L

Delayed scale signals 4.....

```
% MATLAB Program for Lab Experiment No. 4

% Define the time and discrete index ranges
t = -5:0.01:5;  % Continuous time for plotting delayed and scaled signals
n = -5:5;       % Discrete index for x[n]

% Part (a): Generate u(t-1) (unit step delayed by 1 second)
u_t_minus_1 = t >= 1; % Unit step shifted by 1 second
figure;
subplot(3, 2, 1);
plot(t, u_t_minus_1, 'b', 'LineWidth', 1.5);
xlabel('t'); ylabel('u(t-1)');
title('Delayed Unit Step u(t-1)');
grid on;

% Part (b): Generate u(2t+1) (unit step scaled and shifted)
u_2t_plus_1 = (2*t + 1) >= 0; % Scale t by 2 and shift left by 1/2
subplot(3, 2, 2);
plot(t, u_2t_plus_1, 'r', 'LineWidth', 1.5);
xlabel('t'); ylabel('u(2t+1)');
title('Scaled and Shifted Unit Step u(2t+1)');
grid on;

% Part (c): Generate r(t-1) (ramp signal delayed by 1 second)
r_t_minus_1 = (t-1) .* (t >= 1); % Ramp signal with delay of 1 second
subplot(3, 2, 3);
plot(t, r_t_minus_1, 'g', 'LineWidth', 1.5);
xlabel('t'); ylabel('r(t-1)');
title('Delayed Ramp Signal r(t-1)');
grid on;

% Part (d): Generate the sinc function
sinc_function = sinc(t); % sinc(t) = sin(pi*t)/(pi*t)
subplot(3, 2, 4);
plot(t, sinc_function, 'm', 'LineWidth', 1.5);
xlabel('t'); ylabel('sinc(t)');
title('Sinc Function');
grid on;
```

```
% Part (e): Discrete signal x[n] and its transformations
% Define x[n] (example signal for clarity)
x = [0, 1, 2, 3, 2, 1, 0, 0, 0, 0, 0]; % x[n] over n = -5:5
x = [zeros(1,5), x, zeros(1,5)]; % Pad x[n] to match shifts

% Generate the discrete index range after padding
n_extended = -10:10;

% (i) x[n] * u(1-n)
u_1_minus_n = n_extended <= 1; % Flip and shift the unit step
x_u_1_minus_n = x .* u_1_minus_n;
subplot(3, 2, 5);
stem(n_extended, x_u_1_minus_n, 'filled', 'k');
xlabel('n'); ylabel('x[n]u(1-n)');
title('x[n]u(1-n)');
grid on;

% (ii) x[n]{u(n-2) - u(n)}
u_n_minus_2 = n_extended >= 2; % u(n-2)
u_n = n_extended >= 0;         % u(n)
x_u_diff = x .* (u_n_minus_2 - u_n);
subplot(3, 2, 6);
stem(n_extended, x_u_diff, 'filled', 'b');
xlabel('n'); ylabel('x[n]{u(n-2) - u(n)}');
title('x[n]{u(n-2) - u(n)}');
grid on;

% Separate figure for Part (e iii) to avoid clutter
figure;
% (iii) x[n]δ(n-1)
delta_n_minus_1 = (n_extended == 1); % Delta function at n = 1
x_delta = x .* delta_n_minus_1;
stem(n_extended, x_delta, 'filled', 'r');
xlabel('n'); ylabel('x[n]δ(n-1)');
title('x[n]δ(n-1)');
grid on;

% Final instructions for clarity
disp('Plots are generated for each part of the lab experiment.');
disp('Check the figures for time-domain and discrete-time signal transformations.');
```

Exp 3.....
Even and odd components respective waveform

```
% MATLAB Program for Lab Experiment No. 3
% Decompose x(n) into its even and odd components

% Define the discrete-time range
n = -15:15; % Range of n to include negative and positive indices

% Define the unit step functions u(n) and u(n-10)
u_n = (n >= 0);        % Unit step u(n)
u_n_minus_10 = (n >= 10); % Unit step u(n-10)

% Define x(n) = u(n) - u(n-10)
x_n = u_n - u_n_minus_10;

% Compute the even and odd components
% Even component: xe(n) = 0.5 * [x(n) + x(-n)]
x_neg_n = fliplr(x_n); % x(-n): Flipped version of x(n)
x_even = 0.5 * (x_n + x_neg_n);

% Odd component: xo(n) = 0.5 * [x(n) - x(-n)]
x_odd = 0.5 * (x_n - x_neg_n);

% Plot the original signal x(n)
figure;
subplot(3, 1, 1);
stem(n, x_n, 'b', 'filled');
xlabel('n'); ylabel('x(n)');
title('Original Signal x(n)');
grid on;

% Plot the even component xe(n)
subplot(3, 1, 2);
stem(n, x_even, 'r', 'filled');
xlabel('n'); ylabel('xe(n)');
title('Even Component of x(n)');
grid on;

% Plot the odd component xo(n)
```

```
subplot(3, 1, 3);
stem(n, x_odd, 'g', 'filled');
xlabel('n'); ylabel('xo(n)');
title('Odd Component of x(n)');
grid on;

% Display message
disp('Plots generated for x(n), its even component xe(n), and odd component xo(n).');
disp('Even and odd components are derived using symmetry properties.');
```

