سیگنال‌ها و سیستم‌ها - دکتر سلیمی‌بدر

امیرحسین منصوری - ۹۹۲۴۳۰۶۹ - تمرین کامپیوتری سری ۱

سوال ۱

ابتدا توابع $u[n]$ و $\delta[n]$ را پیاده سازی می‌کنیم. ورودی و خروجی این توابع، آرایه‌ای ۱ در n است. برای پیاده‌سازی $u[n]$، خروجی را برابر آرایه‌ای که همه مقادیر آن صفر است قرار می‌دهیم و سپس به ازای هر مقداری در ورودی که بزرگتر یا مساوی ۱ باشد، مقدار ۱ در خروجی قرار می‌دهیم. (تابع size، ابعاد ورودی‌اش را برمیگرداند)

```
function out = unitstep(n)
    out = zeros(size(n));
    out(n >= 0) = 1;
end
```

پیاده‌سازی $\delta[n]$ نیز مشابه سیگنال بالا است، با این تفاوت که به ازای هر مقداری در ورودی که دقیقا برابر صفر باشد، خروجی را ۱ می‌کنیم.

```
function out = unitimpulse(n)
    out = zeros(size(n));
    out(n == 0) = 1;
end
```

حال توابع $x[n]$، $y[n]$ و $z[n]$ را پیاده می‌کنیم:

```
function out = z(n)
    out = cos((2*pi).*n).*x(n);
end

function out = y(n)
    out = 2.*x(n) - x(2.*n);
end

function out = x(n)
    out = unitstep(n + 3) - unitstep(n - 3) + 2.*unitimpulse(n + 3) + 3.*unitimpulse(n + 2)
end
```

این توابع نیز آرایه‌ای به ابعاد ۱ در n گرفته و آرایه‌ای به همین ابعاد را در خروجی می‌دهند. همچنین عملگرهای استفاده شده در این توابع، روی تک تک عناصر آرایه به صورت نظیر به نظیر اجرا می‌شوند. مثلا

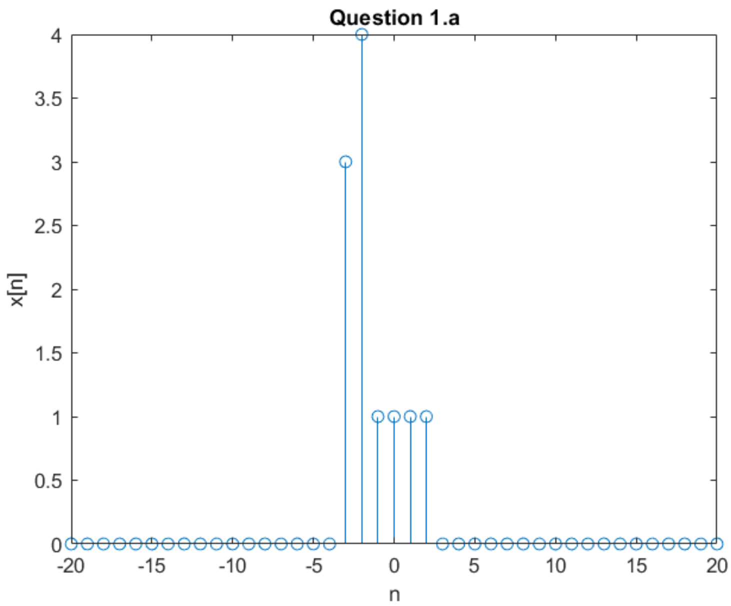
cos(x).*x(n)

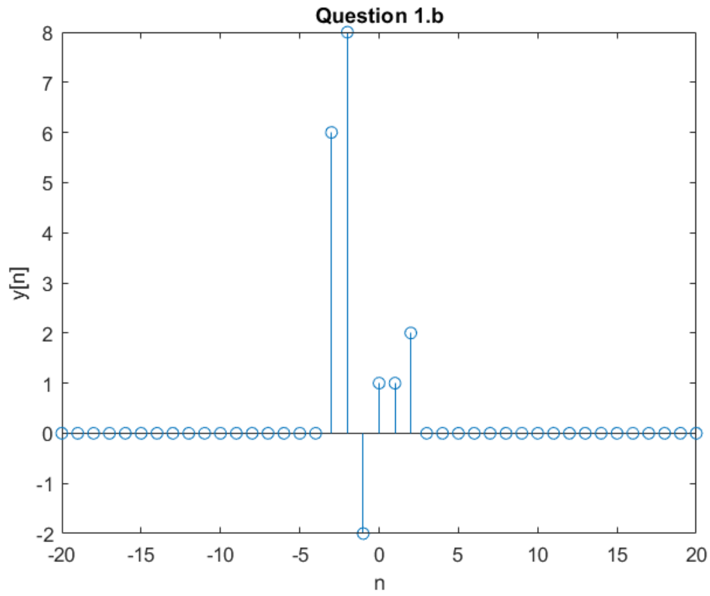
اعضای آرایه‌ای که cos(n) برمی‌گرداند را تک تک در اعضای آرایه‌ای که x(n) می‌دهد ضرب می‌کند و آرایه‌ای جدید می‌دهد (بر خلاف عملگر * که برای ضرب ماتریس استفاده می‌شود).

