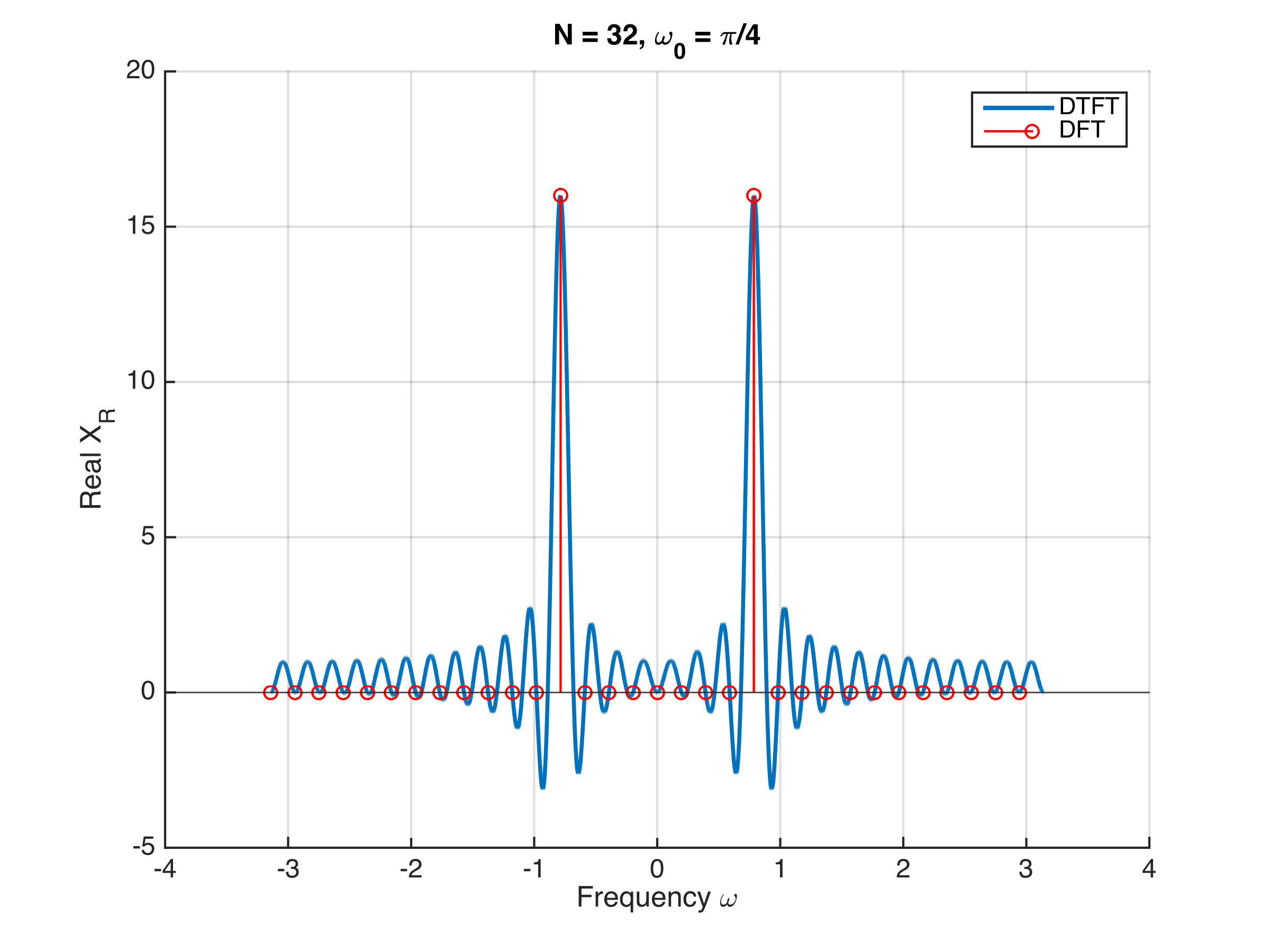
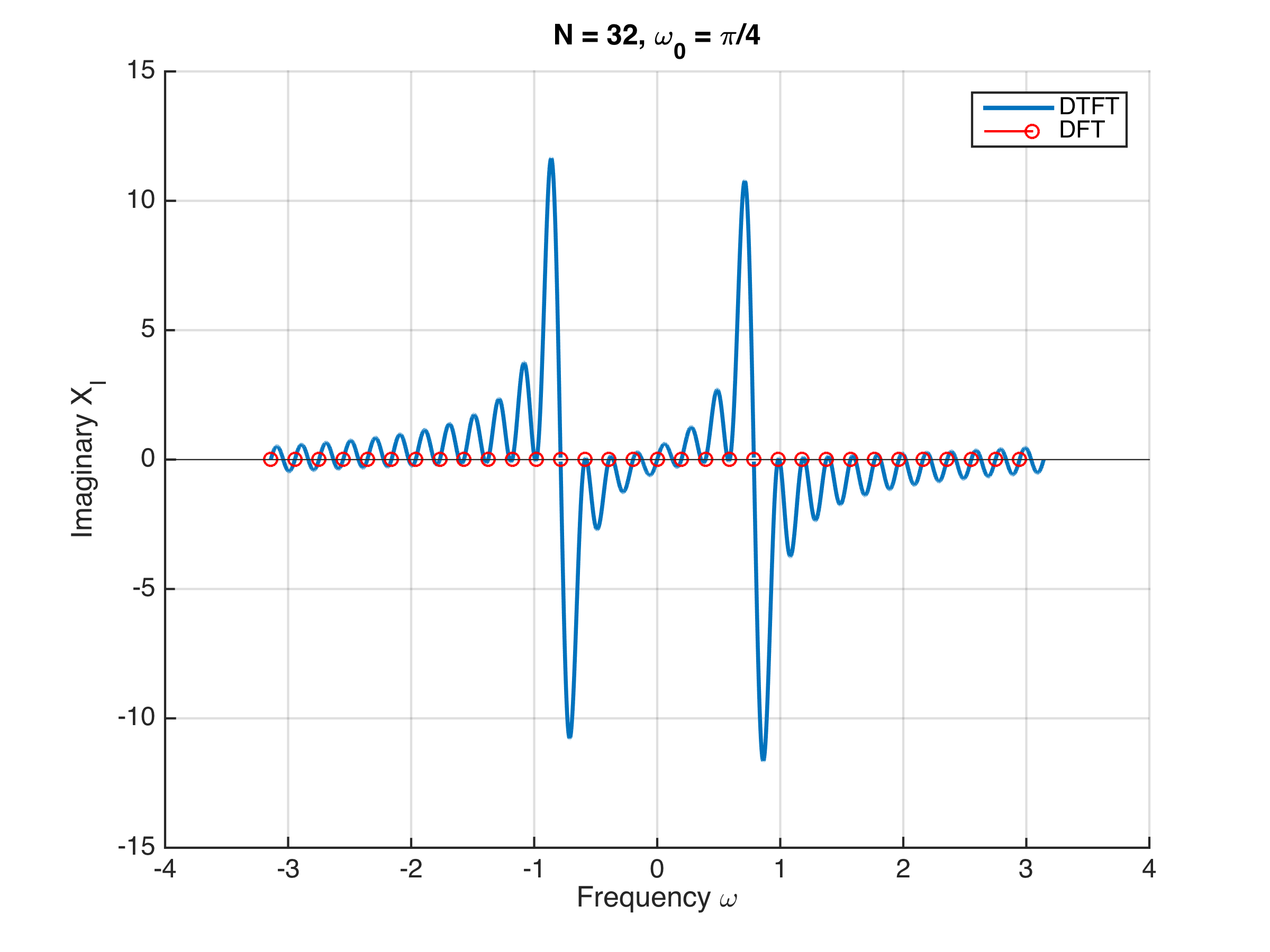
**2.2 b and c)**





