HomeWork (MATLAB Signal & System) - Part #1

2019102136 최성준

Q1 : A rectangular pulse $x(t)$ is defined by

$x(t)$ = 1, $|t| \leq 5$

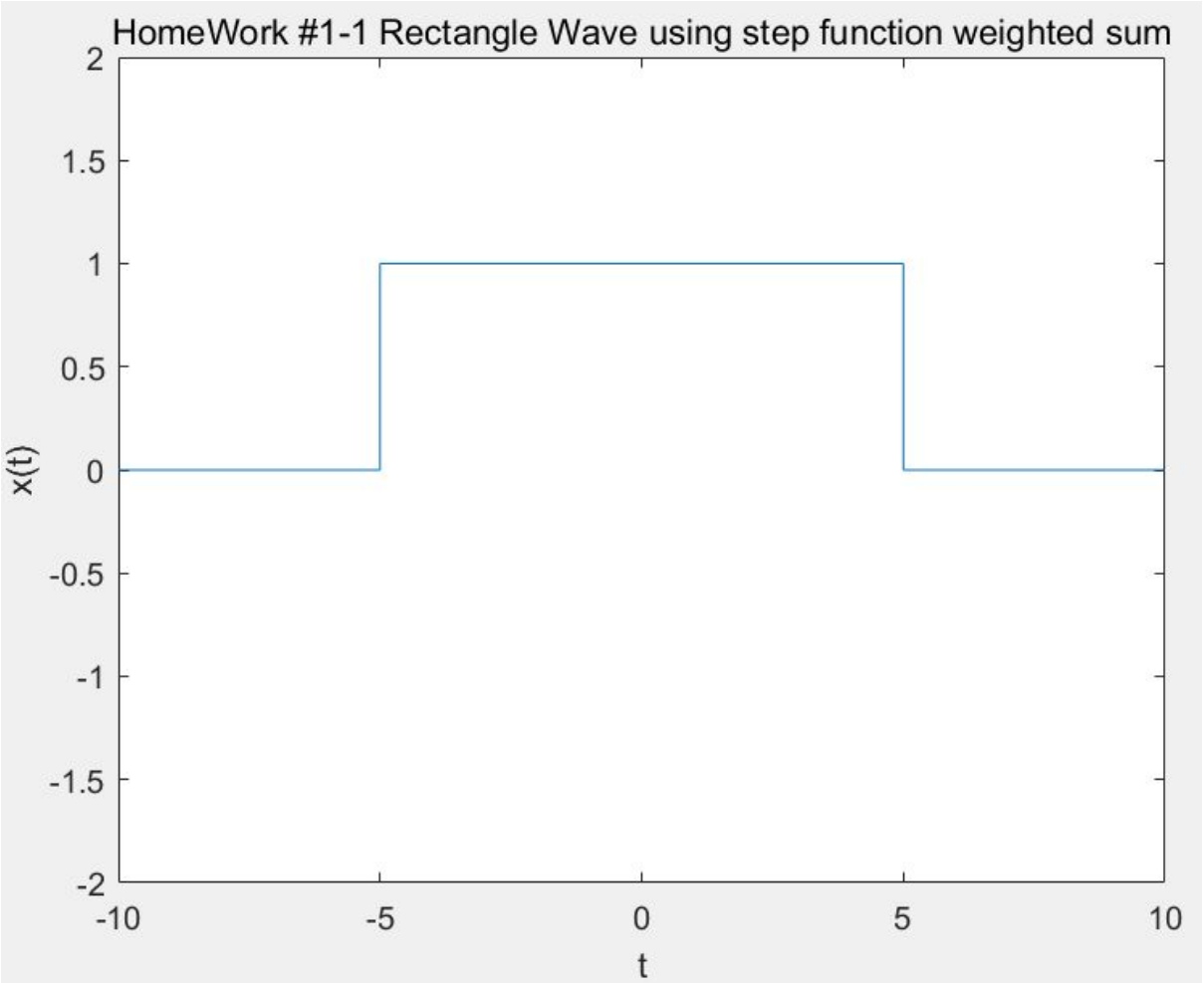
0, otherwise.