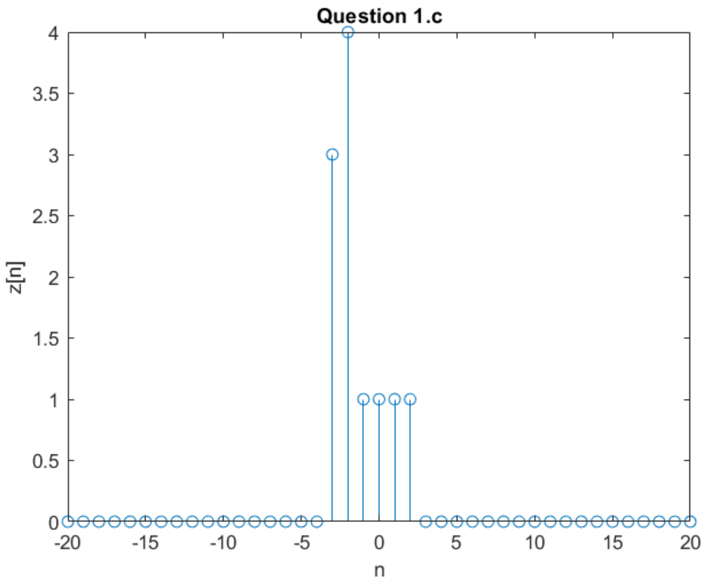
حال برای رسم نمودار، به صورت زیر عمل می‌کنیم.

```
figure;

n = -20:20;
stem(n, x(n));

xlabel('n');
ylabel('x[n]');
title('Question 1.a');
```

ابتدا با استفاده از دستور figure، یک پنجره جدید برای نمودار می‌سازیم. سپس یک آرایه از ۲۰- تا ۲۰+ به نام n می‌سازیم. سپس این آرایه و آرایه‌ای که x[n] بر می‌گرداند را به stem می‌دهیم تا نمودار آن رسم شود.
همچنین در ادامه، عنوانی که کنار محور x و y و همچنین بالای نمودار قرار داده می‌شود را نیز مشخص می‌کنیم.
کد مربوط به رسم $y[n]$ و $z[n]$ نیز دقیقا مشابه کد بالا است. خروجی مربوط به این ۳ سیگنال نیز به صورت زیر است:





