임베디드 신호처리 실습

Convolution 실습 결과보고서

전자공학부 임베디드시스템

2014146004 김 민 섭

1-1 . 임의의 두 이산신호를 입력받아 컨볼루션 연산하는 함수를 만드시오.

(fliplr() 함수와 prod() 함수를 이용할것)

위 실습을 진행함에 있어서 고민해야 할 부분이 몇가지 있었다.

- **시간축의 시작점과 끝점을 설정하여 어떻게 시간축 출력을 설정할지?**
- **fliplr() 함수와 prod() 함수를 어떠한 방식으로 사용할지?**
- **두 입력 신호의 길이가 다를 경우 어떻게 연산을 진행 할 것인지?**
- **컨볼루션 연산을 위한 쉬프트 방식은 어떻게 구현할 것인지?**

크게 위 4가지가 컨볼루션 함수를 구현하는데 해결야하 할 것들이었다. 이를 어떻게 해결하였는지에 대해서 살펴보도록 하겠다. 빠른 이해를 돕기 위하여 함수의 입출력을 설명하자면 다음과 같다.

입력	출력
x , x_n : 첫번째 입력에 대한 시간과 신호	y, num : 출력에 대한 시간과 신호
h, h_n : 두번째 입력에 대한 시간과 신호	

- 시간축의 시작점과 끝점 설정하여 시간 축 출력 설정하기.

이 문제를 해결하기 위해서 우리는 두 입력신호 시간의 최소 값과 최대 값을 이용하기로 했다. 최소 값 두개를 더하여 시작하는 점을 설정하고 최대 값 두개를 더하여 끝점을 설정하는 것이다. 최소 값과 최대 값 사이값들을 구하기 위하여 개수를 센다. 그리고 linspace() 함수를 사용하여서 시간 축 출력 백터를 설정하였다. 구체적인 구현 코드는 다음과 같다.

```
start_of_conv = min(x_n) + min(h_n)
end_of_conv = max(x_n) + max(h_n)
num_of_conv = length(x) + length(h) -1
num = linspace(start_of_conv ,end_of_conv , num_of_conv)
```

위와 같은 방식으로 시간 축 출력 백터를 생성할 수 있었다.

- 두 입력 신호의 길이가 다를 경우 두 입력 신호 길이 맞춰주기.

입력신호의 값 두개의 길이가 다를 경우에는 매틀랩에서 연산이 되지 않는다. 그렇기 때문에 길이를 맞춰주는 작업을 진행하였다. 방법은 간단했다. 양 신호에 서로의 길이만큼 제로폴딩 시켜주는 것이다. 즉 x 신호에는 h의 길이 만큼의 '0' 값을 추가해 주고 h 신호에는 x 의 길이 만큼의 '0'을 추가 시켜 주었다. 구제척인 구현 코드는 다음과 같다.

```
x_num=length(x);
h_num=length(h);
X=[x, zeros(1,h_num)];
H=[h, zeros(1,x_num)];
```

- 주어진 두 함수 fliplr() 과 prod()를 컨볼루션 연산에 사용하기.

주어진 두 함수 중 fliplr()을 어디에 사용해야 할지에 대해서는 큰 문제가 없었다.

$$x1[n] * x2[n] = \sum_{k=-\infty}^{\infty} x1[k] * x2[n-k]$$

다음 컨볼루션 기본식에서 두번째 신호에 사용하게 되면 n-k 를 해줄 필요 없이 연산이 가능하다. prod() 함수는 두 신호를 2행 행렬의 각 행에 집어 넣은 후에 사용하면 1차적인 컨볼루션 연산을 수행 할 수 있게 되어서 이때 사용 하였다. 이후 sum() 함수를 사용하여 그 값들을 더해 주면 한개의 출력 신호 값이 산출 된다. 구체적인 구현 코드는 다음과 같다.

```
fH= fliplr(H);          // fh 는 이후 HT에 들어가게 된다.
A= [XT;HT];             // 여기서의 XT,HT는 shift한 후의 X, H 값이다.
B = prod(A);
YT(k) = sum(B);
```

위의 과정을 for 문 내에서 진행하게 하여 컨볼루션 연산을 진행할 수 있도록 하였다.

- 컨볼루션 연산을 위한 쉬프트 방식 구현하기.

