method used for smoothing noisy data.

$$y(n) = \frac{1}{windowsize}(x[n] + x[n-1] + \cdots + x[n-(windowsize-1)])$$

It slides a window of length $windowsize$ along the data, computing averages of the data contained in each window.

e.g. For a windowsize of 3

$$y[n] = \frac{1}{3}(x[n] + x[n-1] + x[n-2])$$

2-2. Matlab implementation of Moving-Average Filter

Step 1.

Create a 1-by-100 row vector of sinusoidal data that is computed by random noise.

```
t = linspace(-pi, pi, 100);
rng default % initialize random number generator
x = sin(t) + 0.25 * rand(size(t))
```

Step 2.

For a windowsize of 5, compute the numerator and denominator coefficients for the rational transfer function.

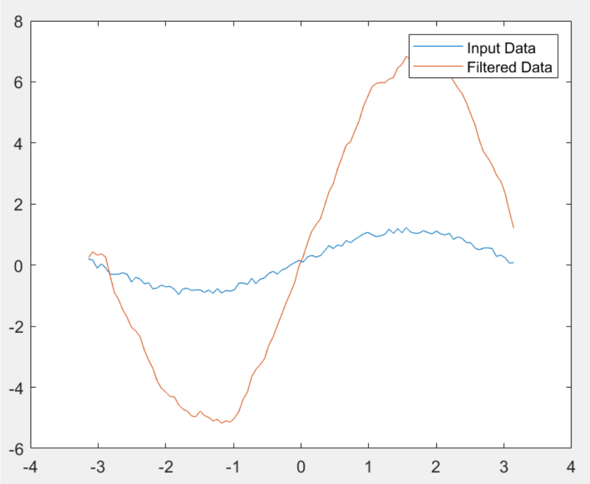
```
windowSize = 5;
b = (1/windowSize) + ones(1, windowSize);
a = 1;
```

Step 3.

Find the moving average of the data and plot it against the original data.

```
y = filter(b, a, x);
plot(t, x);
hold on;
plot(t, y);
legend("Input Data", "Filtered Data");
```

(result)



Ⅱ. Exercises

In this part, there are 2 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/2_convolution

Example 2.11

Given the following difference equation

$$y(n) - y(n-1) + 0.9y(n-2) = x(n);\ \forall n$$

a. Calculate and plot the impulse response h(n) at n = -20, … , 100.

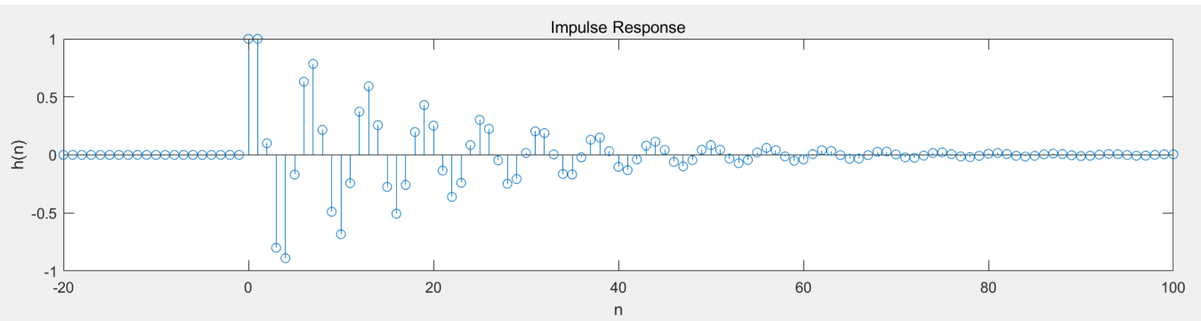
(Matlab code)

편집기 - C:₩Users₩choig₩Documents₩MATLAB₩example2_11_a.m

conv_m.m | example2_10.m | example2_11_a.m

```
b = [1]; a = [1, -1, 0.9]; n = [-20:100];
h = impz(b, a, n);
subplot(2, 1, 1); stem(n, h);
title("Impulse Response"); xlabel("n"); ylabel("h(n)")
```

(Result)



b. Calculate and plot the unit step response s(n) at n = -20, … , 100.

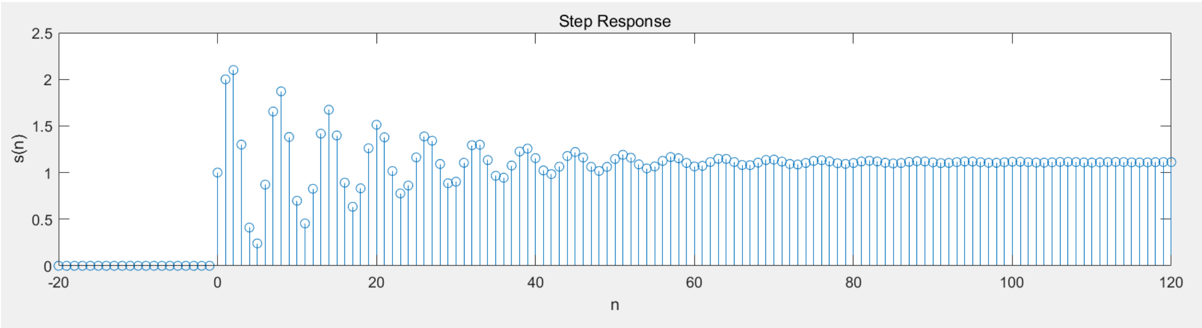
(Matlab code)

편집기 - C:₩Users₩choig₩Documents₩MATLAB₩example2_11_b.m

example2_11_b.m

```
b = [1]; a = [1, -1, 0.9]; n = [-20:120];
x = stepseq(0, -20, 120); s = filter(b, a, x);
subplot(2, 1, 1); stem(n, s);
title("Step Response"); xlabel("n"); ylabel("s(n)")
```

(Result)



c. Is the system specified by h(n) stable?

(Matlab code)

To test the stability of a signal, we have to determine h(n) for all n. For n > 120, h[n] is practically zero. Therefore the summation of h[n] can be determined using Matlab.

편집기 - C:₩Users₩choig₩Documents₩MATLAB₩example2_11_b.m

example2_11_b.m

```
b = [1]; a = [1, -1, 0.9]; n = [-20:120];
h = impz(b, a, n);
subplot(2, 1, 1); stem(n, h);
sum(abs(h))
```

(Result)

명령 창

```
>> example2_11_b

ans =

   14.8785

>>
```

Example 2.12

Let us consider the convolution given in Example 2.7. The input sequence is of finite duration

$$x(n) = u(n) - u(n - 10)$$

while the impulse response is of infinite duration

$$h(n) = (0.9)^n u(n)$$

Determine $y(n) = x(n) * h(n)$.

(Matlab code)

```
b = [1]; a = [1, -0.9];
n = -5:50; x = stepseq(0, -5, 50) - stepseq(10, -5, 50);
y = filter(b, a, x);
subplot(2, 1, 2);

stem(n, y); title("Output sequence")
xlabel("n"); ylabel("y(n)"); axis([-5, 50, -0.5, 8])
```

(Result)