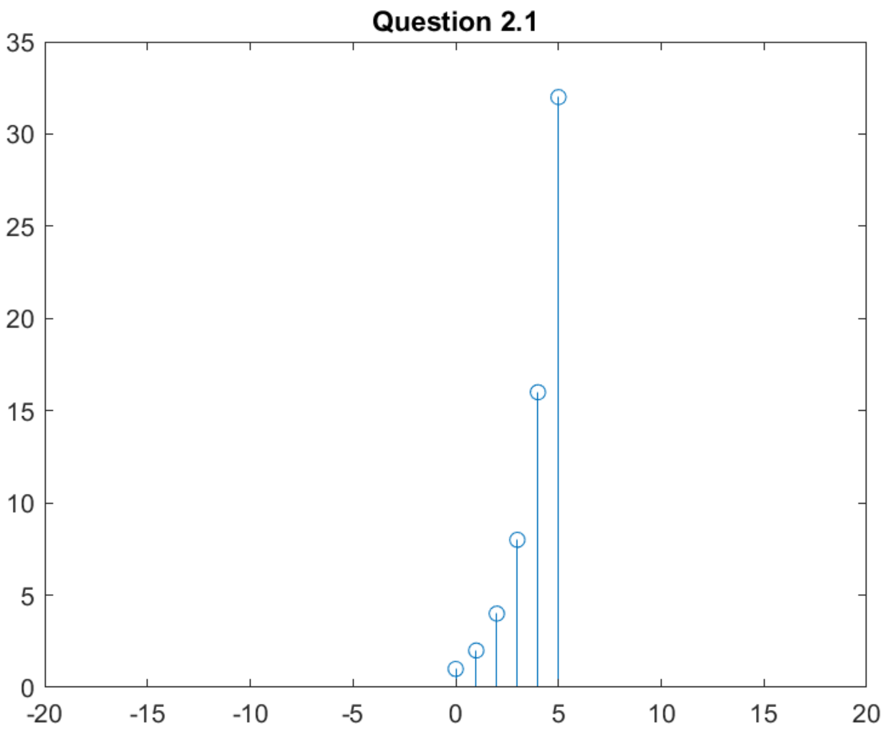


سوال ۲ - الف)

این سیگنال همان سیگنال $x[n] = 2^n$ است که در بازه $[0, 5]$ رسم شده است. همچنین صفحه نمودار، بازه $-20 \leq x \leq 20$ را نشان می‌دهد. با استفاده از کد زیر سیگنال را رسم می‌کنیم.

```
figure;

n = 0:5;
stem(n, 2.^n);

xlim([-20, 20]);
title('Question 2.1');
```

در بالا، مقدار x ها را آرایه‌ای از صفر تا ۵ و مقدار y ها را برابر ۲ به توان تک تک اعداد صفر تا ۵ در نظر می‌گیریم. همچنین با استفاده از xlim، بازه‌ی نمایش داده شده در نمودار را از ۲۰- تا ۲۰+ در نظر می‌گیریم. همچنین عنوان نمودار را در خط آخر مشخص می‌کنیم. نمودار به شکل زیر در خواهد آمد:

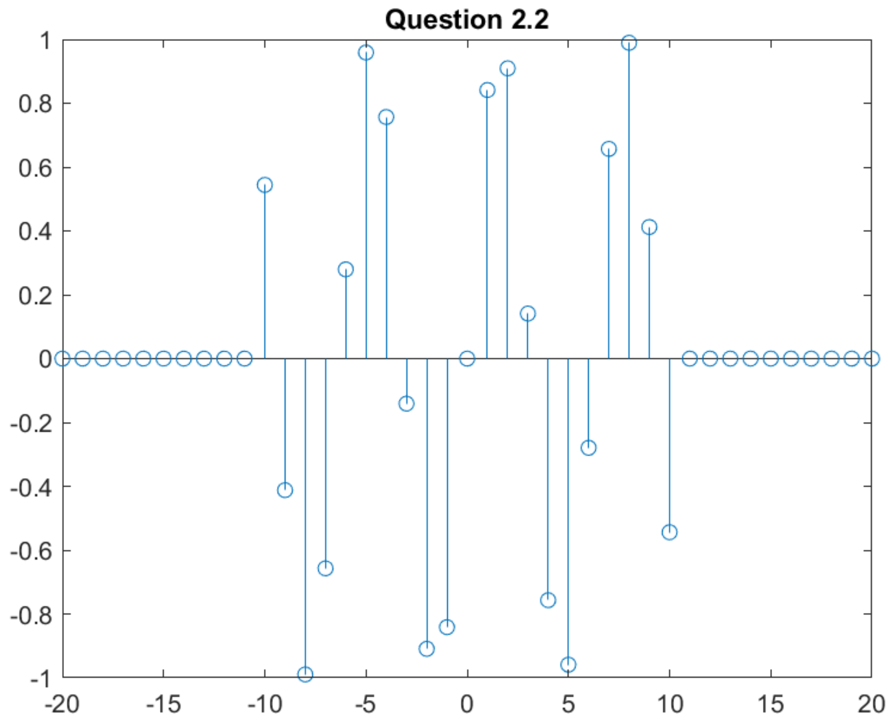


سوال ۲ - ب)

می‌توان حدس زد که سیگنال مشخص شده در شکل، همان سیگنال $\sin[n]$ است که در بازه ۱۰- تا ۱۰+ رسم شده است. همچنین از بازه ۱۱ تا ۲۰ و ۲۰- تا ۱۱- نیز مقدار سیگنال برابر صفر است. برای رسم سیگنال از کد زیر استفاده می‌کنیم.

```
figure;

stem(-20:20, [zeros(1, 10), sin(-10:10), zeros(1, 10)]);

xlim([-20, 20]);
title('Question 2.2');
```