컨볼루션 연산은 한칸씩 밀리면서 연산을 진행한다고 생각 해 볼 수 있다. 그렇기 때문에 쉬프트가 필요하다. 그리고 행렬 연산이기 때문에 각 백터의 개수 조정은 필수적이다. 이를 구현 하기 위해서 제로폴딩을 진행시켜 주었다. 우리는 쉬프트 되어서 움직이는 신호를 X 로 설정하고 가만히 있는 신호를 fH 로 설정하였다. 움직이는 신호 X 에는 앞부분과 뒷부분 모두에 제로폴딩을 진행시켜 주었다. 움직이지 않는 fH 신호에는 X 와 길이를 맞춰주기 위한 뒷 부분에만 제로폴딩을 진행시켜 주었다. 구체적인 구현 코드는 다음과 같다.

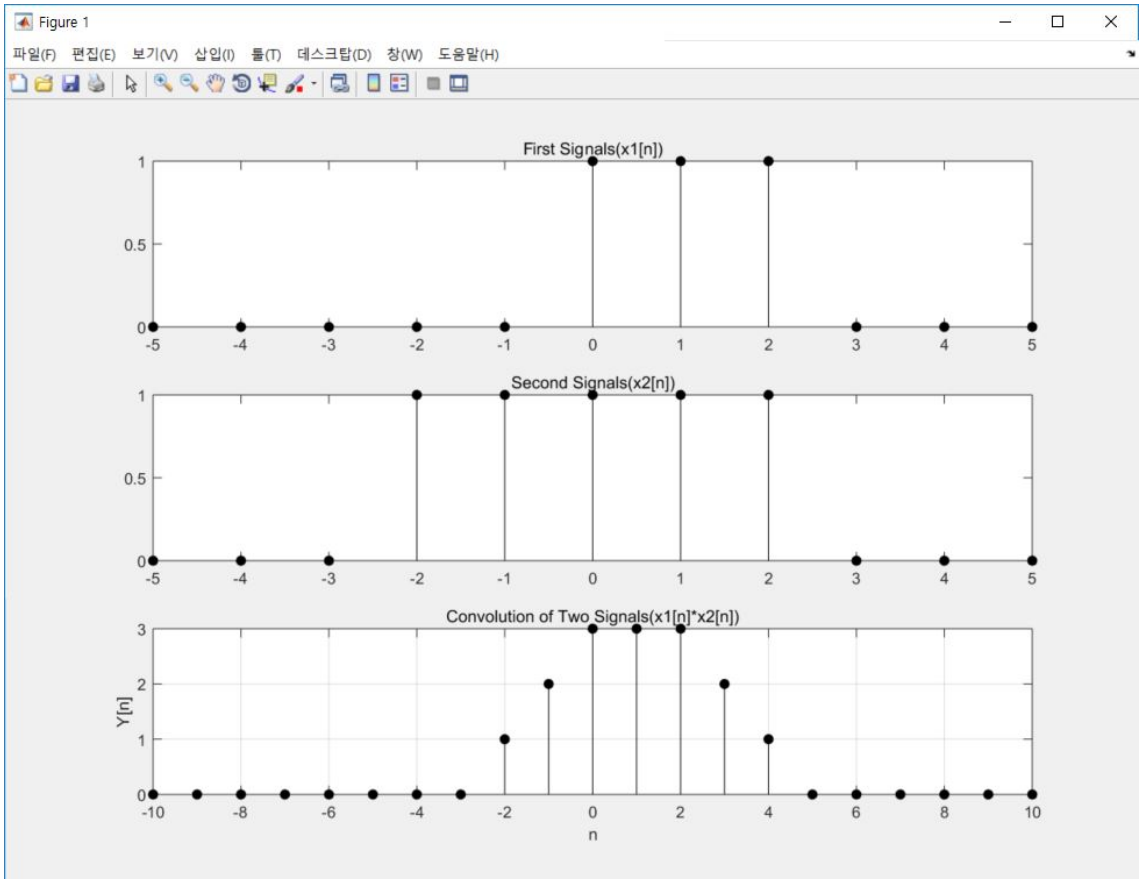
```
for k=1:num_of_conv
    YT(k) = 0;
    XT =[zeros(1,k), X, zeros(1,h_num-k)];

    if (length(XT)-length(X))>0    //첫 시작점 이후에 if 문이 참이게 된다.
       XT_num = length(XT)-length(X);
       HT = [fH, zeros(1,XT_num)];
    else
       HT = fH;       // 처음 시작에는 뒤에 제로페딩을 해줄 필요가 없다.
    end
```

위의 4가지 고민을 해결하면서 임의의 두 이산신호를 입력받아서 컨볼루션 연산을 수행하는 함수를 만들 수 있었다. 최종적으로 완성된 코드는 보고서 제일 마지막에 첨부 하였다.

1-2. 다음 두 신호의 컨볼루션 연산 결과를 구하시오.

