# Table of Contents



```
close all;
clear;
clc;
```

# 2.1.a

Testing myconv function against matlab's default conv

```
x = 1:2:10;
h = 1:5;
myconv(x,h)
conv(x,h)
```

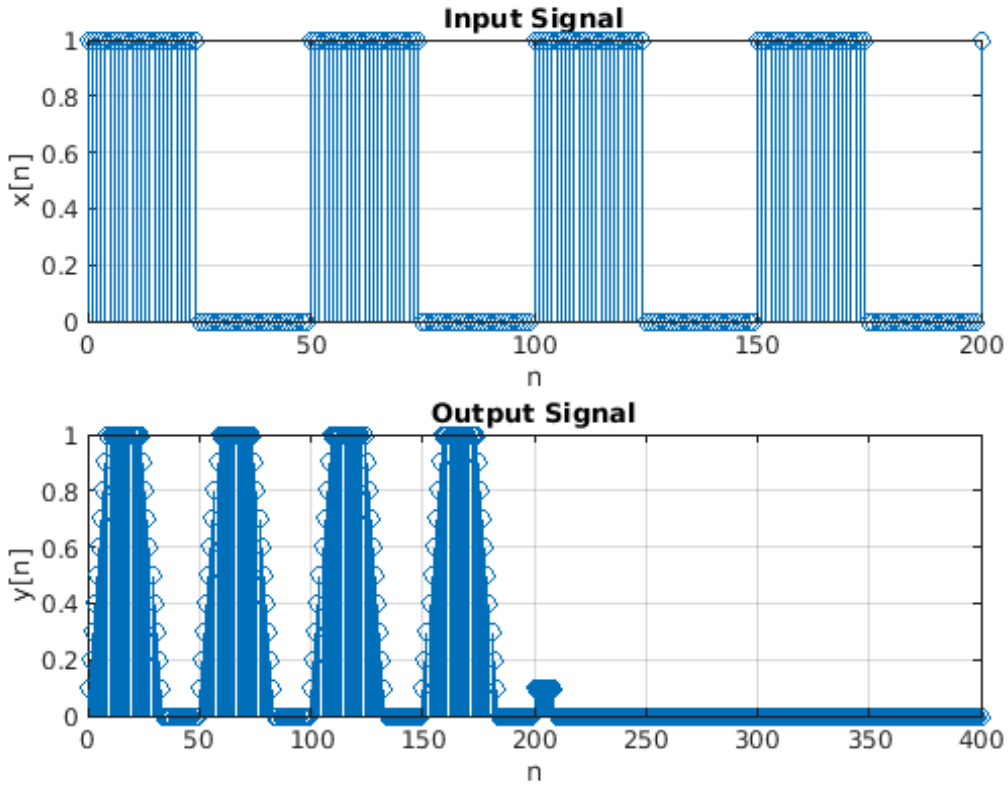
```
ans =

     1     5    14    30    55    74    80    71    45
```

```
ans =

     1     5    14    30    55    74    80    71    45
```

# 2.1.b

```
n = 0:200;
k = 50;
f = 1/k;

h = zeros(1, length(n)); % Integrator
h(1:10) = 0.1 * 1;

x = square(2 * pi * f * n) / 2 + 0.5; % Signal
y = myconv(h, x);


% Plotting the results
figure('Name', 'Integrator convolution');
```

