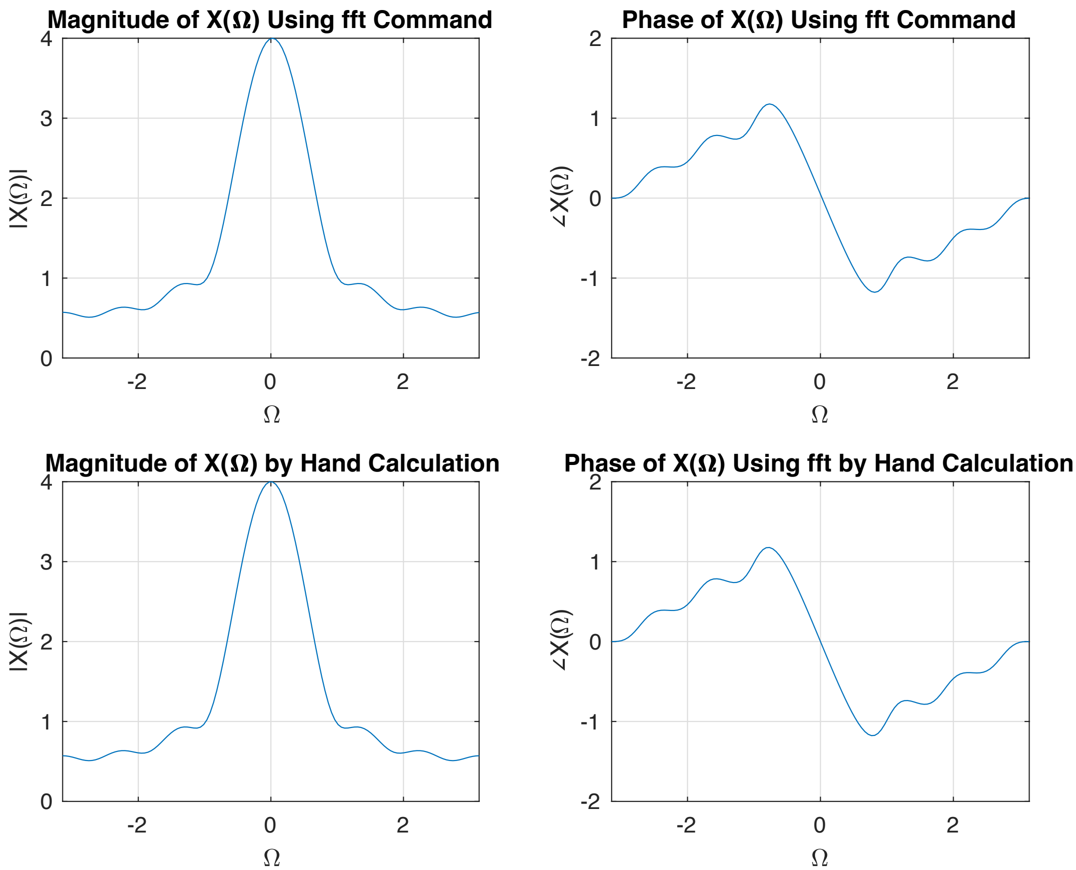
Discrete-Time Fourier Transform

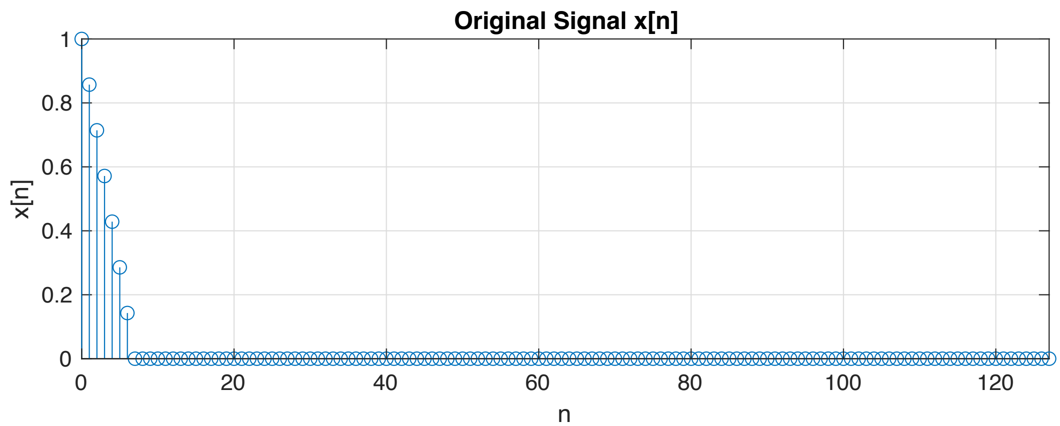
Table of Contents

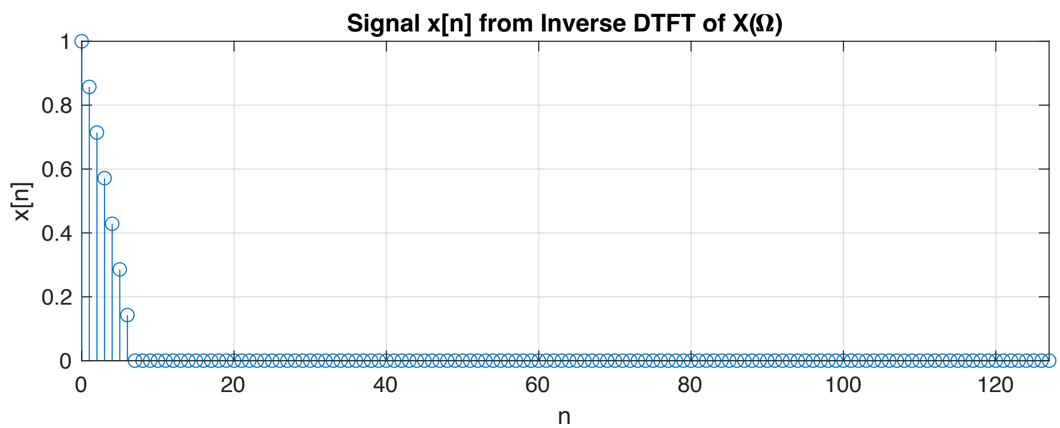
A. Discrete-time Fourier transform (DTFT)

```
% A.1

n = 0:127;
u = @(n) (n >= 0) * 1.0 .* (mod(n,1)==0);
% Signal x[n]
x = @(n) (1-n/7).*(u(n)-u(n-7));
% Compute DTFT of x[n] using fft command
X = fft(x(n));
% Center the zero frequency component
X = fftshift(X);

Omega = linspace(-pi,pi,128);

% A.2
% Compute DTFT of x[n] by hand calculation
X1 = @(Omega) (1 ...
    +   (6/7).*exp(-1*j*Omega)...
    +   (5/7).*exp(-2*j*Omega)...
    +   (4/7).*exp(-3*j*Omega)...
    +   (3/7).*exp(-4*j*Omega)...
    +   (2/7).*exp(-5*j*Omega)...
    +   (1/7).*exp(-6*j*Omega));

figure;
subplot(2,2,1);
plot(Omega,abs(X));
grid;
title("Magnitude of X(\Omega) Using fft Command");
xlabel('\Omega');
ylabel('|X(\Omega)|');
axis([-pi pi 0 4]);

subplot(2,2,2);
plot(Omega,angle(X));
grid;
title("Phase of X(\Omega) Using fft Command");
xlabel('\Omega');
ylabel('\angleX(\Omega)');
axis([-pi pi -2 2]);

subplot(2,2,3);
plot(Omega,abs(X1(Omega)));
grid;
title("Magnitude of X(\Omega) by Hand Calculation");
xlabel('\Omega');
ylabel('|X(\Omega)|');
axis([-pi pi 0 4]);

subplot(2,2,4);
plot(Omega,angle(X1(Omega)));
grid;
title("Phase of X(\Omega) Using fft by Hand Calculation");
xlabel('\Omega');
ylabel('\angleX(\Omega)');
axis([-pi pi -2 2]);
```




```
% The magnitude and phase of DTFT obtained by hand calculation give
% the same result as DTFT computed from fft command from Matlab.