- x1[n] = {0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0} , n = -5, -4, … 4, 5
- x2[n] = {0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0} , n = -5, -4, … 4, 5

두 신호의 값을 1-1에서 만들었던 컨볼루션 함수를 사용하였다.



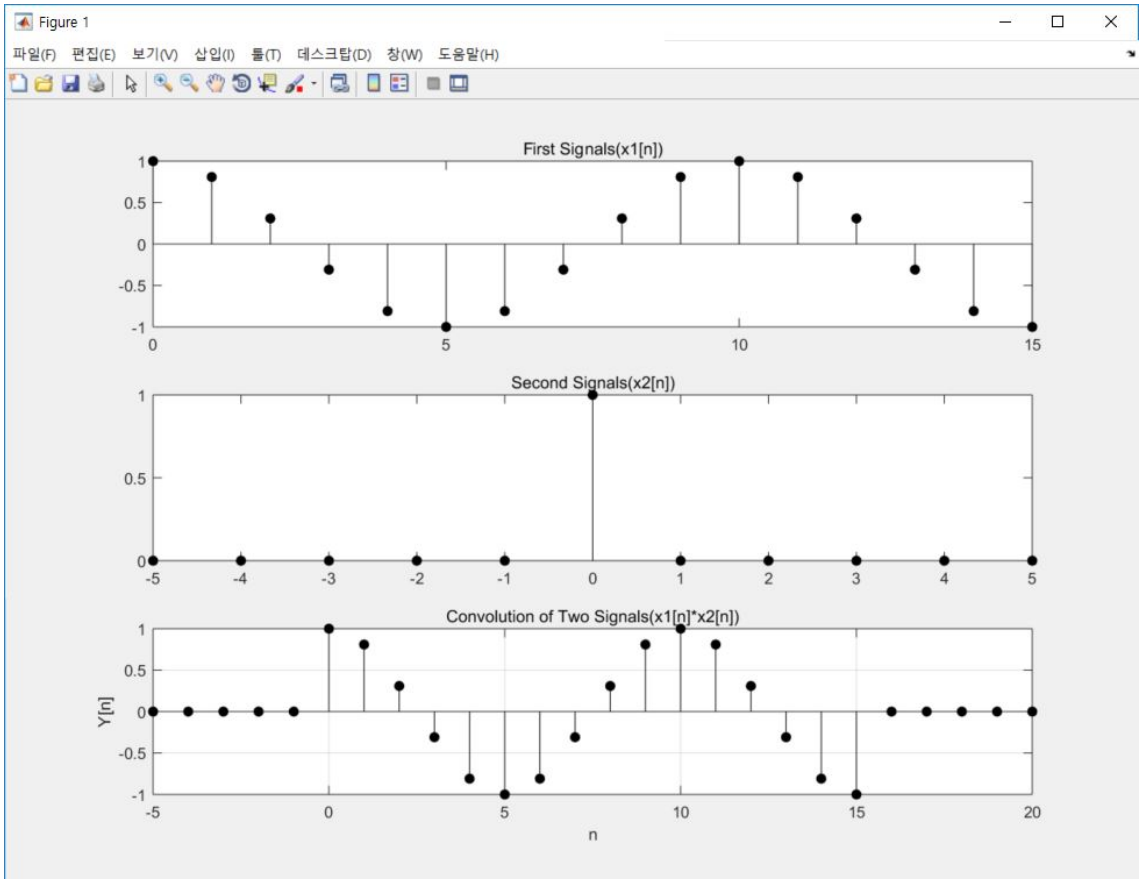
x1와 x2의 컨볼루션 결과 위 그림의 3번째 그림의 결과가 나온다.

1-3. 다음 두 신호의 컨볼루션 연산 결과를 구하시오.