The 32-point DFT is the periodic representation of the signal x(n) and thus represents the periodic signal cos(ω0n) as opposed to the spectrum of the windowed signal given by the DTFT. As expected, because the 32-point DFT encompasses complete cycles of the sinusoid when ω0 = π/4, the resulting DFT spectrum consists of two impulse functions representing a sinusoid of frequency ω0 i.e. π/4.

%% 2.2c

clear all; close all; clc

w0 = pi/4;

N = 32;

A = 1;

w = (-pi:pi/10000:pi);

Xr = (A/2).\*(cos((w-w0).\*(N-1)/2).\*(sin((w-w0).\*N/2)./sin((w-w0)./2))) ...

+(A/2).\*(cos((w+w0).\*(N-1)/2).\*(sin((w+w0).\*N/2)./sin((w+w0)./2)));

Xi = -(A/2).\*(sin((w-w0).\*(N-1)/2).\*(sin((w-w0).\*N/2)./sin((w-w0)./2))) ...

-(A/2).\*(sin((w+w0).\*(N-1)/2).\*(sin((w+w0).\*N/2)./sin((w+w0)./2)));

n = 1:32;

N = length(n);

x\_dft = A.\*cos(w0\*(n-1));

figure

plot(x\_dft)

X\_dft = fftshift(fft(x\_dft));

w\_dft = (0:2/N:2\*(N-1)/N).\*pi-pi;

figure;

hold on

plot(w,Xr)

stem(w\_dft,real(X\_dft),'r')

title('N = 32, \omega\_0 = \pi/4'), ylabel('Real X\_R'),xlabel('Frequency \omega')