Write a MATLAB program to generate $x(t)$ for $|t| < 10$ and plot it

---------- < CODE > ----------

```
clear all
t = -10:.001:10;
u1 = heaviside(t-5);
u2 = heaviside(t+5);
u = u2-u1;

plot(t,u)
xlabel("t")
ylabel("x(t)")
title(["HomeWork #1-1 Rectangle Wave using step function weighted sum"])
axis([-10 10 -2 2])
```



분석 : 위의 조건을 만족하는 함수를 만들기 위해서 0에서 1로 변화하는 지점을 parameter로 받는 **heaviside라는 내장함수를 활용**하여 **step function을 구현한 후 weighted sum**을 통해 t값을 대입했을때 heaviside(t+5)에서 heaviside(t-5)를 뺀 값을 반환하는 plot을 구현하였다. 처음에는 단순 t의 범위만을 지정해주어 그래프를 그려주었으나 사다리꼴의 모양이 나타나서 **미세하게 sampling**을 해주어 원하는 그래프를 도출할 수 있었다.

Q2 : 2. A raised cosine sequence is defined by

$x(n) = \cos(0.1\pi n) + 1, \quad -10 \le n \le 10$

$0, \quad \text{otherwise.}$

Use MATLAB to plot $x[n]$

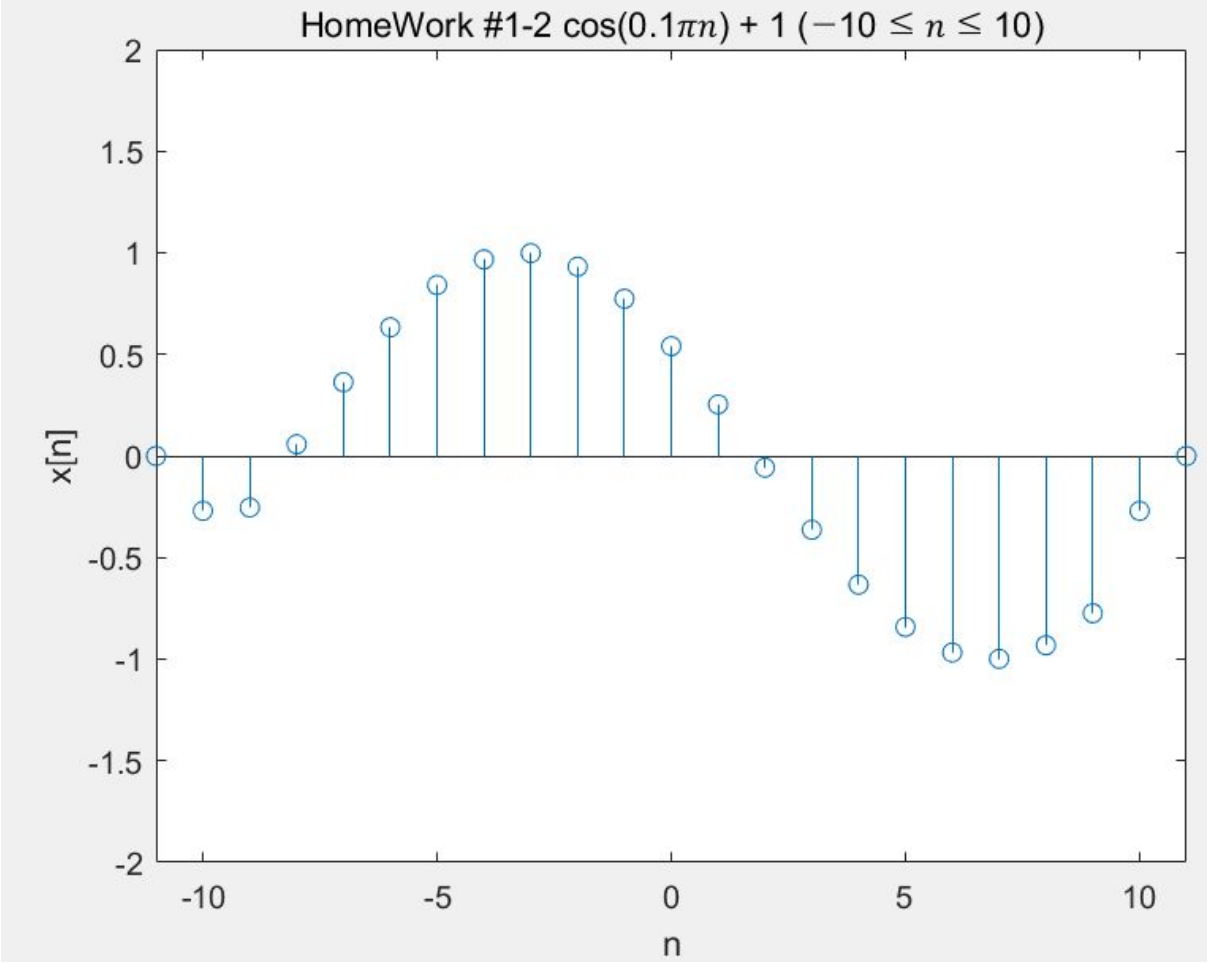
---------- < CODE > ----------

```
clear all

n = -100:100;
y = cos((0.1*pi*n) + 1);

u1 = heaviside(n-10);
u2 = heaviside(n+10);
u = u2-u1;

stem(n, u.*y)
% y에 u(step function weighted sum)를 곱해줌으로써 -10:10이외의 값은 0처리.
xlabel('n')
ylabel('x[n]')
title('HomeWork #1-2 cos(0.1πn) + 1 (−10 ≤ n ≤ 10)')
axis([-11 11 -2 2])
```