-x1[n] = Acos(2πf'n + θ) , A =1 f' = 0.1, θ = 0, n = 0, 1, … , 14, 15

-x2[n] = δ[n], n = -5, -4, … 4, 5

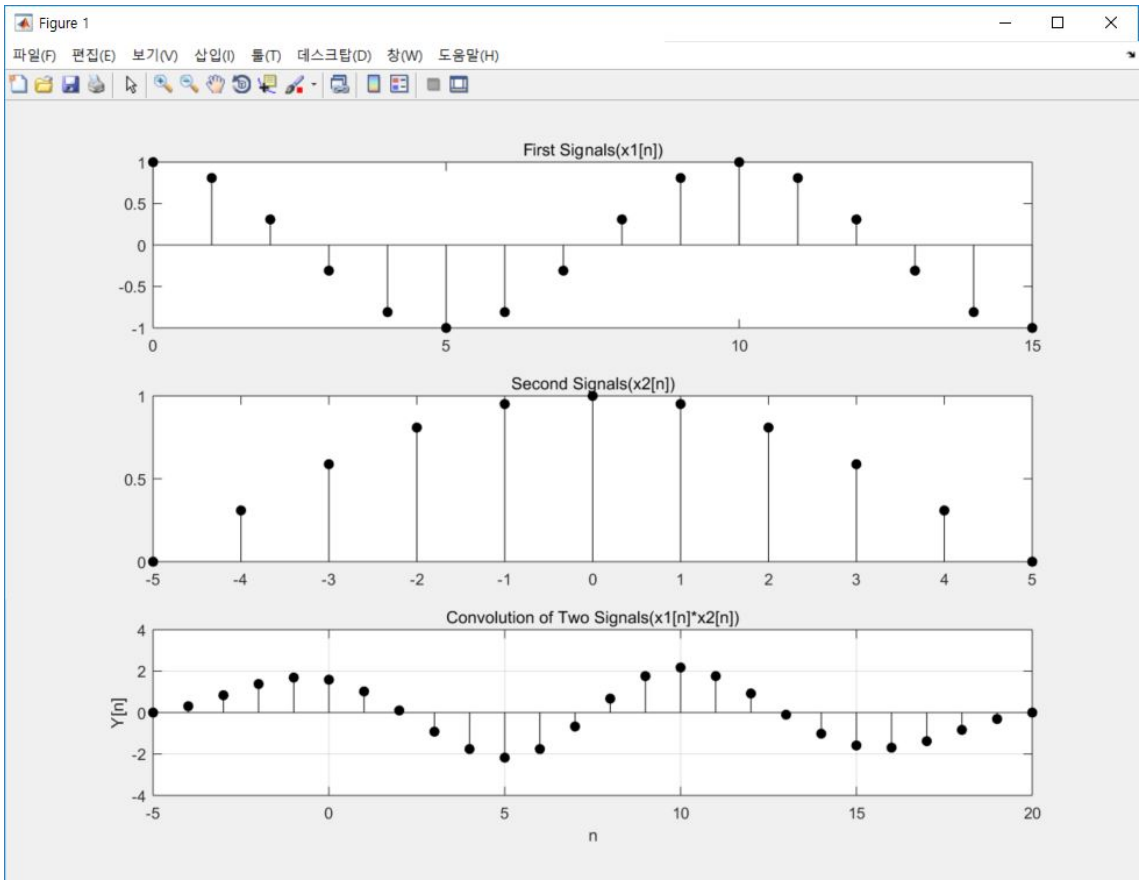
두 신호의 값을 1-1에서 만들었던 컨볼루션 함수를 사용하였다.



x1은 델타함수로 전체 구간에 걸쳐 값이 0이지만 특정한 한 점에서만 값을 가지는 함수임으로 두 함수를 컨볼루션 한 결과는 그림에서 3번째 신호가 나온다.

1-4. 다음 두 신호의 컨볼루션 연산 결과를 구하시오.

- x1[n] = Acos(2πf'n + θ) , A =1 f' = 0.1, θ = 0, n = 0, 1, … , 14, 15
- x2[n] = Acos(2πf'n + θ) , A =1 f' = 0.05, θ = 0, n = -5,-4, … , 4, 5