در تابع stem، مقدار x ها را از ۲۰- تا ۲۰+ مشخص می‌کنیم و برای مقدار y ها، آرایه‌ای از ۱۰ صفر را به اول و آخر سیگنال $\sin[n]$ اضافه می‌کنیم. سینتکس $[A,\ B]$، دو (یا چند) آرایه را به دنبال هم اضافه می‌کند (یا concatenate می‌کند). در نهایت، بازه نشان داده شده توسط نمودار و عنوان را نیز مشخص می‌کنیم. نمودار نهایی به شکل زیر در می‌آید.



سوال ۳ - الف)

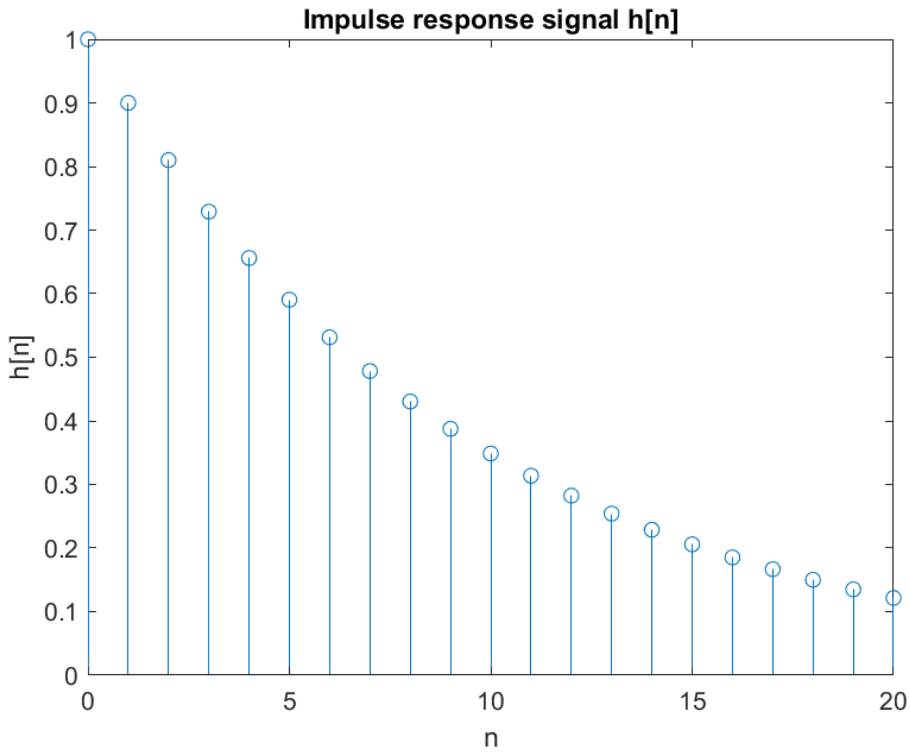
توابع را تعریف می‌کنیم.

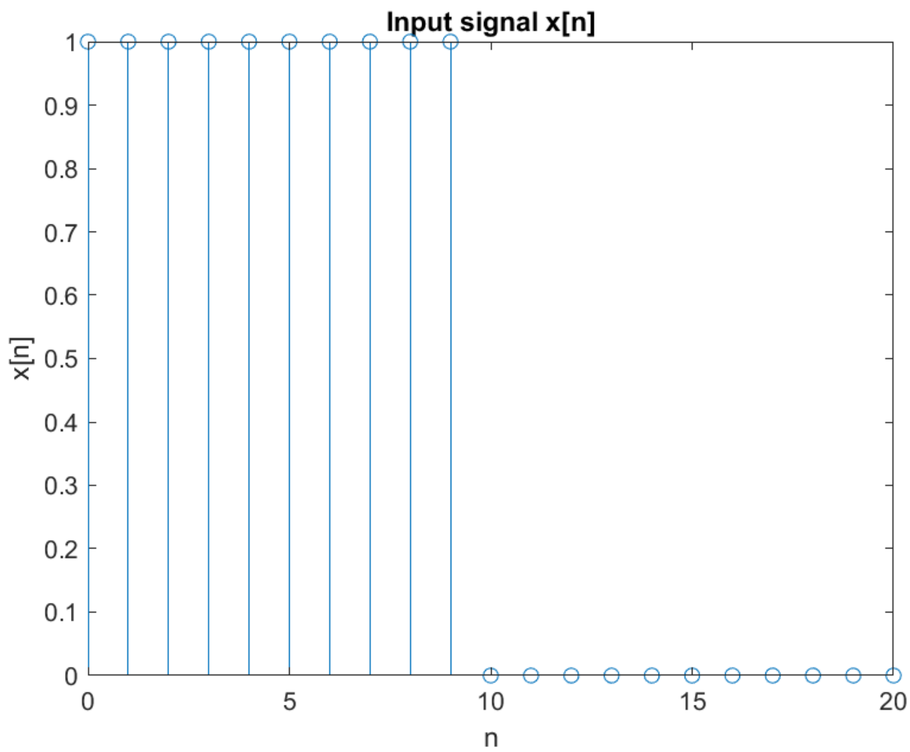
```
function out = h(n)
    out = (0.9 .^ n) .* unitstep(n);
end

function out = x(n)
    out = unitstep(n) - unitstep(n - 10);
end
```