분석 : 함수는 단순 cos인이라는 내장함수를 활용하여 그릴 수 있었으나 문제는 otherwise를 구현하는 과정이었다. 이 문제를 어떻게 해결할까 생각하던 중 '**Q1**'문제를 활용하여 원하는 범위만 곱해주면 어떨까라는 생각으로 위 문제와 범위만 다른 함수를 구현하여 곱**(.*)**해주었더니 원하는 함수를 그릴 수 있었다. 이외에도 몇가지 문제가 더 있었다.

1. n의 범위를 -10:10로 설정하니 other을 구현했어도 그래프가 표현되지 않았다. 이는 조금 투박하기는 하나 단순하게 **범위(-100:100)를 넓혀주어** 해결하였다.
2. 문제를 자세히 읽어보니 continuous한 함수를 구현하는 것이 아닌 discrete한 함수를 구현하는 것이었다. 이는 plot을 사용하는 것이 아닌 parameter는 동일한 **stem이라는 함수를 통해 구현**하였다.
3. 문제에서 지정해주지 않아 해결하지 못한 찜찜한 문제도 있다. **'꼭 sampling 빈도를 정수로 해야돼나?', 'otherwise라 하면 어느정도의 범위까지 표현을 해야하는 거지?' 등** 헷갈리는 게 많았으나 이 문제를 통해 해결한 것이 더 많은 것 같다.