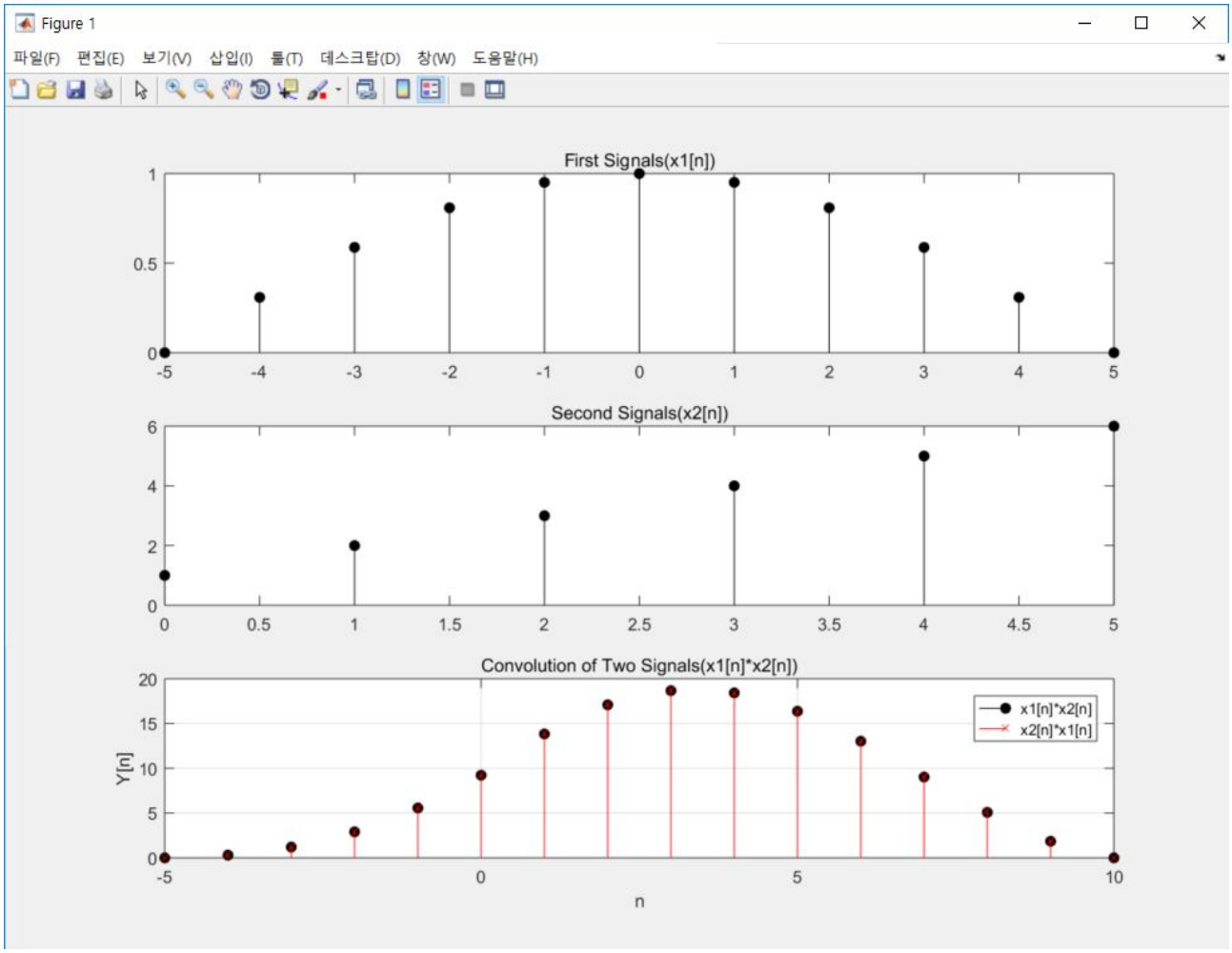


x1 의 주파수를 반으로 하고 –5 ~ 5까지 샘플링한 신호가 x2 이다. 이 두 신호를 컨볼루션을 한 결과 그림에서 맨 마지막 신호가 나온다.

2-1 . 다음 두 이산신호의 컨볼루션에 대해서 교환법칙이 성립함을 보여라.

- x1[n] = Acos(2πf'n + θ) , A =1 f' = 0.05, θ = 0, n = -5,-4, … , 4, 5
- x2[n] = { 1, 2, 3, 4, 5, 6} , n = 0, 1, 2, 3, 4, 5

교환법칙이 성립하려면 x1[n]*x2[n] = x2[n]*x1[n] 이 성립하면 된다.
성립되는 것을 보여주기 위하여 1-1에서 만든 컨벌루션 함수를 이용하여서 해보았다.