hold off

% print -dpng -r300 ./hw1\_2-2c\_real.png

figure

hold on

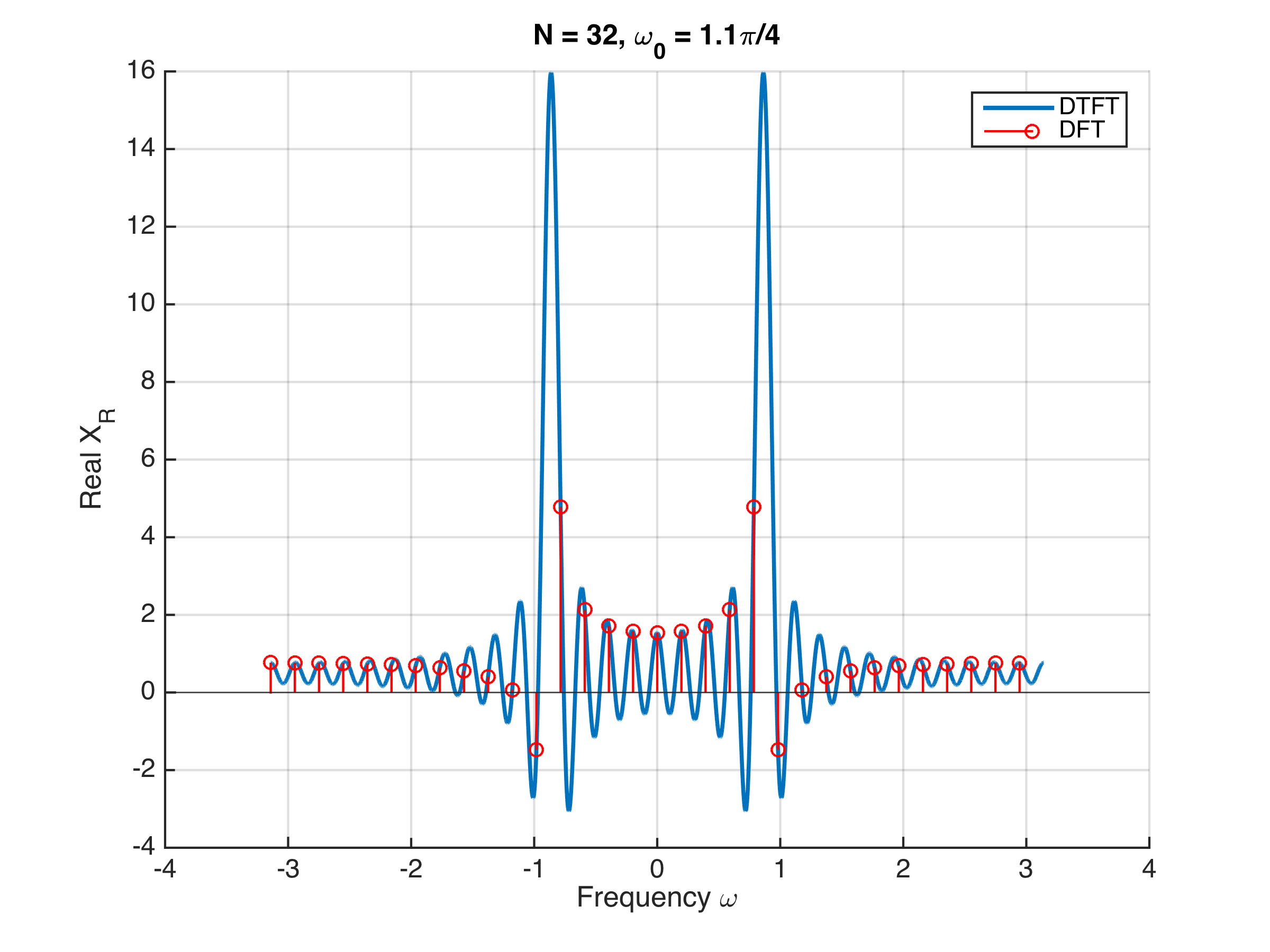
plot(w,Xi)

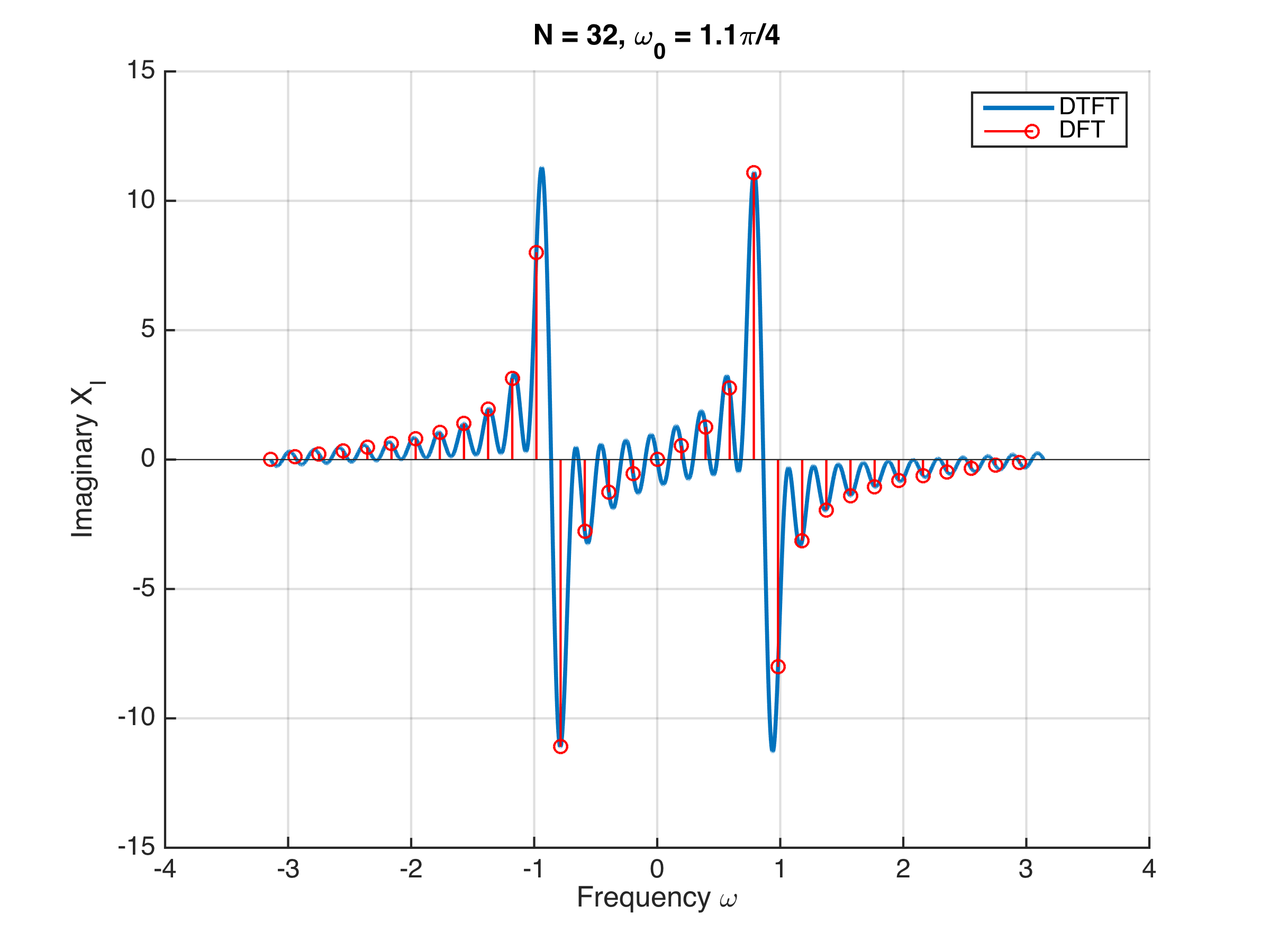
stem(w\_dft,imag(X\_dft),'r')