Q3 : Plot the following sequences using MATLAB.
a. $x[n] = \delta[n + 2] - 2\delta[n] + \delta[n - 2], -5 \le n \le 5$

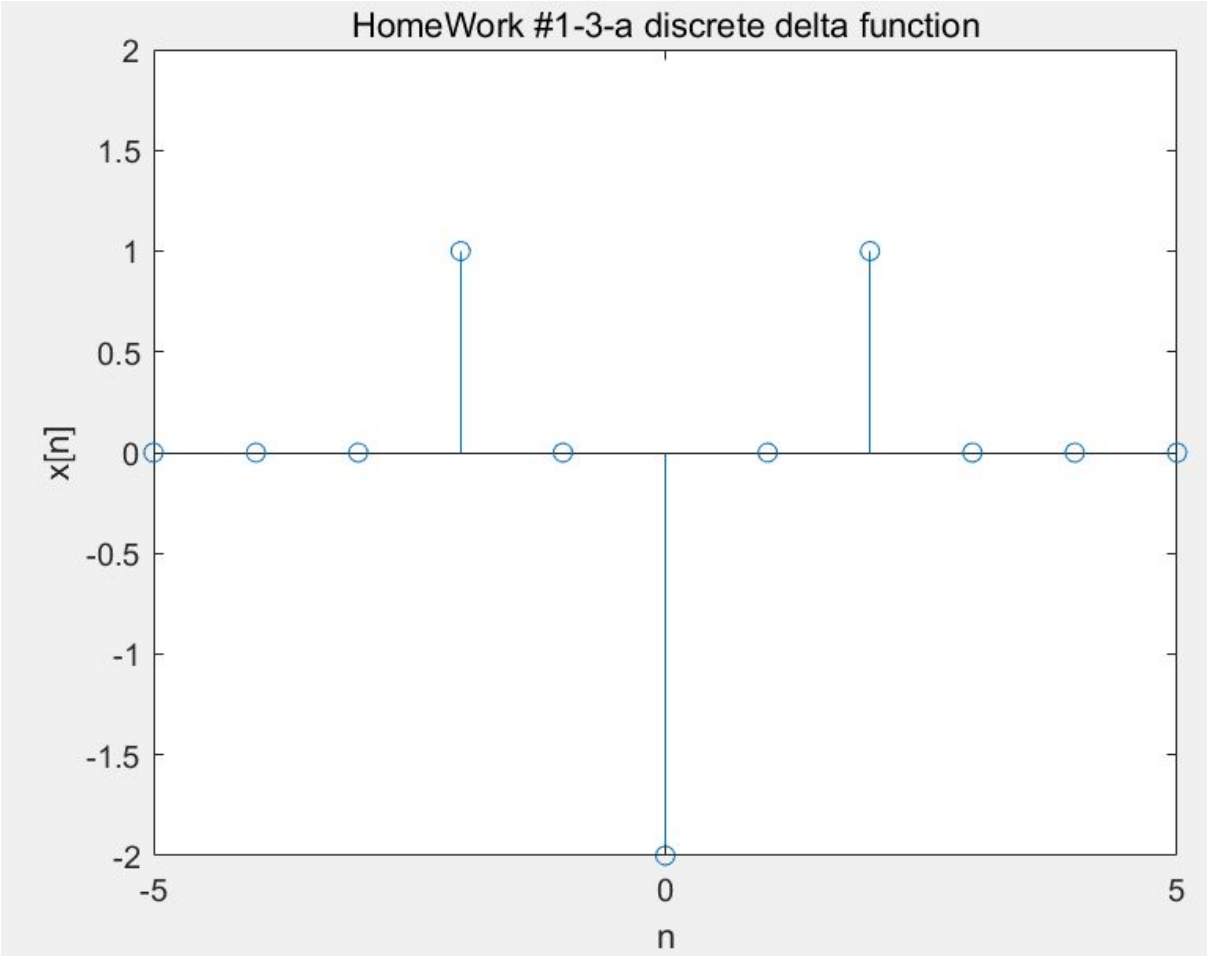
---------- < CODE > ----------

```
clear all

n = -5:5;

impulse1 = n==2;
impulse2 = n==0;
impulse3 = n==-2;
u = impulse1-2*impulse2+impulse3;

stem(n, u)
% y에 u(step function weighted sum)를 곱해줌으로써 -10:10이외의 값은 0처리.
% stem : sampling값만(discrete), plot : sampling값을 연결해서 그래프(continuous)
xlabel('n')
ylabel('x[n]')
title('HomeWork #1-3-a discrete delta function')
axis([-5 5 -2 2])
```