و سپس با کد زیر آن‌ها را در بازه صفر تا ۲۰ رسم می‌کنیم.

```
n = 0:20;
figure;

stem(n, x(n));

title('Input signal x[n]');
xlabel('n');
ylabel('x[n]');

figure;

stem(n, h(n));
title('Impulse response signal h[n]');
xlabel('n');
ylabel('h[n]');
```

و نتیجه به صورت زیر در خواهد آمد.





سوال ۳ - ب)

برای به دست آوردن convolution داریم:

$$y[n] = \sum_{k=-\infty}^{+\infty} x[k]h[n-k] = \sum_{k=-\infty}^{+\infty} (u[k] - u[k-10])h[n-k] = \sum_{k=0}^{9} h[n-k]$$

$$= \sum_{k=0}^{9} (0.9)^{n-k} u(n-k) = (0.9)^n \sum_{k=0}^{9} (0.9)^{-k} u(n-k)$$

حال $y[n]$ را بازه‌بندی می‌کنیم:

$$n < 0 \qquad \Rightarrow y[n] = 0$$

$$0 \le n \le 9 \Rightarrow y[n] = (0.9)^n \sum_{k=0}^{n} (0.9)^{-k} = (0.9)^n \times \frac{(\frac{1}{0.9})^{n+1} - 1}{(\frac{1}{0.9}) - 1}$$

$$n > 9 \qquad \Rightarrow y[n] = (0.9)^n \sum_{k=0}^{9} (0.9)^{-k} \approx 16.81(0.9)^n$$

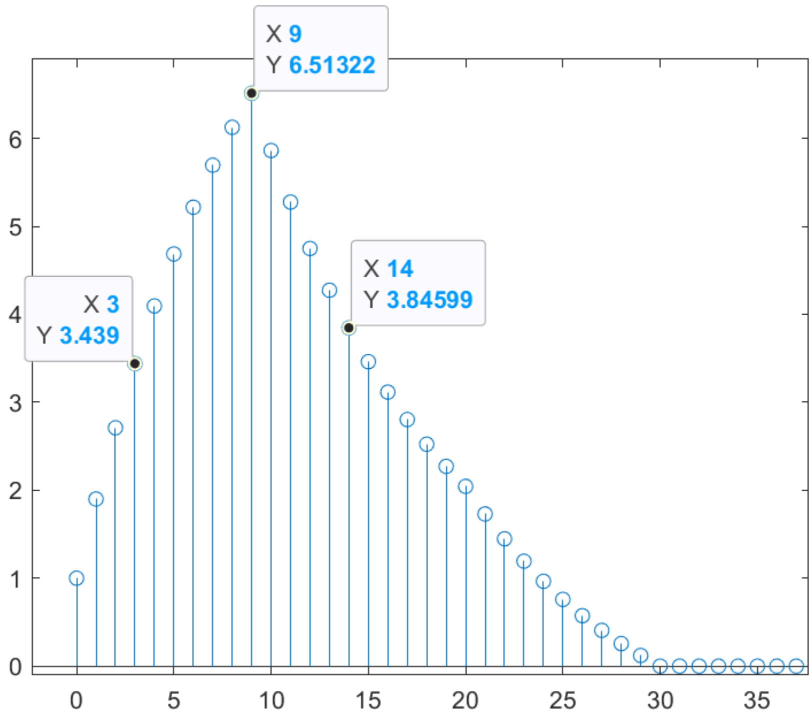
سوال ۳ - ج)

با کد زیر، حاصل convolution دو سیگنال را به دست می‌آوریم.

```
y = conv(x(n), h(n));
stem(0:size(y,2)-1, y);
```

در تابع stem، مقدار x ها را از صفر تا طول آرایه حاصل از convolution دو سیگنال قرار می‌دهیم (سمت راست بازه را منهای ۱ می‌کنیم، چون بازه‌ها در MatLab بسته هستند).