Experiment 2

```
% MATLAB Program for Lab Experiment No. 2
% Signal generation: unit step, unit impulse, unit ramp, and rectangular pulse

% Part (a): Unit Step Signal
% Continuous-time unit step signal (time-shifted)
t_cont = -5:0.01:5; % Continuous time
u_cont_shifted = (t_cont >= 2); % Shifted by 2 units
figure;
subplot(4, 2, 1);
plot(t_cont, u_cont_shifted, 'b', 'LineWidth', 1.5);
xlabel('t'); ylabel('u(t-2)');
title('Continuous-Time Unit Step (u(t-2))');
grid on;

% Discrete-time unit step signal (time-shifted)
n_disc = -10:10; % Discrete-time indices
u_disc_shifted = (n_disc >= 2); % Shifted by 2 units
subplot(4, 2, 2);
stem(n_disc, u_disc_shifted, 'r', 'filled');
xlabel('n'); ylabel('u[n-2]');
title('Discrete-Time Unit Step (u[n-2])');
grid on;

% Part (b): Unit Impulse Signal (time-shifted)
delta_disc = (n_disc == 2); % Delta function at n=2
subplot(4, 2, 3);
stem(n_disc, delta_disc, 'g', 'filled');
xlabel('n'); ylabel('δ[n-2]');
```

```
title('Discrete-Time Unit Impulse (δ[n-2])');
grid on;

% Part (c): Unit Ramp Signal
% Continuous-time ramp signal (time-shifted)
ramp_cont = (t_cont - 2) .* (t_cont >= 2); % Ramp signal starting at t=2
subplot(4, 2, 4);
plot(t_cont, ramp_cont, 'm', 'LineWidth', 1.5);
xlabel('t'); ylabel('r(t-2)');
title('Continuous-Time Ramp Signal (r(t-2))');
grid on;

% Discrete-time ramp signal (time-shifted)
ramp_disc = (n_disc - 2) .* (n_disc >= 2); % Ramp signal starting at n=2
subplot(4, 2, 5);
stem(n_disc, ramp_disc, 'k', 'filled');
xlabel('n'); ylabel('r[n-2]');
title('Discrete-Time Ramp Signal (r[n-2])');
grid on;

% Part (d): Rectangular Pulse
T = 2; % Pulse width
rect_pulse = (t_cont >= -T/2) & (t_cont <= T/2); % Rectangular pulse from -1 to 1
subplot(4, 2, 6);
plot(t_cont, rect_pulse, 'c', 'LineWidth', 1.5);
xlabel('t'); ylabel('Amplitude');
title('Rectangular Pulse (Width T=2)');
grid on;

% Display message
disp('Signals generated:');
disp('- Continuous and discrete-time unit step (time-shifted)');
disp('- Discrete-time unit impulse (time-shifted)');
disp('- Continuous and discrete-time unit ramp (time-shifted)');
disp('- Rectangular pulse of width T=2 and amplitude one');
```