분석 : 처음에 또 'unit function의 합인가' 생각했는데 **delta function**이었다. 이를 단순히 값을 대입하여 구현해야 하나하고 찾다보니 impulse함수를 따로 구현을 하는 방법이 있어 참고하여 구현한 후 원하는 범위와 discrete으로 바꿔주었다.
그러고나서 소스코드를 검토하다가 위의 impulse함수를 구현하는 방법이 단순 대입이었다는 생각이 들었다. **impulse = n==0;**이렇게 impulse라고 선언해주어 따로 구현하는 방법인 줄 알았으나 다른 분이 임시로 지정한 변수가 impulse였던 것이었다.

Q3 : Plot the following sequences using MATLAB.
b. $x[n] = u[n + 5] + u[n + 3] + u[n + 1] - u[n - 2] - u[n - 4] - u[n - 6], -10 \leq n \leq 10$

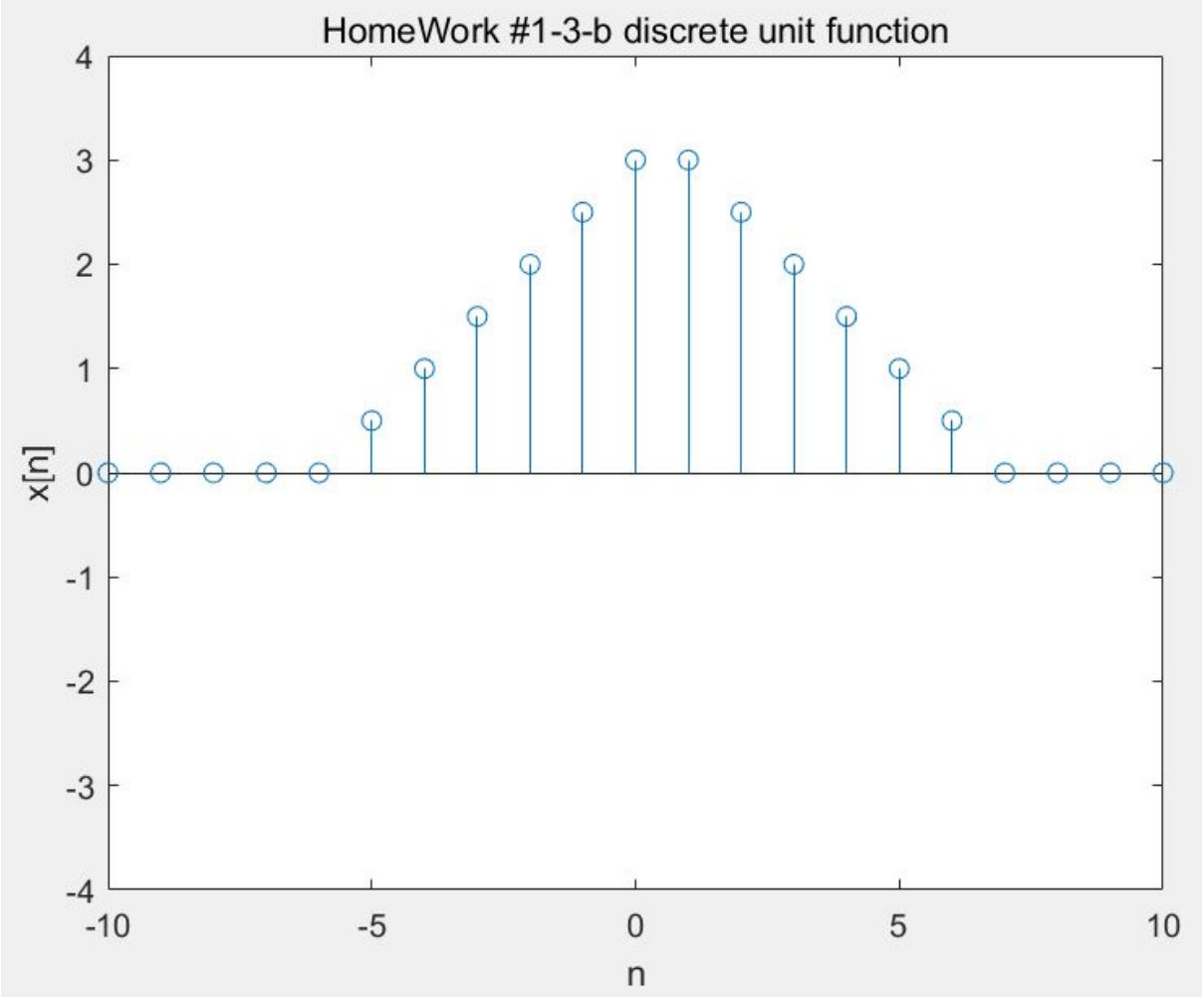
---------- < CODE > ----------

```
clear all

n = -10:10;

u1 = heaviside(n+5);
u2 = heaviside(n+3);
u3 = heaviside(n+1);
u4 = heaviside(n-2);
u5 = heaviside(n-4);
u6 = heaviside(n-6);
u = u1+u2+u3-u4-u5-u6;

stem(n, u) % y에 u(step function weighted sum)를 곱해줌으로써 -10:10이외의 값은
0처리.
% stem : sampling값만(discrete), plot : sampling값을 연결해서 그래프(continuous)
xlabel('n')
ylabel('x[n]')
title('HomeWork #1-3-b discrete unit function')
axis([-10 10 -4 4])
```