title('N = 32, \omega\_0 = \pi/4'), ylabel('Imaginary X\_I'),xlabel('Frequency \omega')

hold off

**2.2 d)**

****

****

When the digital frequency is increased to ω0 = 1.1 π/4, the points sampled for the 32-point DFT no longer represent complete cycles of the windowed sinusoid, i.e. the last cycle is incomplete. Therefore, the periodic representation of the signal does not fully represent a sinusoid and has sharp discontinuities where the 32-point sequences are stitched together. As a result, we observe frequency content beyond just the two impulses that were seen in the ω0 = π/4 condition.

%% 2.2d

clear all; close all; clc

w0 = 1.1\*pi/4;

N = 32;

A = 1;

w = (-pi:pi/10000:pi);

Xr = (A/2).\*(cos((w-w0).\*(N-1)/2).\*(sin((w-w0).\*N/2)./sin((w-w0)./2))) ...

+(A/2).\*(cos((w+w0).\*(N-1)/2).\*(sin((w+w0).\*N/2)./sin((w+w0)./2)));

Xi = -(A/2).\*(sin((w-w0).\*(N-1)/2).\*(sin((w-w0).\*N/2)./sin((w-w0)./2))) ...

-(A/2).\*(sin((w+w0).\*(N-1)/2).\*(sin((w+w0).\*N/2)./sin((w+w0)./2)));

n = 1:32;

N = length(n);

x\_dft = A.\*cos(w0\*(n-1));

figure

stem(x\_dft)

X\_dft = fftshift(fft(x\_dft));

w\_dft = (0:2/N:2\*(N-1)/N).\*pi-pi;

figure;

hold on

plot(w,Xr)

stem(w\_dft,real(X\_dft),'r')

title('N = 32, \omega\_0 = 1.1\pi/4'), ylabel('Real X\_R'),xlabel('Frequency \omega')

hold off

% print -dpng -r300 ./hw1\_2-2d\_real.png

figure

hold on

plot(w,Xi)