Exp 10 domain and frequency domain

```
% MATLAB Program: Time and Frequency Domain Analysis of Window Functions

% Define parameters for the windows
```

```
N = 64; % Length of the window
fs = 1000; % Sampling frequency (arbitrary for analysis)

% Define the windows
rect_window = rectwin(N);       % Rectangular window
hamming_window = hamming(N);    % Hamming window
hann_window = hann(N);          % Hanning window
blackman_window = blackman(N);  % Blackman window

% Time axis for plotting
n = 0:N-1; % Sample indices

% Frequency axis for FFT
f = (0:N-1)*(fs/N); % Frequency values (in Hz)

% Compute the FFT of the windows
rect_fft = fft(rect_window, N);
hamming_fft = fft(hamming_window, N);
hann_fft = fft(hann_window, N);
blackman_fft = fft(blackman_window, N);

% Normalize FFT for proper magnitude scaling
rect_fft_mag = abs(rect_fft) / max(abs(rect_fft));
hamming_fft_mag = abs(hamming_fft) / max(abs(hamming_fft));
hann_fft_mag = abs(hann_fft) / max(abs(hann_fft));
blackman_fft_mag = abs(blackman_fft) / max(abs(blackman_fft));

% Plot the time-domain windows
figure;
subplot(4, 2, 1);
plot(n, rect_window, 'b');
title('Rectangular Window (Time Domain)');
xlabel('Samples');
ylabel('Amplitude');
grid on;

subplot(4, 2, 3);
plot(n, hamming_window, 'r');
title('Hamming Window (Time Domain)');
xlabel('Samples');
ylabel('Amplitude');
```

```
grid on;

subplot(4, 2, 5);
plot(n, hann_window, 'g');
title('Hanning Window (Time Domain)');
xlabel('Samples');
ylabel('Amplitude');
grid on;

subplot(4, 2, 7);
plot(n, blackman_window, 'm');
title('Blackman Window (Time Domain)');
xlabel('Samples');
ylabel('Amplitude');
grid on;

% Plot the frequency-domain responses (Magnitude)
subplot(4, 2, 2);
plot(f, 20*log10(rect_fft_mag), 'b'); % Magnitude in dB
title('Rectangular Window (Frequency Domain)');
xlabel('Frequency (Hz)');
ylabel('Magnitude (dB)');
grid on;

subplot(4, 2, 4);
plot(f, 20*log10(hamming_fft_mag), 'r');
title('Hamming Window (Frequency Domain)');
xlabel('Frequency (Hz)');
ylabel('Magnitude (dB)');
grid on;

subplot(4, 2, 6);
plot(f, 20*log10(hann_fft_mag), 'g');
title('Hanning Window (Frequency Domain)');
xlabel('Frequency (Hz)');
ylabel('Magnitude (dB)');
grid on;

subplot(4, 2, 8);
plot(f, 20*log10(blackman_fft_mag), 'm');
title('Blackman Window (Frequency Domain)');
```

```
xlabel('Frequency (Hz)');
ylabel('Magnitude (dB)');
grid on;

% Explain what is being plotted
disp('The left column shows the time-domain representations of the windows.');
disp('The right column shows the frequency-domain magnitude responses (in dB).');
disp('Notice the trade-off between main lobe width and side lobe suppression.');
```

Experiment 8 causal system plot graph

```
% Coefficients of the difference equation
a = [1, -0.9]; % Denominator coefficients (y terms)
b = [1];       % Numerator coefficients (x terms)

% 1. Compute the Transfer Function H(z)
% Using the 'freqz' function to analyze the system
[H, w] = freqz(b, a, 'whole');

% 2. Plot the Pole-Zero Plot
figure;
zplane(b, a);
title('Pole-Zero Plot of H(z)');
grid on;

% 3. Plot the Magnitude and Phase Spectrum
% Magnitude Response
figure;
subplot(2, 1, 1);
plot(w/pi, abs(H)); % Normalized frequency
xlabel('Normalized Frequency (\times\pi rad/sample)');
ylabel('Magnitude');
title('Magnitude Response |H(e^{jw})|');
grid on;

% Phase Response
subplot(2, 1, 2);
plot(w/pi, angle(H));
xlabel('Normalized Frequency (\times\pi rad/sample)');
```

```
ylabel('Phase (radians)');
title('Phase Response \angleH(e^{jw})');
grid on;



% Coefficients of the difference equation
a = [1, -0.9]; % Denominator coefficients (y terms)
b = [1];       % Numerator coefficients (x terms)

% 1. Compute the Transfer Function H(z)
% Using the 'freqz' function to analyze the system
[H, w] = freqz(b, a, 'whole');

% 2. Plot the Pole-Zero Plot
figure;
zplane(b, a);
title('Pole-Zero Plot of H(z)');
grid on;

% 3. Plot the Magnitude and Phase Spectrum
% Magnitude Response
figure;
subplot(2, 1, 1);
plot(w/pi, abs(H)); % Normalized frequency
xlabel('Normalized Frequency (\times\pi rad/sample)');
ylabel('Magnitude');
title('Magnitude Response |H(e^{jw})|');
grid on;

% Phase Response
subplot(2, 1, 2);
plot(w/pi, angle(H));
xlabel('Normalized Frequency (\times\pi rad/sample)');
ylabel('Phase (radians)');
title('Phase Response \angleH(e^{jw})');
grid on;
```