분석 : 이 문제는 위 '**Q1**', '**Q2**'에서 구현한 step function이 많이 weighted sum되어있는 것이다. 단순히 문제를 구현하고 보니 **discrete한 그래프가 값이 급격히 바뀌는 부분에서 양쪽 값의 평균값을 취하는 점**이었다. 이는 그 부분만 약간 비껴지나가도록 sampling하거나 그 값만 제외하여 sampling하면 해결되긴하지만 그런 식으로 sampling한 것이 진짜 그래프의 형태라고 생각하여, 위와 같은 code를 유지하였다.

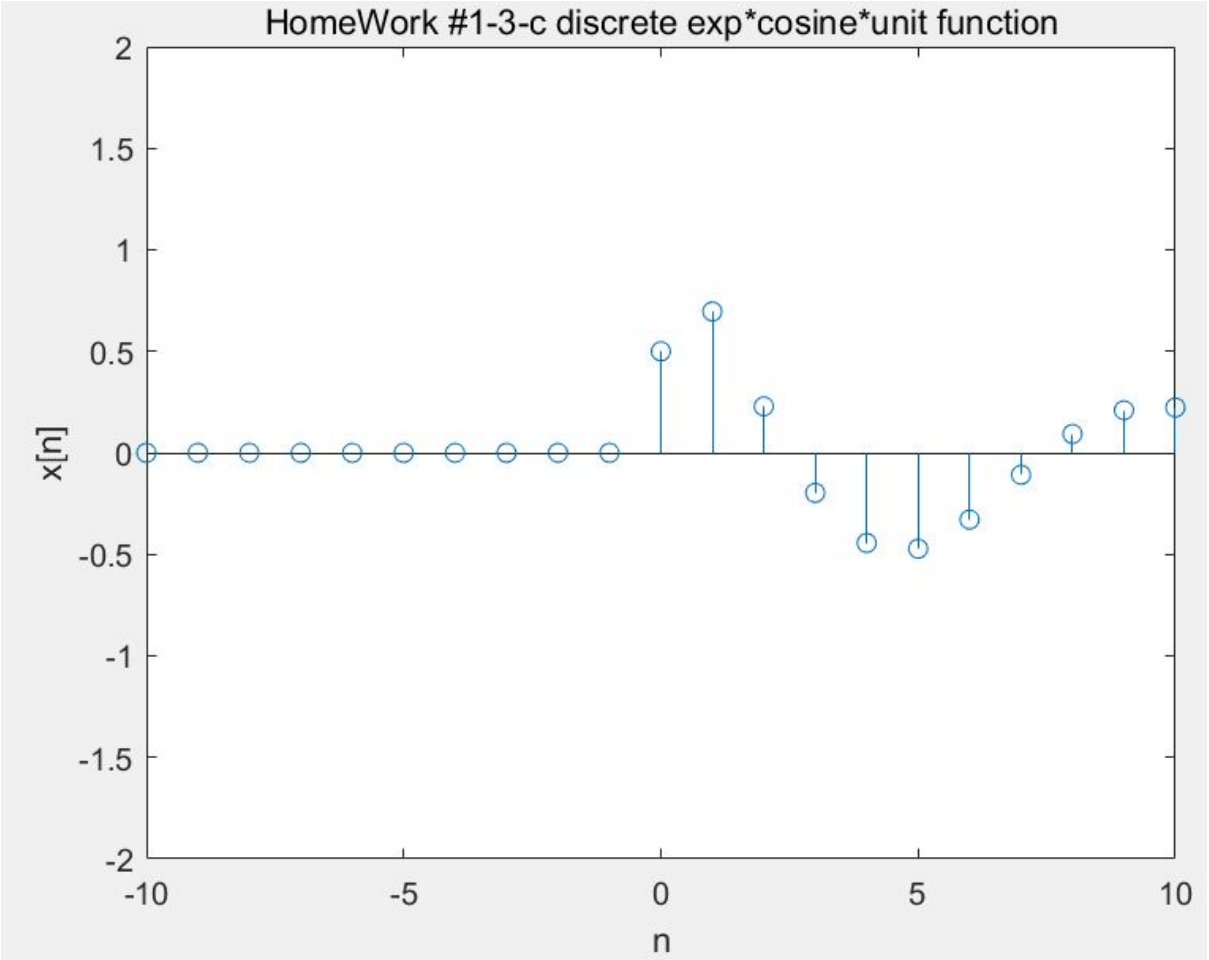
Q3 : Plot the following sequences using MATLAB.
c. $x[n] = e^{-0.15n}\cos(0.2\pi n)\,u[n], -10 \le n \le 10$

---------- < CODE > ----------

```
clear all

n = -10:10;
u1 = heaviside(n);
u = exp(-0.15*n).*cos(0.2*pi*n).*u1;

stem(n, u) % y에 u(step function weighted sum)를 곱해줌으로써 -10:10이외의 값은
0처리.
% stem : sampling값만(discrete), plot : sampling값을 연결해서 그래프(continuous)
xlabel('n')
ylabel('x[n]')
title('HomeWork #1-3-c discrete exp*cosine*unit function')
axis([-10 10 -2 2])
```



분석 : 어찌보면 위와 동일한 문제이지만 여기서는 어렵게 구성한다는게 더 많은 힌트를 준 것 같다. '**Q2**'에서는 '어떻게 함수 내의 특정범위의 만을 살리지?'라는 생각에 고민을 했었는데, 이 문제는 step function을 곱하라고 문제 내에서 지정해 주어 더욱 쉽게 구현할 수 있었다. 지수함수를 구현하기 위해서는 **exp()라는 내장함수**를 활용하였고, cos함수를 구현하기 위해서는 **cos()라는 내장함수**를 활용하였다. 이후 각각을 곱하는 과정에서 위의 문제에서도 사용했었지만 **행렬 곱셈이 아닌 요소별 곱셈을 수행하기 위해 .* 연산자를 활용**하여 최종적인 x[n]을 구현하였다.