% A.3

% Inverse zero-frequency shift
X = ifftshift(X);
% Compute x[n] from inverse DTFT using ifft
x_n = ifft(X);

figure;
subplot(2, 1, 1);
stem(n, x(n));
grid;
title('Original Signal x[n]');
xlabel('n');
ylabel('x[n]');
axis([0 127 0 1]);

subplot(2, 1, 2);
stem(n,x_n);
grid;
title('Signal x[n] from Inverse DTFT of X(\Omega)');
xlabel('n');
ylabel('x[n]');
axis([0 127 0 1]);
```



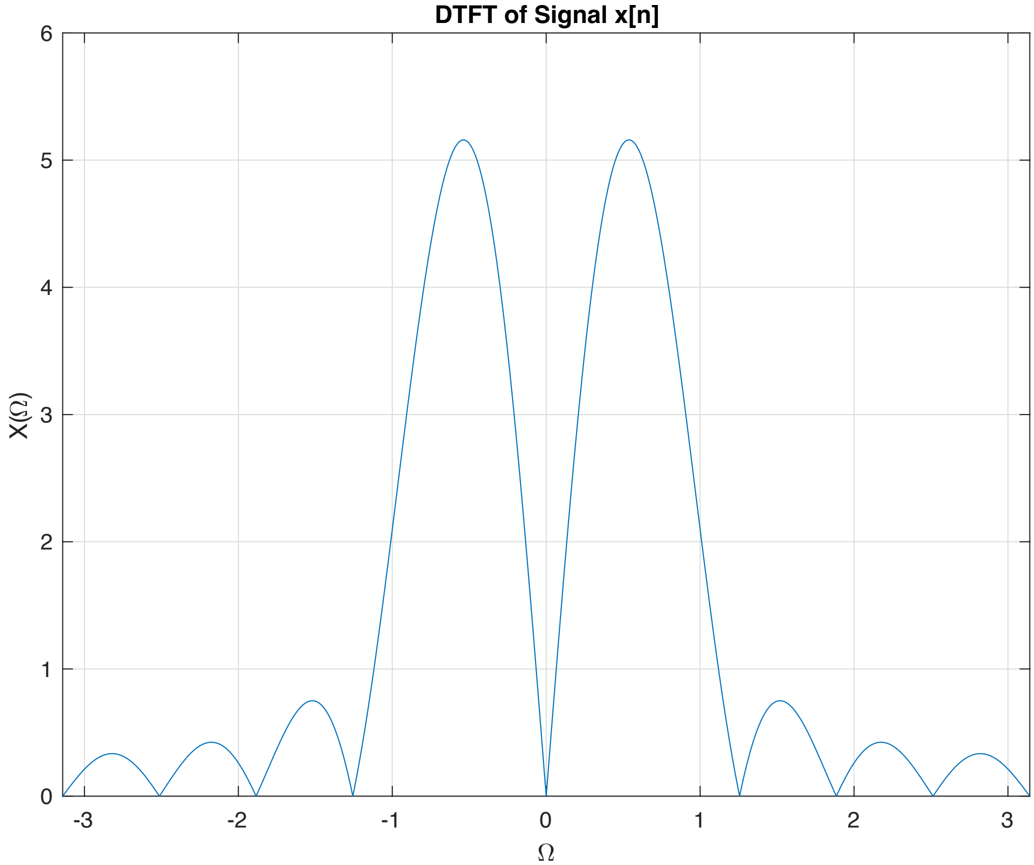



```
% The obtained result is the same as signal x[n] since ifft command
% is used to computed the inverse DTFT of X(Ω).
```

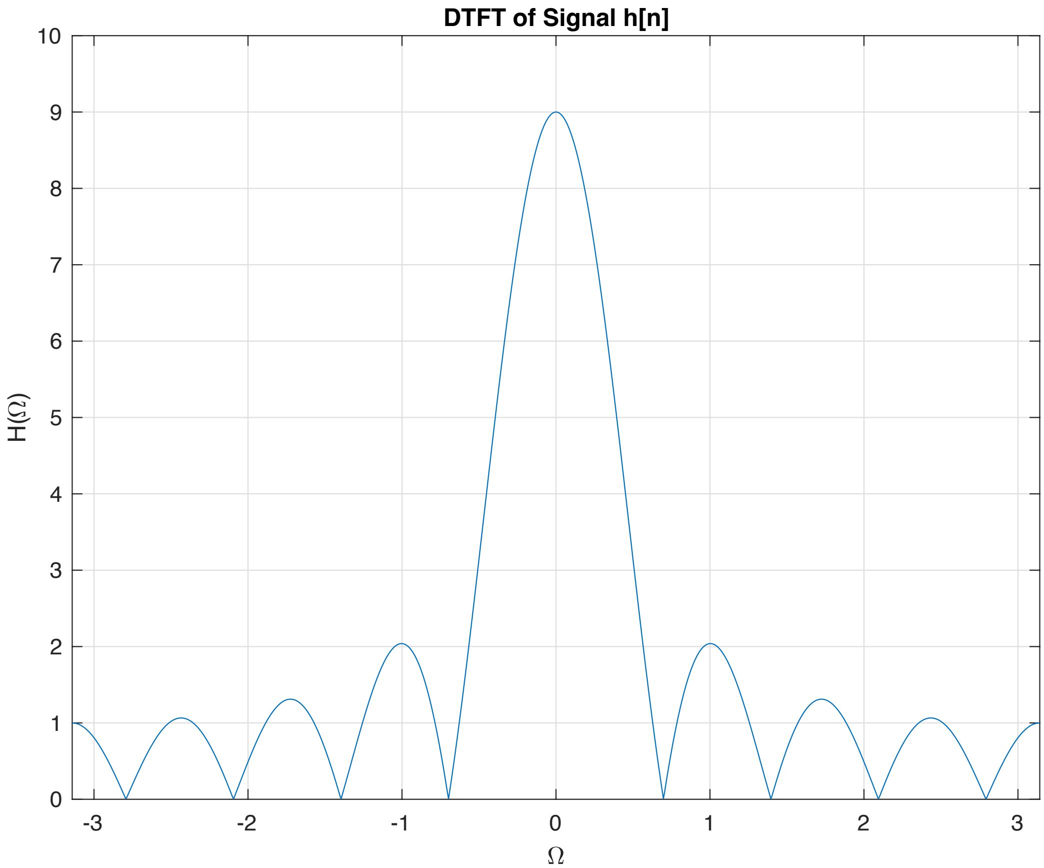
B. Time convolution

```
% B. Time Convolution

% B.1

n = 0:1000;

u = @(n) (n >= 0) * 1.0 .* (mod(n,1)==0);
% Signal x[n]
x = @(n) sin(2*pi*n/10).*(u(n)-u(n-10));

% Compute DTFT of signal x[n]