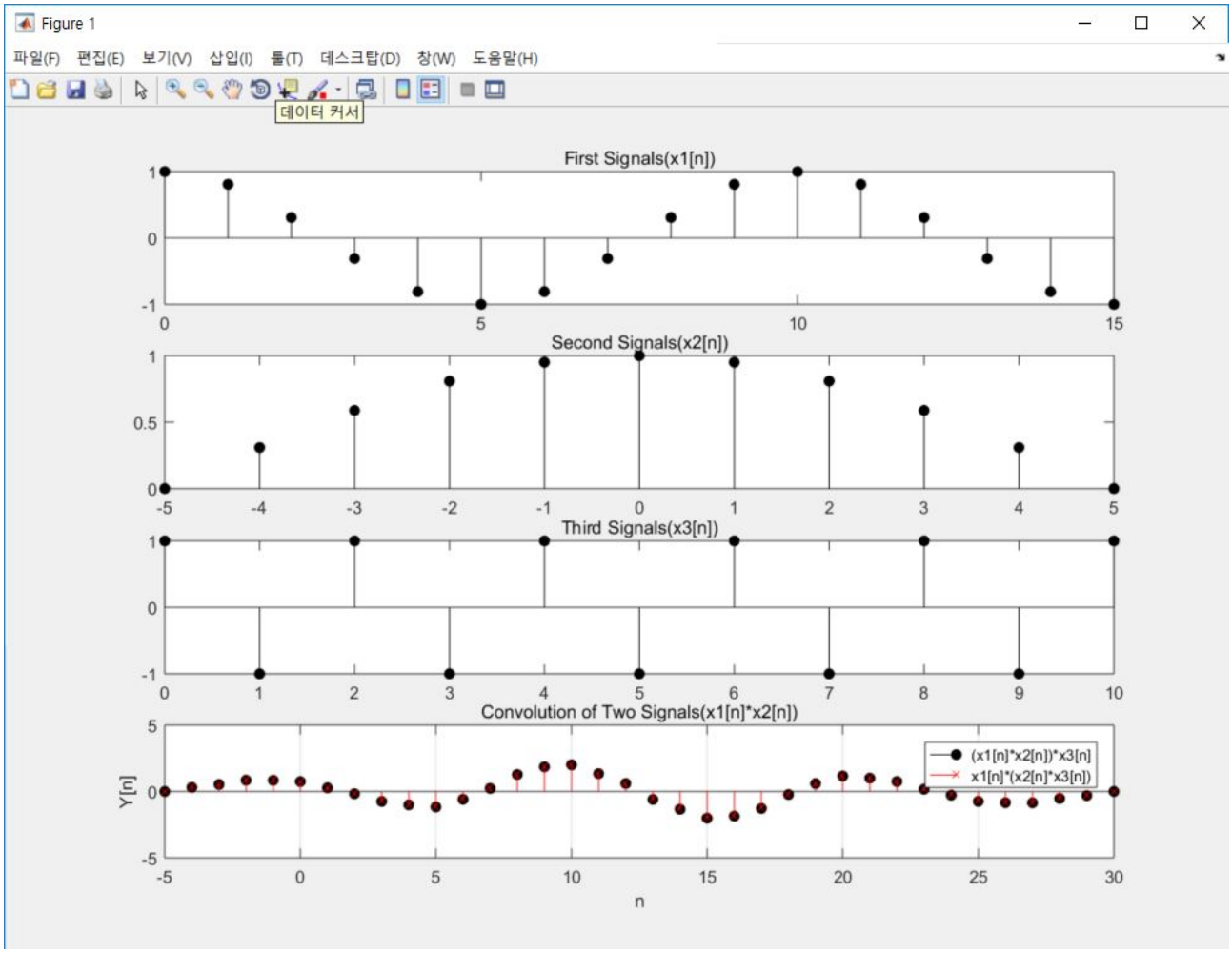


가장 밑에 있는 그래프를 보면 알듯이 컨볼루션에서 교환법칙이 성립함을 알 수 있다.

2-2 . 다음 세 이산신호의 컨볼루션에 대해서 결합법칙이 성립함을 보여라.

- x1[n] = Acos(2πf'n + θ) , A =1 f' = 0.1, θ = 0, n = 0, 1, … , 14, 15
- x2[n] = Acos(2πf'n + θ) , A =1 f' = 0.05, θ = 0, n = -5,-4, … , 4, 5
- x3[n] =$(-1)^n$, n = 0, 1, 2, … 9, 10

결합법칙이 성립함을 보이기 위해서는 (x1[n]*x2[n])*x3[n] = x1[n]*(x2[n]*x3[n]) 의 결과가 참임을 보이면 된다. 이또한 마찬가지로 1-1 에서 만들었던 컨볼루션 함수를 사용해 보았다.