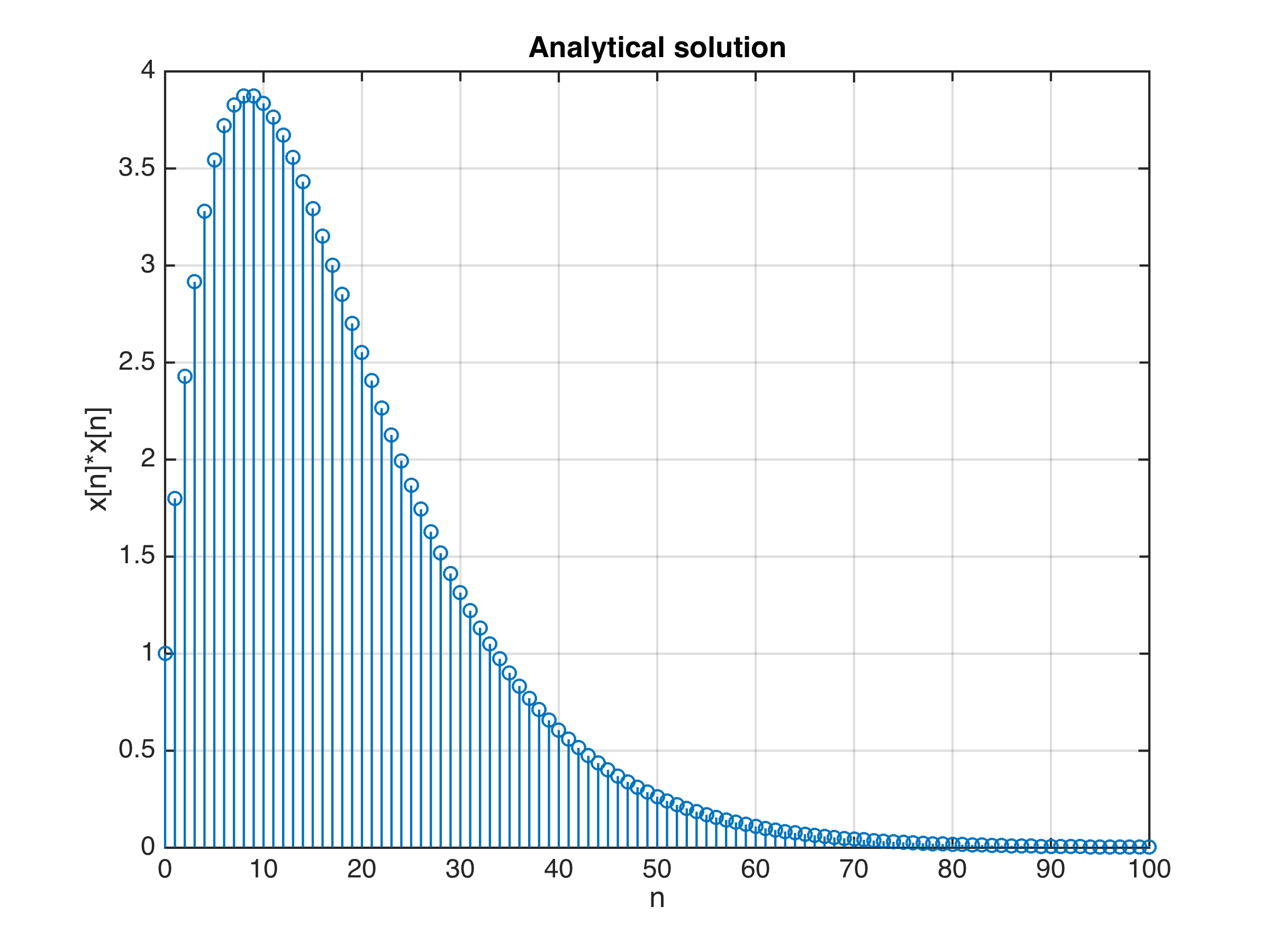
stem(w\_dft,imag(X\_dft),'r')

title('N = 32, \omega\_0 = 1.1\pi/4'), ylabel('Imaginary X\_I'),xlabel('Frequency \omega')