omega = linspace(-pi, pi, 1001);
W_omega_x = exp(-j).^((0:length(x(n))-1)'*omega);
X = (x(n)*W_omega_x);

figure;
plot(omega,abs(X));
grid;
title('DTFT of Signal x[n]');
xlabel('\Omega');
```

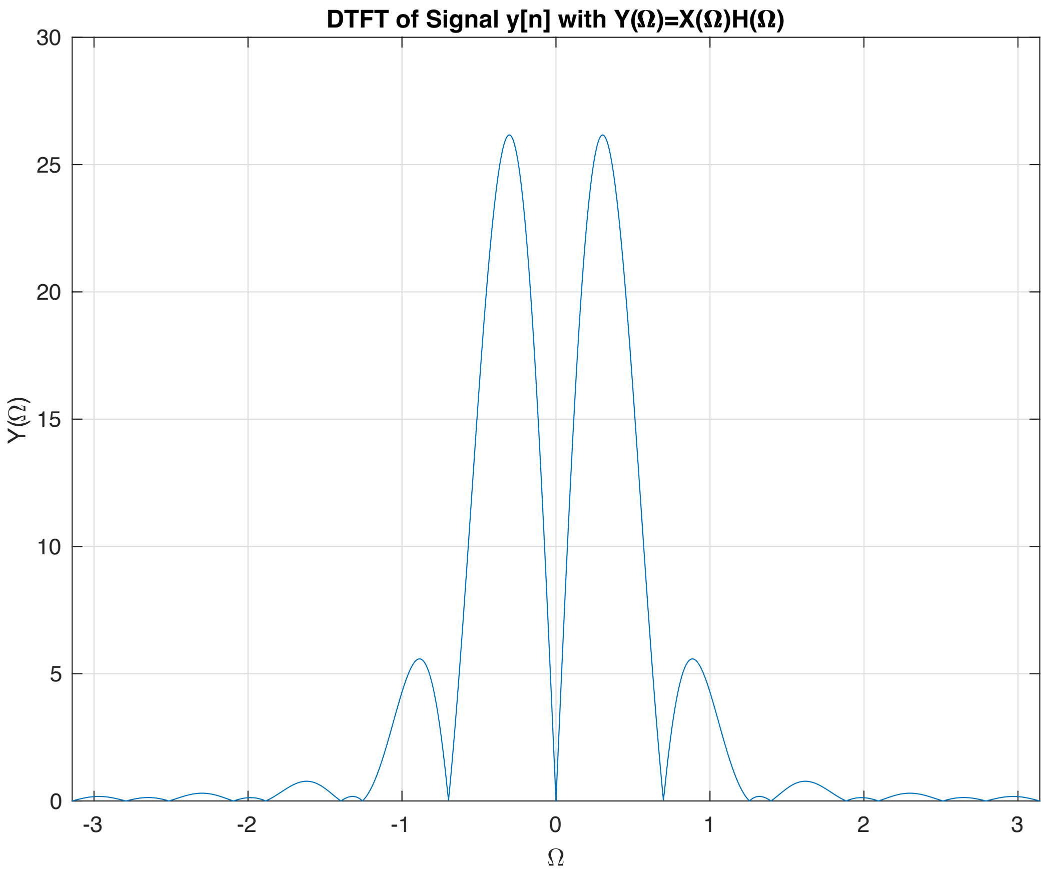
```
ylabel('X(\Omega)');
axis([-pi pi 0 6]);
```



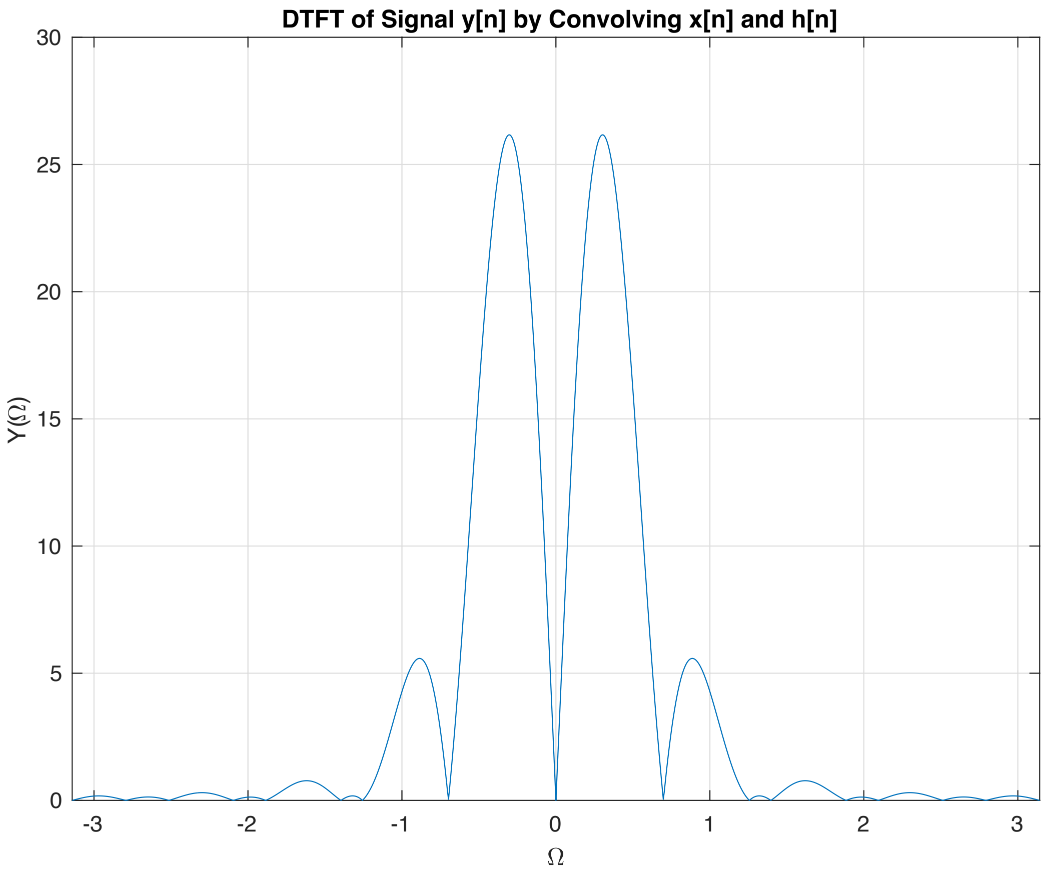

```
% B.2

% Signal h[n]
h = @(n) u(n)-u(n-9);
% Compute DTFT of signal h[n]
W_omega_h = exp(-j).^((0:length(h(n))-1)'*omega);
H = (h(n)*W_omega_h);

figure;
plot(omega,abs(H));
grid;
title('DTFT of Signal h[n]');
xlabel('\Omega');
ylabel('H(\Omega)');
axis([-pi pi 0 10]);
```




```
% B.3
Y1 = X.*H;

figure;
plot(omega,abs(Y1));
grid;
title('DTFT of Signal y[n] with Y(\Omega)=X(\Omega)H(\Omega)');
xlabel('\Omega');
ylabel('Y(\Omega)');
axis([-pi pi 0 30]);