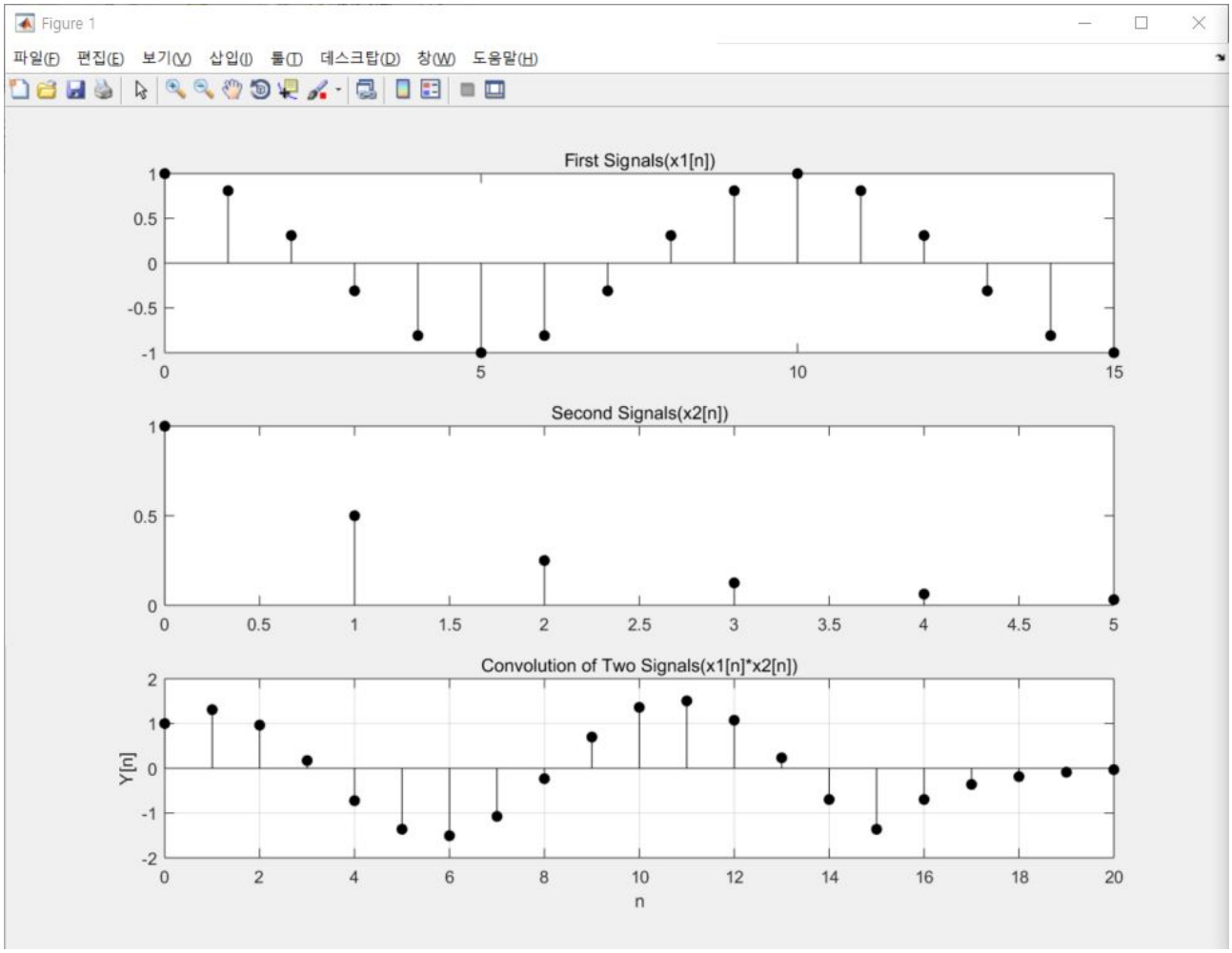


그래프 결과를 보면 알 수 있듯이 (x1[n]*x2[n])*x3[n] = x1[n]*(x2[n]*x3[n]) 값이 서로 같음을 알 수 있다. 그러므로 컨볼루션에서는 결합법칙이 성립한다고 볼 수 있다.