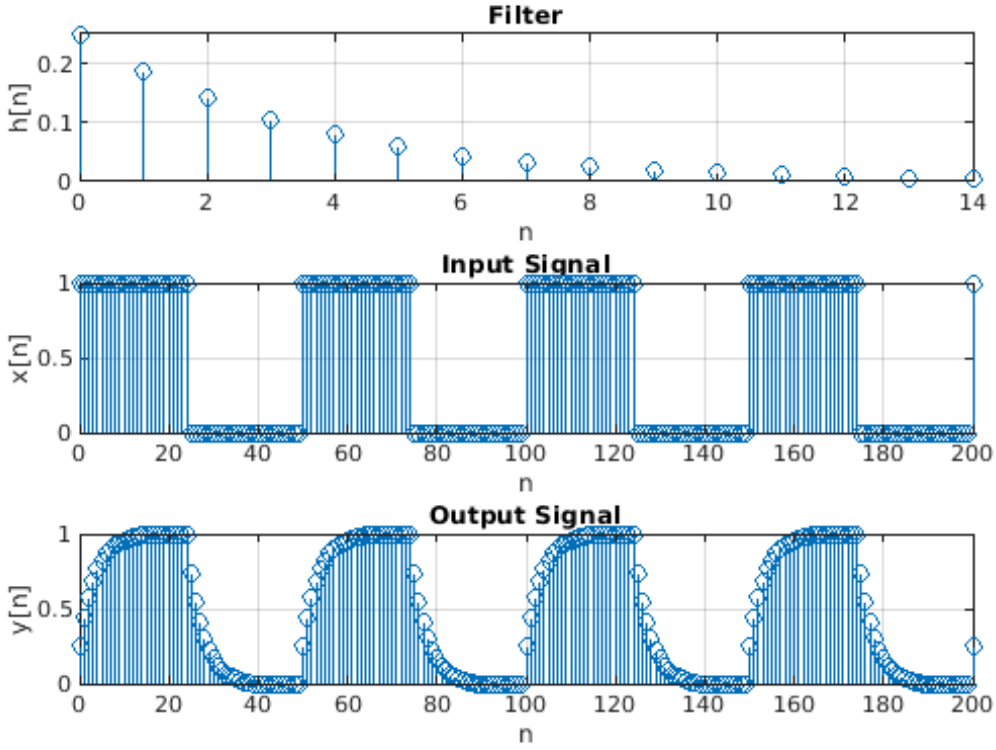
```
subplot(2,1,1);
stem(n, x);
xlabel('n') ;
ylabel('x[n]') ;
title('Input Signal') ;
grid on ;

subplot(2,1,2);
stem(0:length(y) - 1, y);
xlabel('n') ;
ylabel('y[n]') ;
title('Output Signal') ;
grid on ;
```




# 2.1.c

```
n = 0:14;
h2 = 0.25 * 0.75.^n; % Filter definition
y = myconv(h2, x);

% Plotting the results
figure('Name', 'Filter Convolution');
subplot(3,1,1);
stem(n, h2);
xlabel('n') ;
```

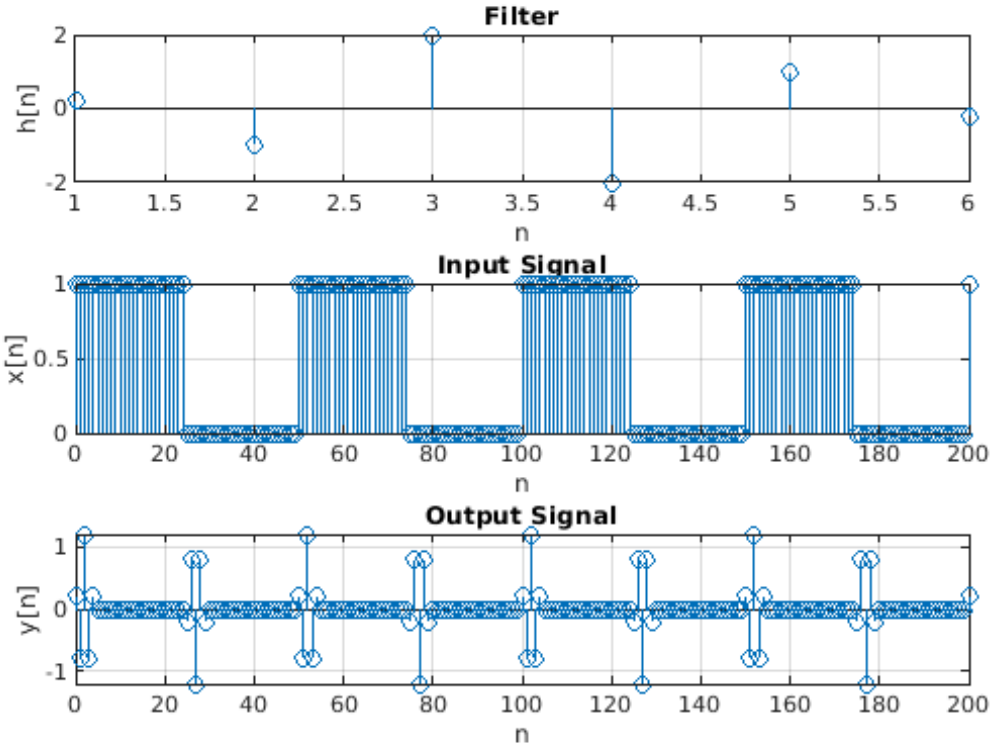
```
ylabel('h[n]') ;
title('Filter') ;
grid on ;

subplot(3,1,2);
stem(0:length(x) - 1, x);
xlabel('n') ;
ylabel('x[n]') ;
title('Input Signal') ;
xlim([0,200]);
grid on ;

subplot(3,1,3);
stem(0:length(y) - 1, y);
xlabel('n') ;
ylabel('y[n]') ;
title('Output Signal') ;
xlim([0,200]);
grid on ;
```