نمودار حاصل به صورت زیر خواهد بود.



با بررسی مقادیر حاصل در نمودار و حاصل convolution که در ۳-ب حساب کردیم، مشخص است که نتایج مطابقت دارند.

سوال ۴ - الف)

کد تابع my_conv به صورت زیر است:

```
function out = my_conv(x1, x2)
    out = zeros(1, length(x1) + length(x2) - 1);
    for n = 1:length(out)
        for k = 1:length(x1)
            if n - k + 1 >= 1 && n - k + 1 <= length(x2)
                out(n) = out(n) + (x1(k) * x2(n - k + 1));
            end
        end
    end
end
```

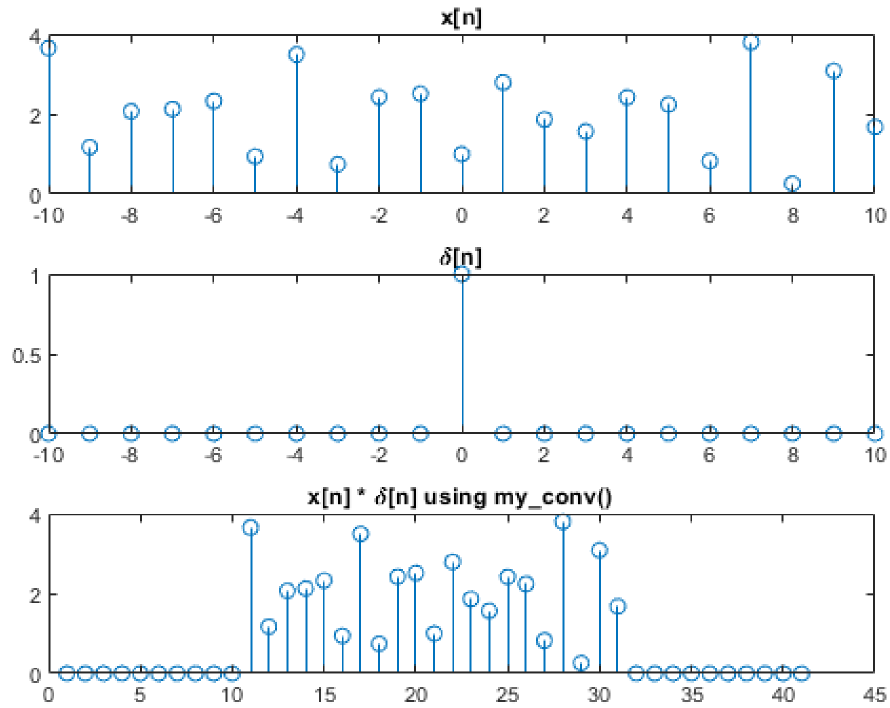
طول آرایه حاصل از convolution دو آرایه داده شده، برابر یکی کمتر از مجموع طول دو آرایه است (در واقع حاصل convolution برای بقیه‌ی مقادیر صفر است، در نتیجه آن‌ها را دیگر در خروجی نمی‌آوریم). در نتیجه خروجی را برابر آرایه‌ای با مقدار صفر و با همین طول قرار می‌دهیم. سپس به ازای هر کدام از اعضای خروجی (هر n)، حاصل سیگما که در تعریف convolution داریم را حساب می‌کنیم. همچنین اندیس‌هایی که برای محاسبه جمع استفاده می‌شود را نیز بررسی می‌کنیم تا خارج از محدوده آرایه نباشند.

سوال ۴ - ب)

حاصل convolution سیگنال $x[n]$ و $\delta[n]$ را رسم می‌کنیم.