```




```
% B.4

% Compute y[n] by convolving x[n] and h[n]
y = @(n) conv(x(n),h(n));

% B.5
% Compute DTFT of signal y[n]
W_omega_y = exp(-j).^((0:length(y(n))-1)'*omega);
Y2 = (y(n)*W_omega_y);

figure;
plot(omega,abs(Y2));
grid;
title('DTFT of Signal y[n] by Convolving x[n] and h[n]');
xlabel('\Omega');
ylabel('Y(\Omega)');
axis([-pi pi 0 30]);
```

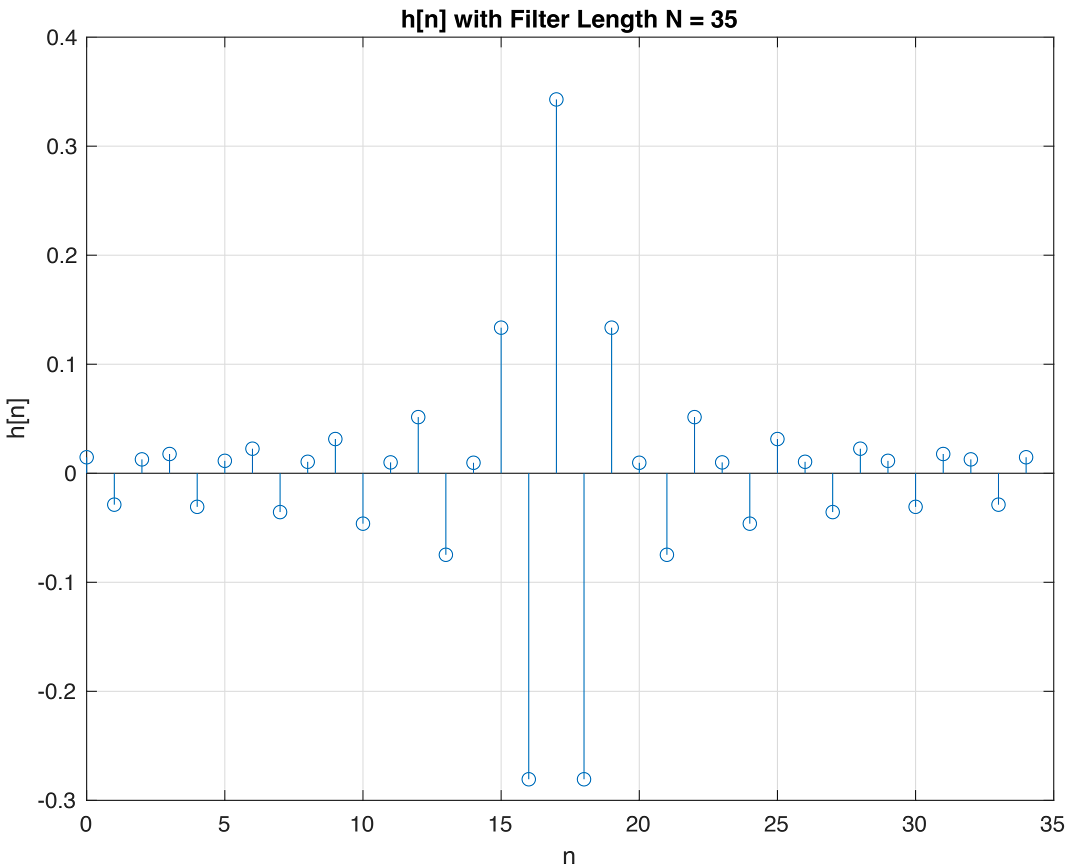



```
% B.6

% The results in part 3 and 5 are the same as it shows the time
% convolution property of the DTFT. Specifically, the convolution
% of signals in the time domain will be transformed into the
% multiplication of their Fourier transforms in the frequency domain.
```

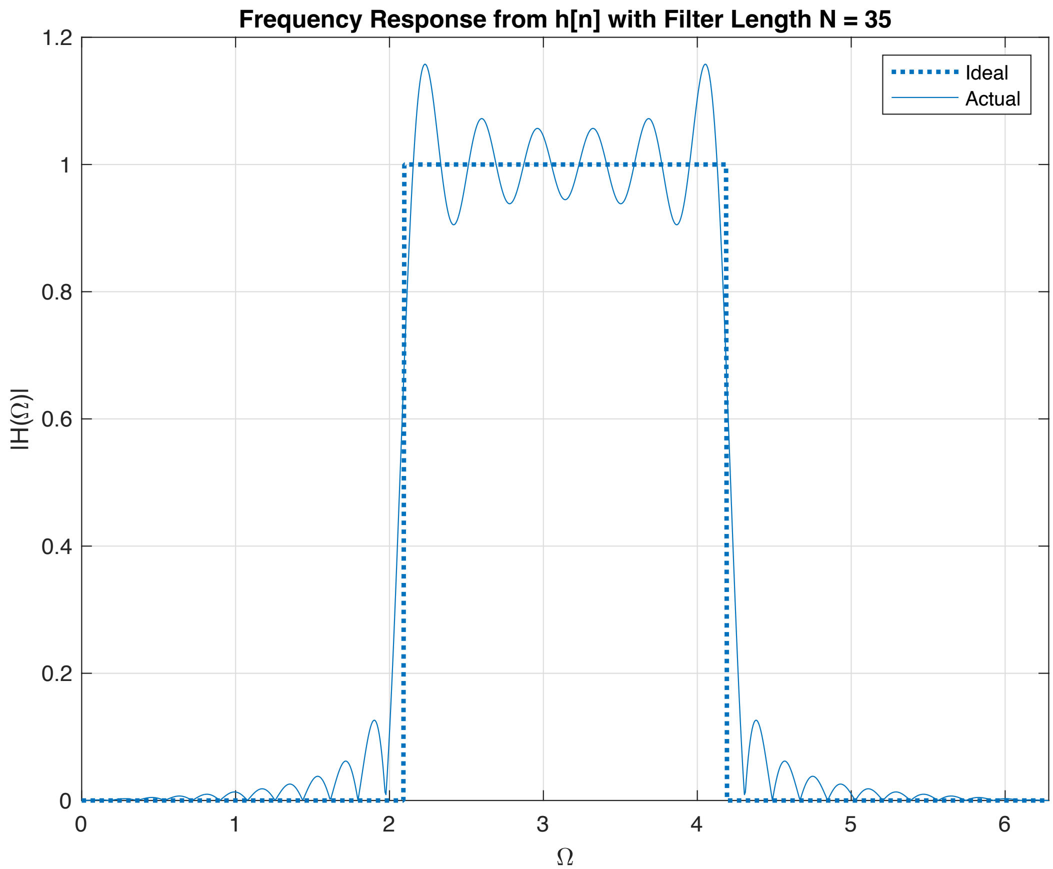
C. FIR filter design by frequency Sampling

```
% C.1

N = 35; % filter length