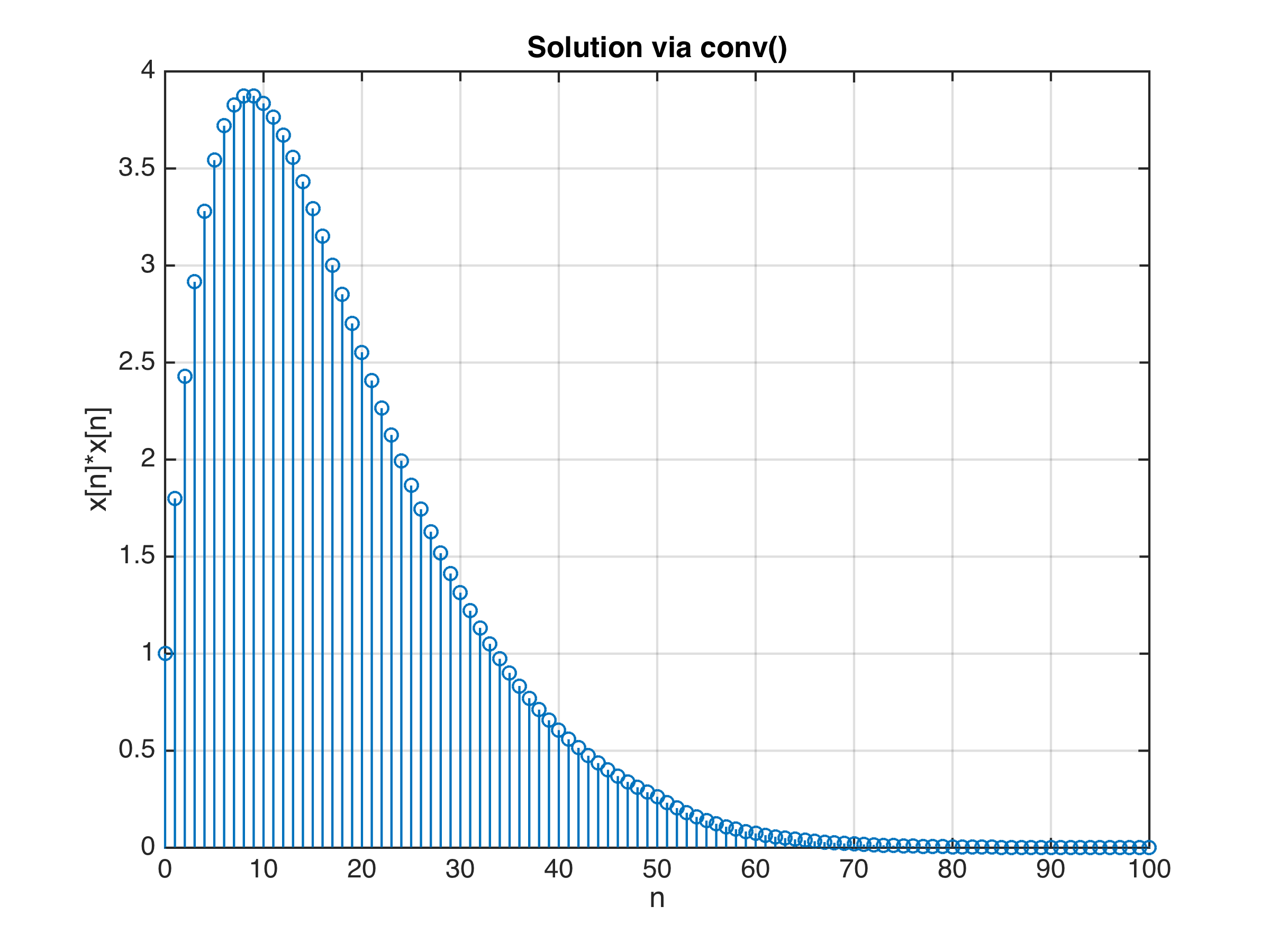
hold off

% print -dpng -r300 ./hw1\_2-2d\_im.png

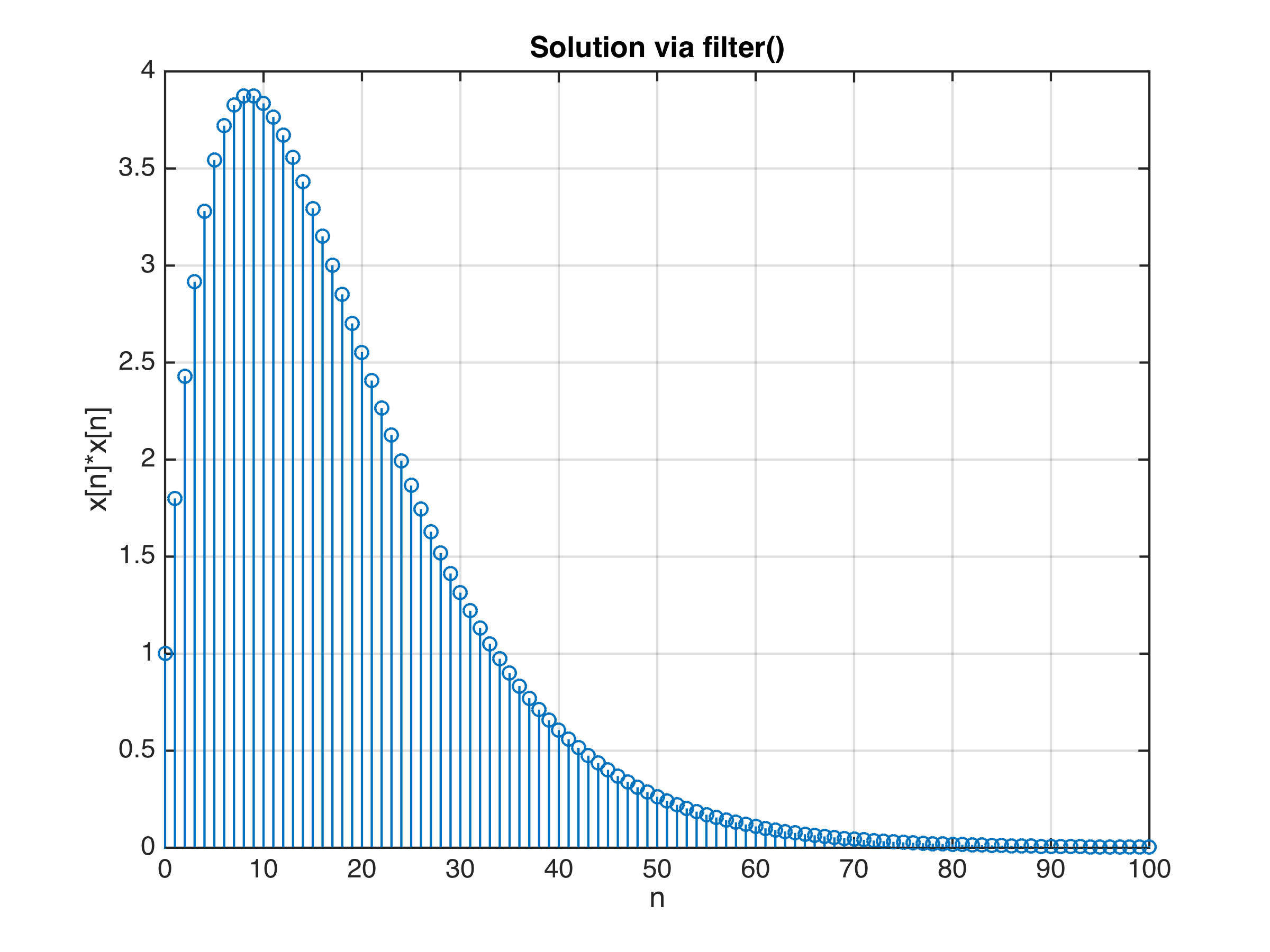
**2.5 a)**

****

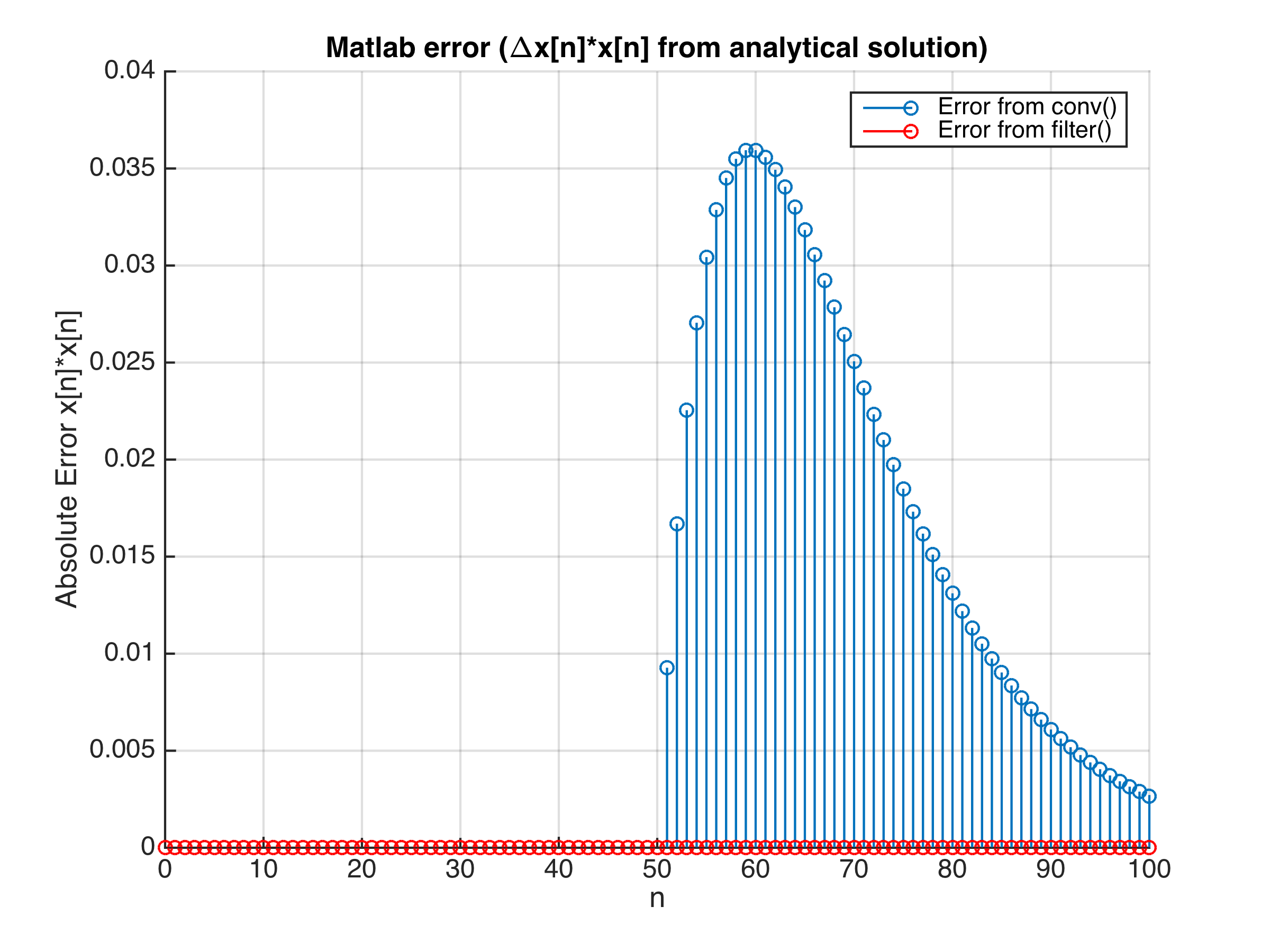
**2.5 b)**

****

**2.5 c)**

****

**2.5 d)**

****

Solutions generated from both the conv() and filter() Matlab functions very closely resemble the expected analytical solution. However, when plotting the absolute error of both methods, the differences in accuracy between the two methods can be observed. Because the conv() requires a signal to be truncated, we can see an increase in error for higher values of n where the signal is truncated. On the other hand, because the filter() method is able to represent the impulse response of a system via coefficients that describe the system in its entirety, this method does not lose significant accuracy with larger signals. Thus, the filter() approach is best suited for infinite length signals since conv() requires both input signals to be truncated.

%% 2.5a-d

clear all; close all; clc

% analytical solution

n\_an = 0:100;

xx\_an = (n\_an+1).\*(0.9).^n\_an;

figure

stem(n\_an,xx\_an)

xlabel('n'), ylabel('x[n]\*x[n]'),title('Analytical solution')

% print -dpng -r300 ./hw1\_2-5a.png

% convolution via matlab

n = 0:50;

x = (0.9).^n;

xx\_conv = conv(x,x);

n\_conv = 0:length(xx\_conv)-1;

figure

stem(n\_conv,xx\_conv)

xlabel('n'), ylabel('x[n]\*x[n]'),title('Solution via conv()')

% print -dpng -r300 ./hw1\_2-5b.png

% filter via matlab

n\_filt = 0:100;

x = (0.9).^n\_filt;

b = 1;

a = [1 -0.9];

xx\_filt = filter(b,a,x);

figure

stem(n\_filt,xx\_filt)

xlabel('n'), ylabel('x[n]\*x[n]'),title('Solution via filter()')