Experiment 9

a) MATLAB Code Circular convolution

```
clc
clear all
close all
x=[1 2];
h=[1 2 4 6 5];
lh=length(h);
lx=length(x);
N=max(lh,lx);
if lh==lx
x1=x;
y1=y;
else if lx>lh
k=lx-lh;
x1=x;
h1=[h,zeros(1,k)];
else
k=lh-lx;
x1=[x,zeros(1,k)];
h1=h;
end
end
% method 1
% for n=1:N
% y(n)=0;
% for k=1:N
% y(n)=y(n)+h1(k)*x1(mod(n-k,N)+1);
% end
% end
% method 2
for n=1:N
y(n)=0;
for k=1:N
if (n-k)>=0
y(n)=y(n)+h1(k)*x1(n-k+1);
else
y(n)=y(n)+h1(k)*x1(N+n-k+1);
end
end
end
verify=cconv(x,h,N);
% check if cconv and y equal
```

b) linear convolution using circular convolution

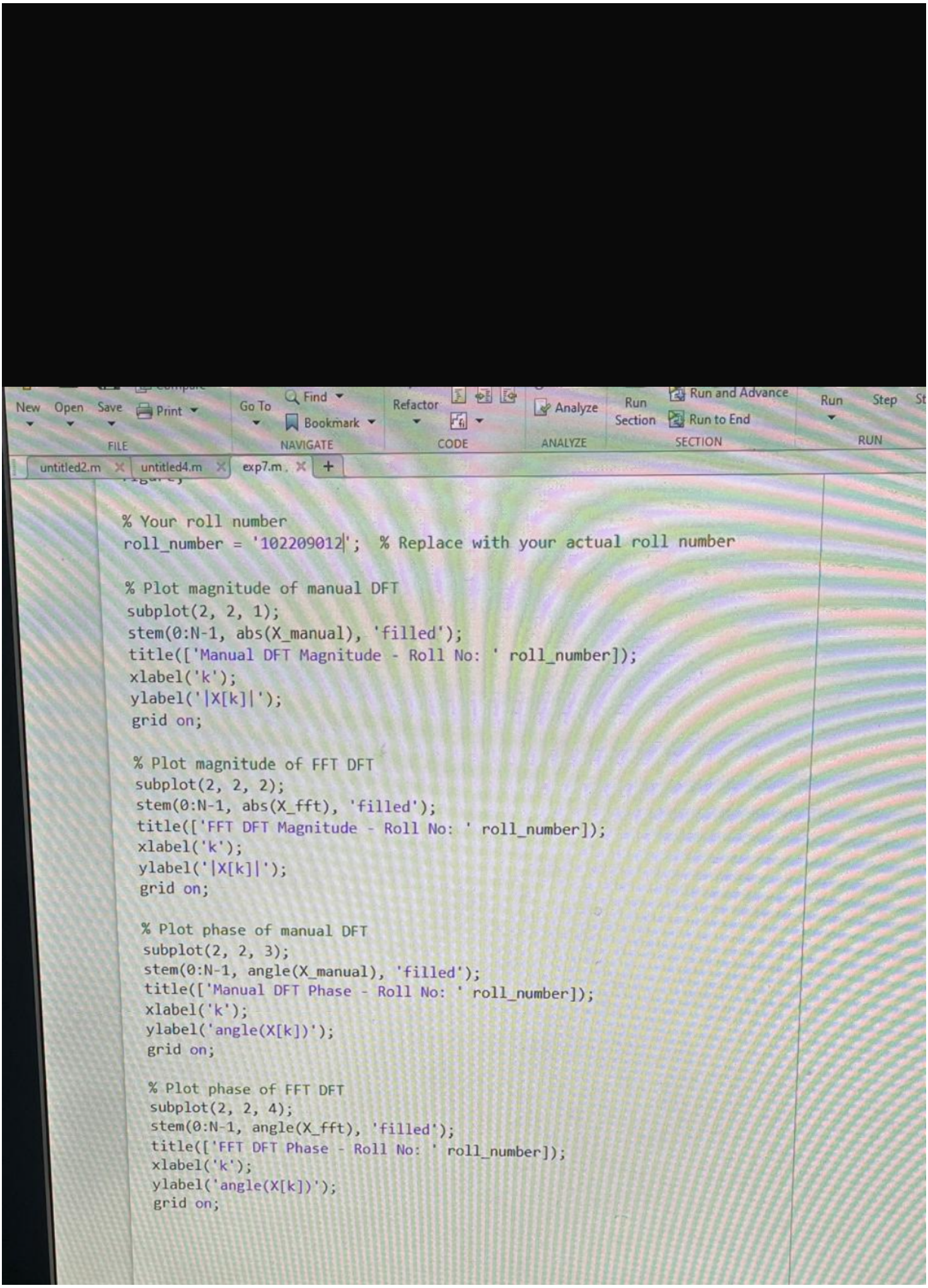
```
clc
clear all
close all
x=[1 2];
h=[1 2 4 6 5];
lh=length(h);
lx=length(x);
tlen=lh+lx-1;
x1=[x zeros(1,tlen-lx)];
h1=[h zeros(1,tlen-lh)];
N=tlen;
for n=1:N
y(n)=0;
for k=1:N
y(n)=y(n)+h1(k)*x1(mod(n-k,N)+1);
end
end
verify=conv(x,h);
```