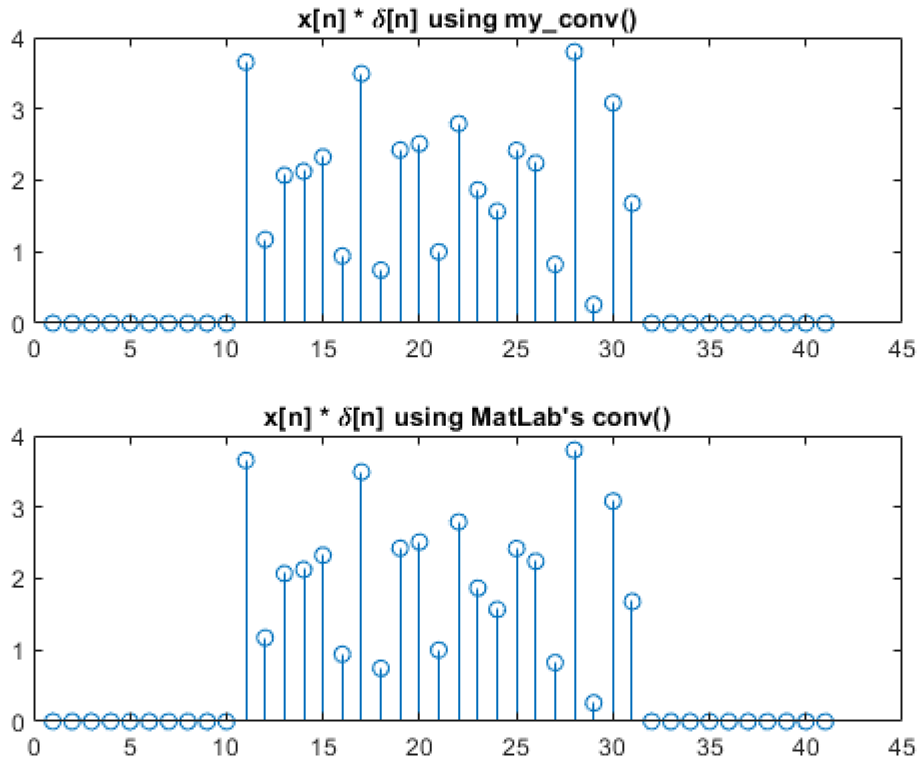
```
figure;
n = -10:10;

subplot(3, 1, 1);
stem(n, x(n));
title('x[n]');

subplot(3, 1, 2);
stem(n, unitimpulse(n));
title('\delta[n]');

subplot(3, 1, 3);
stem(my_conv(x(n), unitimpulse(n)));
title('x[n] * \delta[n] using my\_conv()');
```

تابع subplot(n, m, i)، صفحه نمودار را به یک جدول n در m تقسیم می‌کند، و خانه i ام این جدول را برای رسم نمودار انتخاب می‌کند. حاصل رسم نمودار به صورت زیر خواهد بود.



سوال ۴ - ج)

مقایسه تابع my_conv که در بالا پیاده شده و تابع conv متلب در زیر مشخص است:



مشخص است که هر دو تابع نتایج یکسانی تولید کردند. همچنین شکل سیگنال خروجی my_conv نیز دقیقا مشابه سیگنال $x[n]$ است که همان چیزی است که انتظار داشتیم؛ چون $x[n] * \delta[n] = x[n]$.

سوال ۵

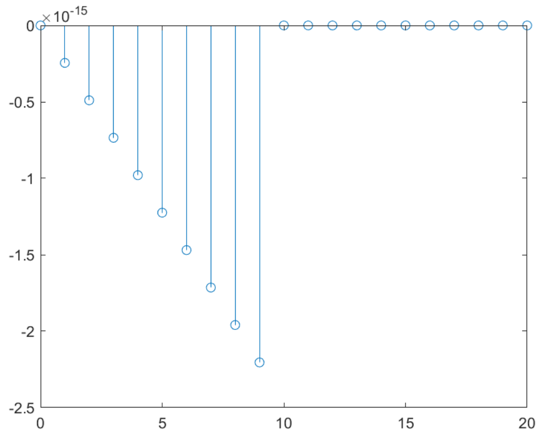
سیگنال $x[n]$ را تعریف می‌کنیم.

```
function out = x(n)
    out = sin(2*pi*n) .* (unitstep(n) - unitstep(n - 10));
end
```

حال سیگنال را از صفر تا ۲۰ رسم می‌کنیم.

```
n = 0:20;
stem(n, x(n));
```

و حاصل به شکل زیر خواهد بود.