% print -dpng -r300 ./hw1\_2-5c.png

figure

hold on

s1 = stem(n\_an,abs(xx\_an-xx\_conv));

s2 = stem(n\_an,abs(xx\_an-xx\_filt),'r');

xlabel('n'), ylabel('Absolute Error x[n]\*x[n]')

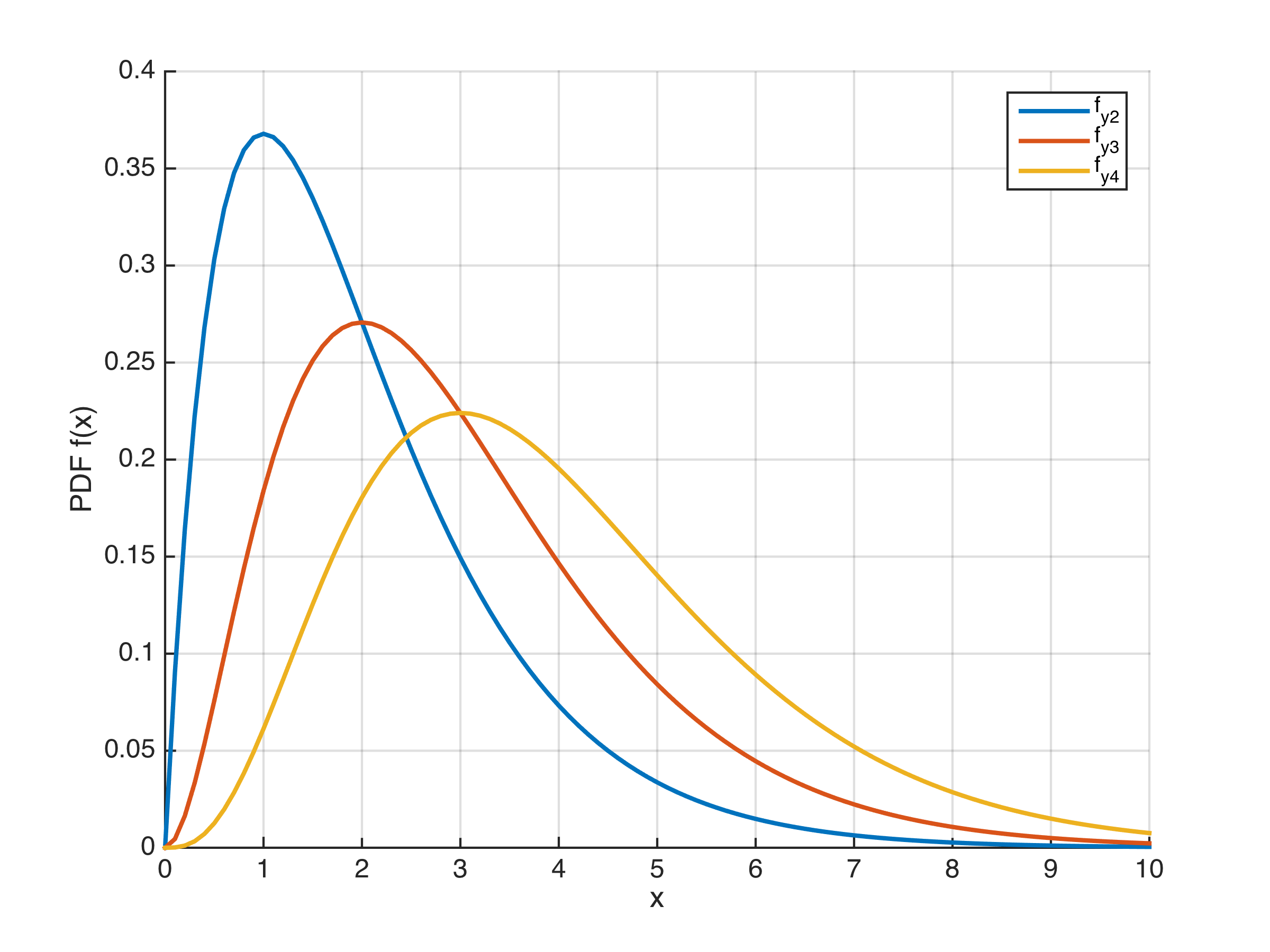
title('Matlab error (\Deltax[n]\*x[n] from analytical solution)')

hold off

legend([s1(1),s2(1)],{'Error from conv()','Error from filter()'})

% print -dpng -r300 ./hw1\_2-5d.png

**3.7 a-d)**

****

As can be seen in the figure above, density functions for increasing values of k converge towards a Gaussian density function. The resulting density function for y4(ξ) thus closely resembles a Gaussian density function with a slight skew.

%% 3.7

clear all; close all; clc

x = 0:0.1:10;

fy2 = x.\*exp(-x);

fy3 = (x.^2/2).\*exp(-x);

fy4 = (x.^3/6).\*exp(-x);

sum(fy4)

sum(fy3)

hold on;

p1 = plot(x,fy2)

p2 = plot(x,fy3)

p3 = plot(x,fy4)

hold off

legend([p1 p2 p3],{'f\_y\_2','f\_y\_3','f\_y\_4'})

xlabel('x'), ylabel('PDF f(x)')

print -dpng -r300 ./hw1\_3.7.png