# 2.1.d

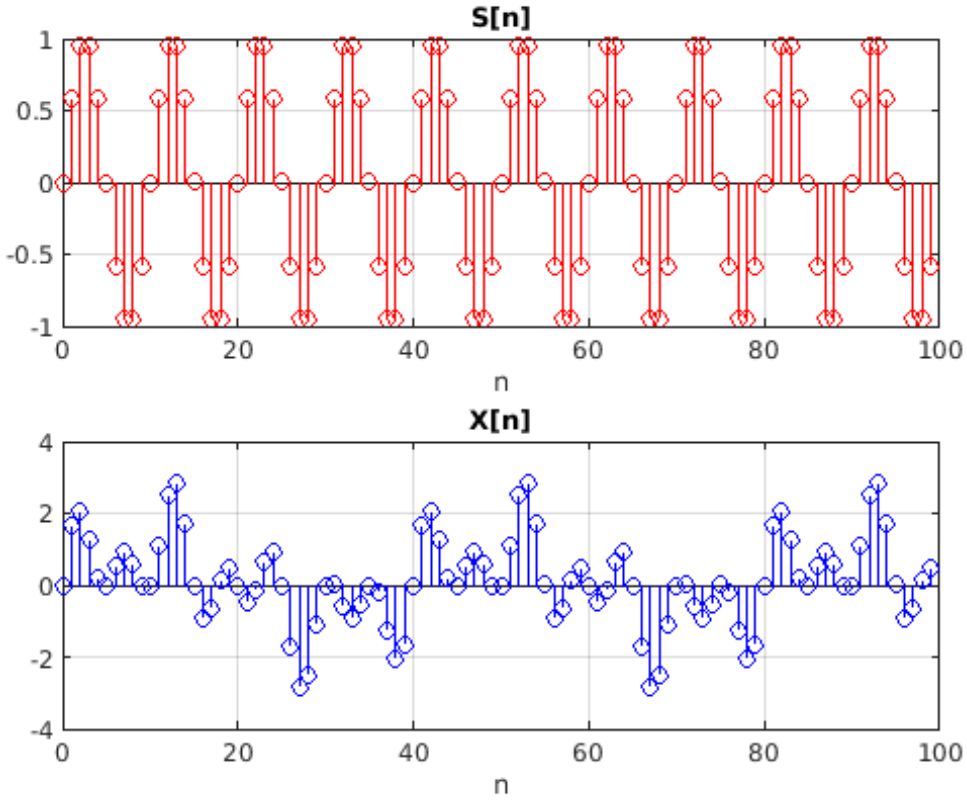
```
% (0.2 (z^5 #(n) - 5 z^4 #(n) + 10 z^3 #(n) - 10 z^2 #(n) + 5 z #(n) - #(n)))/
z^5
h3 = 0.2*[1,-5,10,-10,5,-1]; % Filter definition
y = myconv(h3, x);
```

```
% Plotting the results
figure('Name', 'Filter Convolution');
subplot(3,1,1);
stem(h3);
xlabel('n') ;
ylabel('h[n]') ;
title('Filter') ;
grid on ;

subplot(3,1,2);
stem(0:length(x) - 1, x);
xlabel('n') ;
ylabel('x[n]') ;
title('Input Signal') ;
xlim([0,200]);
grid on ;

subplot(3,1,3);
stem(0:length(y) - 1, y);
xlabel('n') ;
ylabel('y[n]') ;
title('Output Signal') ;
xlim([0,200]);
grid on ;
```