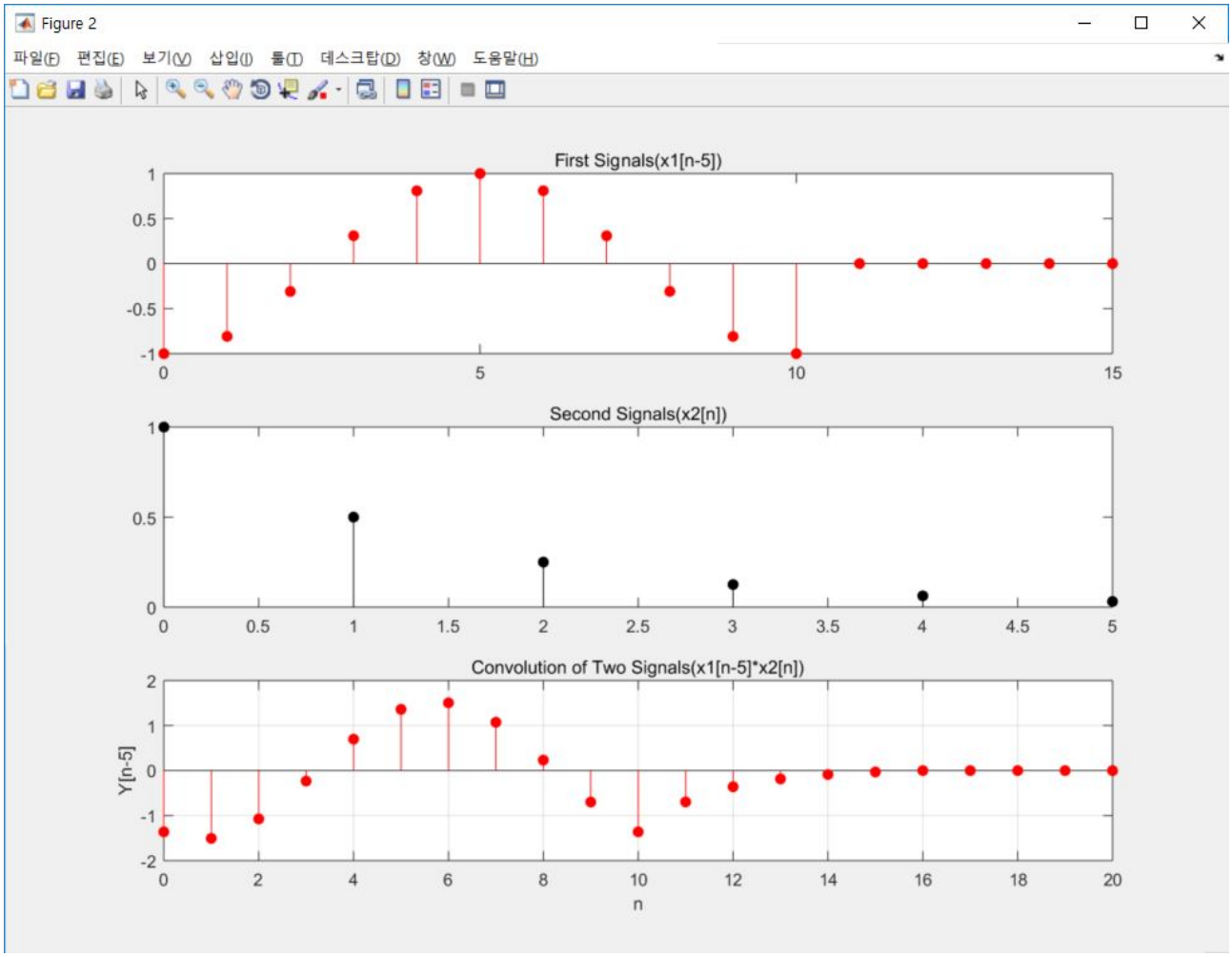
2-3 . 다음 이산신호 입력 x[n]와 임펄스응답 h[n]으로 이동성질 y[n - n0] = x[n-n0]*h[n] 이 만족함을 보여라.

- x[n] = Acos(2πf'n + θ) , A =1 f' = 0.1, θ = 0, n = 0, 1, … , 14, 15
- x[n] =$(0.5)^n$, n = 0, 1, … 4, 5

먼저 이동하기 전의 컨볼루션 연산 결과를 살펴보고자 한다. 결과는 다음과 같다.



n0 = 5라고 설정한다면 x[n-5] 의 끝점은 10 이 될것이다. 그리고 이동성질을 만족한다면 y[n-5] 의 끝점도 20이 아니라 15가 될것이다. 다음은 n0=5로 설정한 후의 결과 값이다.



결과를 보면 알 수 있듯이 x[n-5] 의 끝점은 10이 되고 y[n-5]의 끝점은 15가 되었다.
그러므로 컨볼루션 연산에서 이동성질은 성립된다고 할 수 있다.

컨볼루션 함수 구현 코드

```
function [y,num] = convo(x, h ,x_n,h_n)

    start_of_conv = min(x_n) + min(h_n);
    end_of_conv = max(x_n) + max(h_n)    ;
    num_of_conv = length(x) + length(h) -1;

    x_num=length(x);
    h_num=length(h);
    X=[x, zeros(1,h_num)];
    H=[h, zeros(1,x_num)];
    fH= fliplr(H);

    num = linspace(start_of_conv ,end_of_conv , num_of_conv);

    for k=1:num_of_conv
        YT(k) = 0;
        XT =[zeros(1,k), X, zeros(1,h_num-k)];
        if (length(XT)-length(X))>0
            XT_num = length(XT)-length(X);
            HT = [fH, zeros(1,XT_num)];
        else
            HT = fH;
        end
        A= [XT;HT];
        B = prod(A);
        YT(k) = sum(B);
    end

    y=fliplr(YT);
```