% Create N equally spaced frequency samples
Omega = linspace(0,2*pi*(1-1/N),N)';

% Design of a high pass FIR filter
H_d_35 = @(Omega) (mod(Omega,2*pi)>2*pi/3) & (mod(Omega,2*pi)<2*pi-2*pi/3);

% Sample the desired magnitude response and create h[n]
H_35 = H_d_35(Omega);

% Define phase to shift h[n] by (N - 1)/2
H_35 = H_35.*exp(-j*Omega*((N-1)/2));
H_35(fix(N/2)+2:N,1) = H_35(fix(N/2)+2:N,1)*((-1)^(N-1));
```

```
% Create h[n]
h_35 = real(ifft(H_35));

figure;
stem(0:N-1,h_35);
grid;
title('h[n] with Filter Length N = 35');
xlabel('n');
ylabel('h[n]');
```

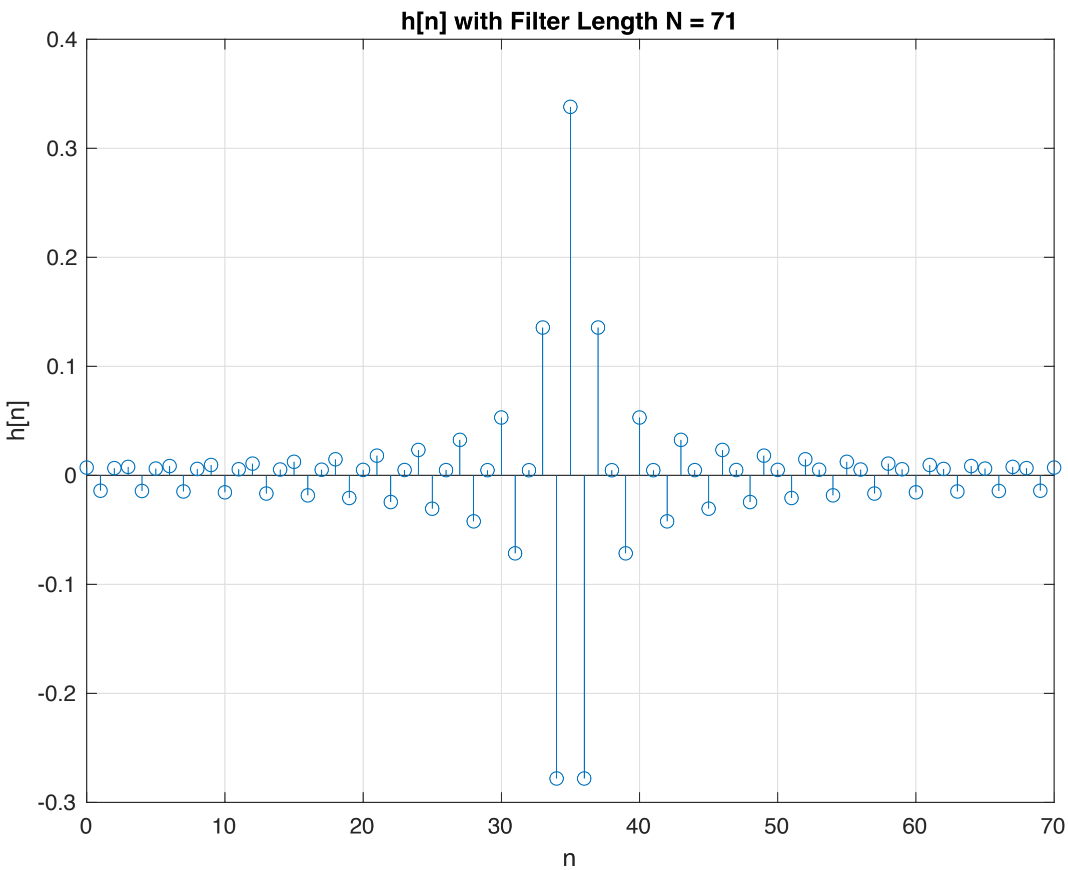


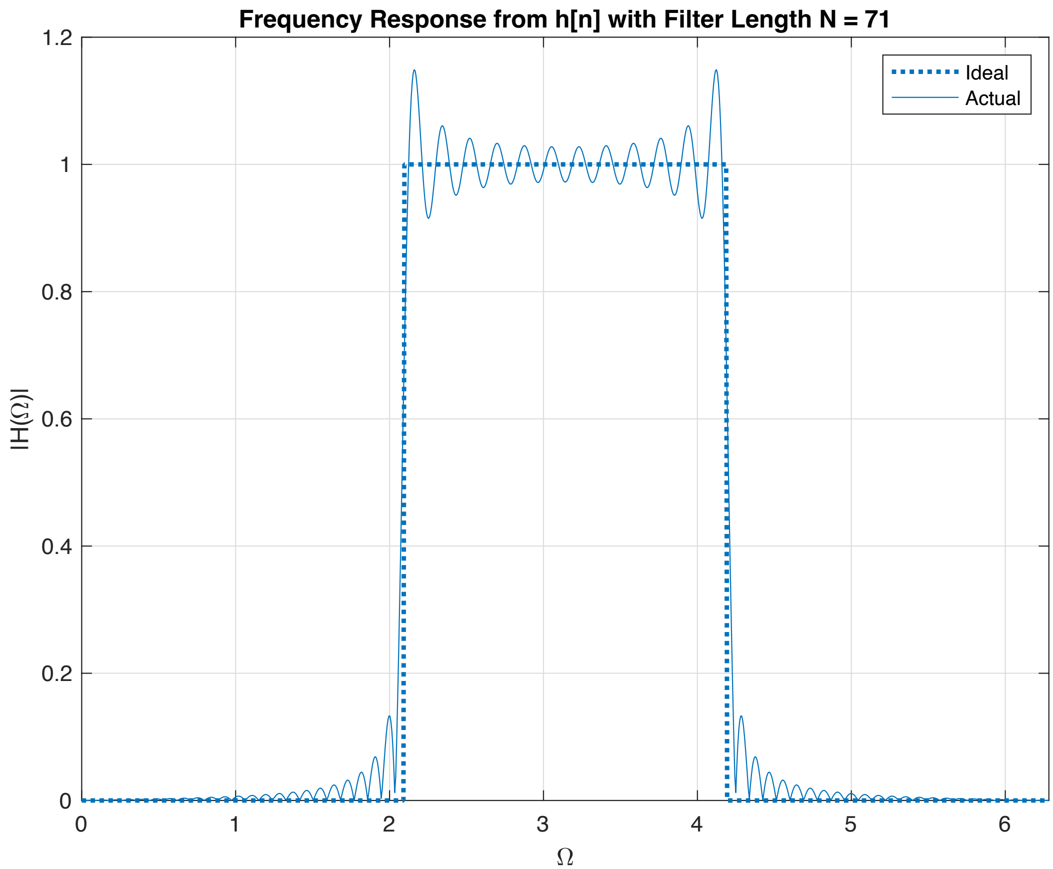

```
% C.2
Omega = linspace (0, 2*pi, 1002);
H_35 = freqz(h_35,1,0:2*pi/1001:2*pi);
figure;
plot(Omega,H_d_35(Omega),':','LineWidth',2);