# 2.2.a

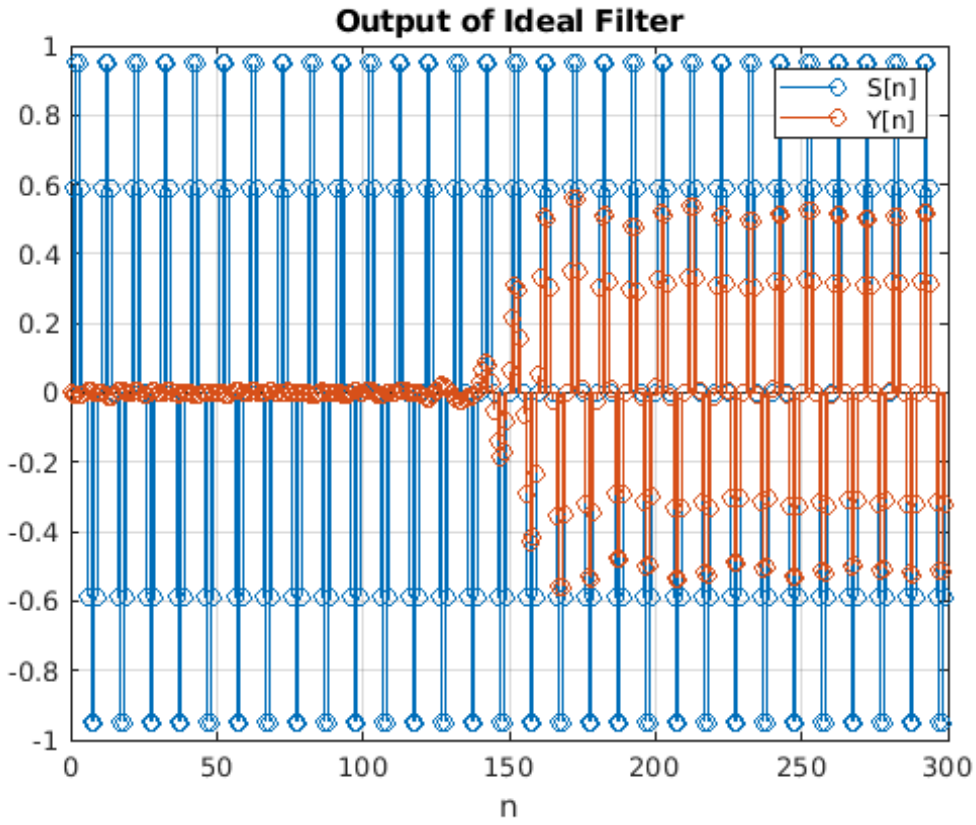
```
% Defining input variables
M  = 100;
n  = 0 : M - 1;
w1 = 0.05 * pi;
w2 = 0.20 * pi;
w3 = 0.35 * pi;
wa = 0.15 * pi;
wb = 0.25 * pi;

s  = sin(w2 * n); % Signal
v  = sin(w1 * n) + sin (w3 * n); % Noise
x  = s + v; % Noisy Signal

w  = 0.54 - 0.46 * sin(2 * pi * n / M);
h  = w .* ( wb * sinc(wb * (n - M/2)/pi)/pi - wa * sinc(wa * (n - M/2)/pi)/
pi );

% Plotting the results
figure('Name', 'desired and final signal with noise');
subplot(2,1,2);
stem(n, x,'b');
title('X[n]'); xlabel('n'); ylabel(''); grid on;

subplot(2,1,1);
stem(n, s,'r');
title('S[n]'); xlabel('n'); ylabel(''); grid on;
```