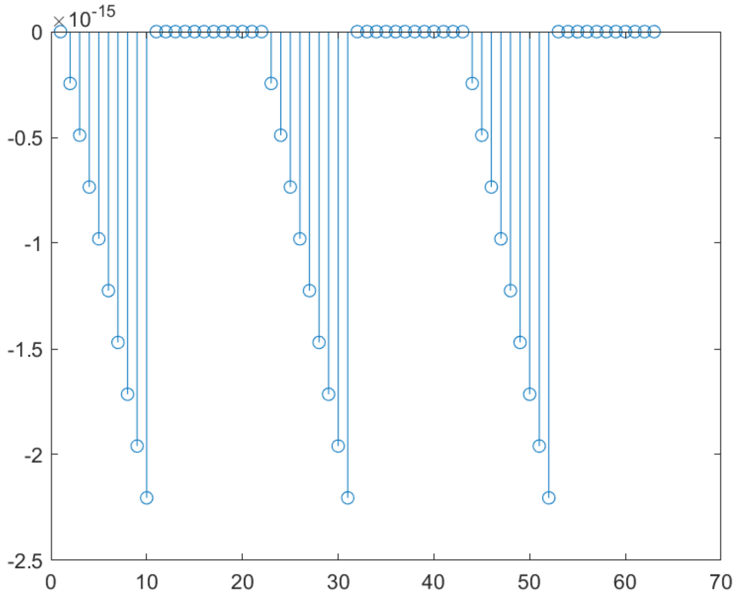


(احتمالا به دلیل خطای محاسبات ممیز شناور، $x[n]$ در بعضی نقاط دقیقا برابر صفر نشده است و مقداری نزدیک صفر را گرفته است. به صورت ایده‌آل، همه نقاط باید دقیقا برابر صفر باشند.)

حال سیگنال را با استفاده از repmat تا ۳ دفعه تکرار می‌کنیم.

```
y_rep = repmat(x(n), 1, 3);
stem(y_rep);
```

تابع repmat(A, n, m)، آرایه A را به صورت یک ماتریس n در m تکرار می‌کند و آرایه‌ای جدید برمی‌گرداند. در بالا سیگنال $x[n]$ را به صورت ستونی ۳ بار تکرار کردیم. حاصل رسم سیگنال تکرار شده به صورت زیر است.



(خطای ممیز شناور همچنان قابل مشاهده است!)

سوال ۶

ابتدا سیگنال $x(t)$ را تعریف می‌کنیم.

```
function out = x(t)
    out = exp(3j.*t) + exp(5j.*t);
end
```

تابع exp(x)، حاصل e^x را حساب می‌کند.

حال سیگنال‌های خواسته شده را رسم می‌کنیم.

```
t = linspace(-10*pi, 10*pi);

phase_x = angle(x(t));
abs_x = abs(x(t));


subplot(2, 2, 1);
plot(t, phase_x);
title('Phase of x(t)');

subplot(2, 2, 3);
plot(t, abs_x);
title('|x(t)|');

subplot(2, 2, 2);
plot(t, real(x(t)))
title('Re\{x(t)\}');

subplot(2, 2, 4);
plot(t, imag(x(t)))
title('Im\{x(t)\}');
```

تابع linspace، آرایه‌ای از اعداد با فاصله یکسان در بازه مشخص شده را می‌دهد (به صورت پیش‌فرض، این تابع ۱۰۰ عدد برمی‌گرداند). با استفاده از این تابع، نقاطی در بازه -10π تا 10π را در t ذخیره کرده‌ایم. با استفاده از تابع angle، می‌توانیم فاز یا زاویه هر نقطه از سیگنال $x(t)$ را به دست آوریم. به طور مشابه، تابع abs نیز اندازه اعداد مختلطی که در ورودی می‌گیرد را برمی‌گرداند (یا اگر اعداد داده شده حقیق باشند، قدر مطلق آن‌ها را برمی‌گرداند). در نهایت، توابع real و imag نیز قسمت‌های حقیقی و موهومی اعداد مختلط داده شده به آن‌ها را بر می‌گردانند که از آن‌ها برای رسم نمودار $x(t)$ استفاده می‌کنیم.
در نهایت، با استفاده از subplot صفحه را به ۴ قسمت تقسیم می‌کنیم و نمودارهای خواسته شده را رسم می‌کنیم.