hold on;
plot(Omega,abs(H_35),'color',[0.00 0.45 0.74]);
hold off;
grid;
title('Frequency Response from h[n] with Filter Length N = 35');
xlabel('\Omega');
ylabel('|H(\Omega)|');
axis([0 2*pi 0 1.2]);
legend('Ideal','Actual');
```




```
% C.3

% The frequency response with  filter length N = 35 has general
% shape which is close to the ideal filter with occurrence of ripple.

% C.4
N = 71; % filter length

% Create N equally spaced frequency samples
Omega = linspace(0,2*pi*(1-1/N),N)';

% Design of a high pass FIR filter
H_d_71 = @(Omega) (mod(Omega,2*pi)>2*pi/3) & (mod(Omega,2*pi)<2*pi-2*pi/3);

% Sample the desired magnitude response
H_71 = 1.0*H_d_35(Omega);

% Define phase to shift h[n] by (N - 1)/2
H_71 = H_71.*exp(-j*Omega*((N-1)/2));
H_71(fix(N/2)+2:N,1) = H_71(fix(N/2)+2:N,1)*((-1)^(N-1));

% Create h[n]
h_71 = real(ifft(H_71));

figure;
stem(0:N-1,h_71);
grid;
title('h[n] with Filter Length N = 71');
```

```
xlabel('n');
ylabel('h[n]');

Omega = linspace (0, 2*pi, 1002);
H_71 = freqz(h_71,1,0:2*pi/1001:2*pi);
figure;
plot(Omega,H_d_71(Omega),':','LineWidth',2);
hold on;
plot(Omega,abs(H_71),'color',[0.00 0.45 0.74]);
hold off;
grid;
title('Frequency Response from h[n] with Filter Length N = 71');
xlabel('\Omega');
ylabel('|H(\Omega)|');
axis([0 2*pi 0 1.2]);
legend('Ideal','Actual');
```






```
% C.5

% When the length of filter increased to 71, the ripple in the
% frequency response is significantly reduced as it has more samples.
% Therefore, the quality of the filter is improved and the response is
% closer to the ideal filter.
```