# 2.2.b

```
% Defining input variables
M  = 300;
n  = 0 : M - 1;
w1 = 0.05 * pi;
w2 = 0.20 * pi;
w3 = 0.35 * pi;
wa = 0.15 * pi;
wb = 0.25 * pi;

s  = sin(w2 * n);
v  = sin(w1 * n) + sin (w3 * n);
x  = s + v;

w  = 0.54 - 0.46 * sin(2 * pi * n / M);
h  = w .* (wb * sinc(wb * (n - M/2)/pi)/pi - wa * sinc(wa * (n - M/2)/pi)/pi);

y = filter(h,1,x);

% Plotting the results
figure('Name', 'Original vs Filtered Signal');

stem(n, s);
title("Output of Ideal Filter");
```

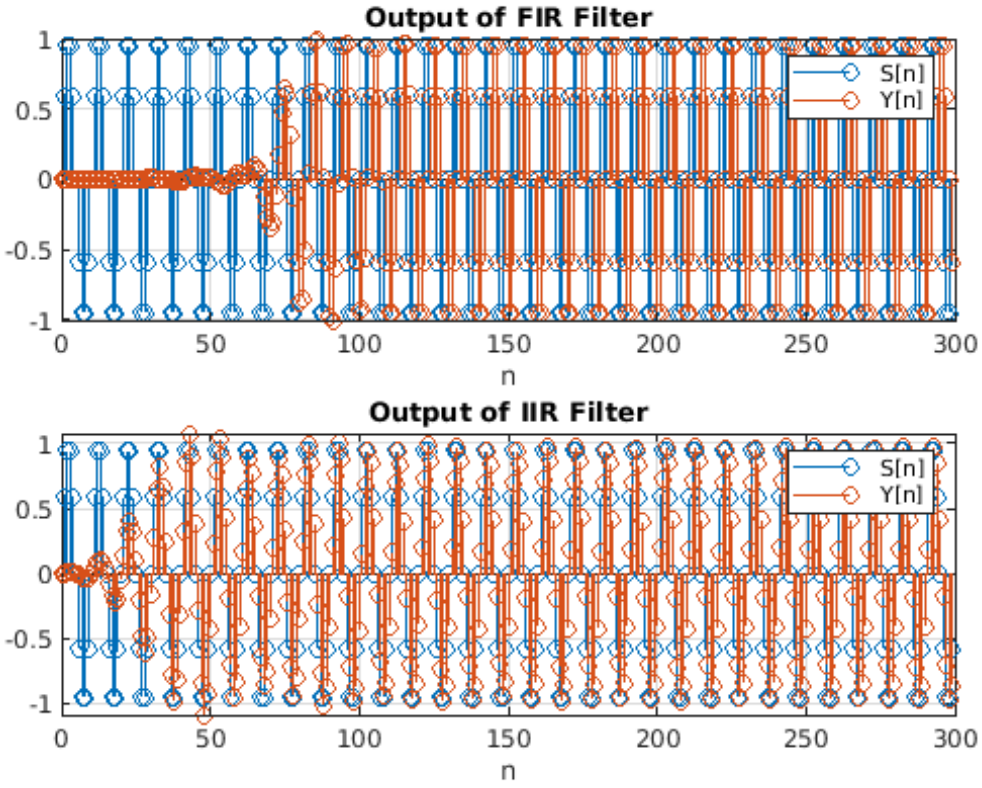
```
xlabel('n'); ylabel(''); grid on;
hold on;

stem(n, y);
xlabel('n'); ylabel(''); grid on;

legend('S[n]', 'Y[n]')
```




# 2.2.c

```
% Loading and applying FIR filter
load('Filter_FIR.mat');
y1 = filter(Num,1,x);


% Loading and applying IIR filter
load('Filter_IIR.mat');
y2 = prod(G)*sosfilt(SOS,x);

% Plotting the results
figure('Name', 'Original vs IIR vs FIR');

subplot(2,1,1);
stem(n, s);
title("Output of FIR Filter");
```

```
xlabel('n'); ylabel(''); grid on;
hold on;

stem(n, y1);
xlabel('n'); ylabel(''); grid on;
legend('S[n]', 'Y[n]');


subplot(2,1,2);
stem(n, s);
title("Output of IIR Filter");
xlabel('n'); ylabel(''); grid on;
hold on;

stem(n, y2);
xlabel('n'); ylabel(''); grid on;
legend('S[n]', 'Y[n]');
```




*Published with MATLAB® R2023a*