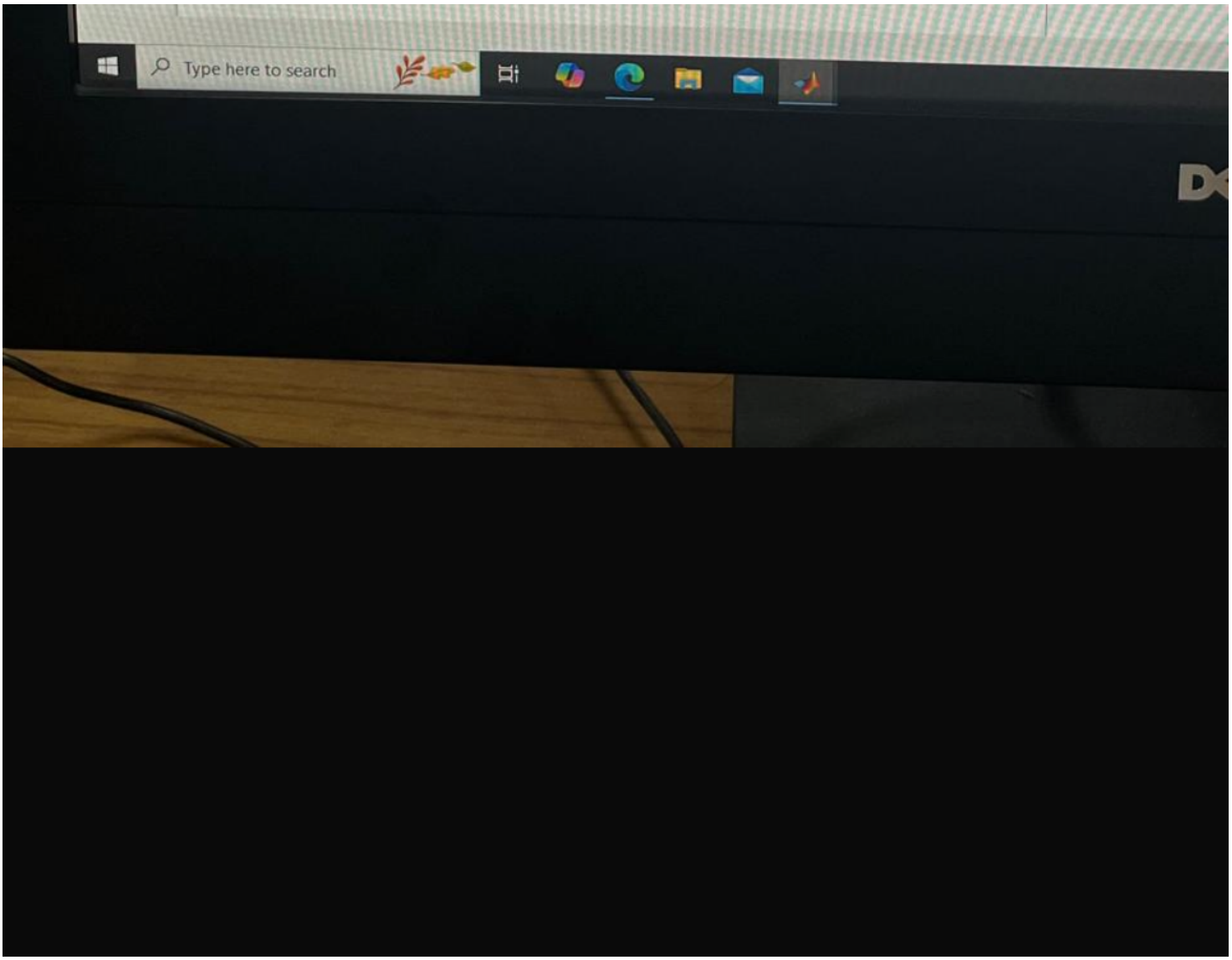
c) circular convolution using linear convolution

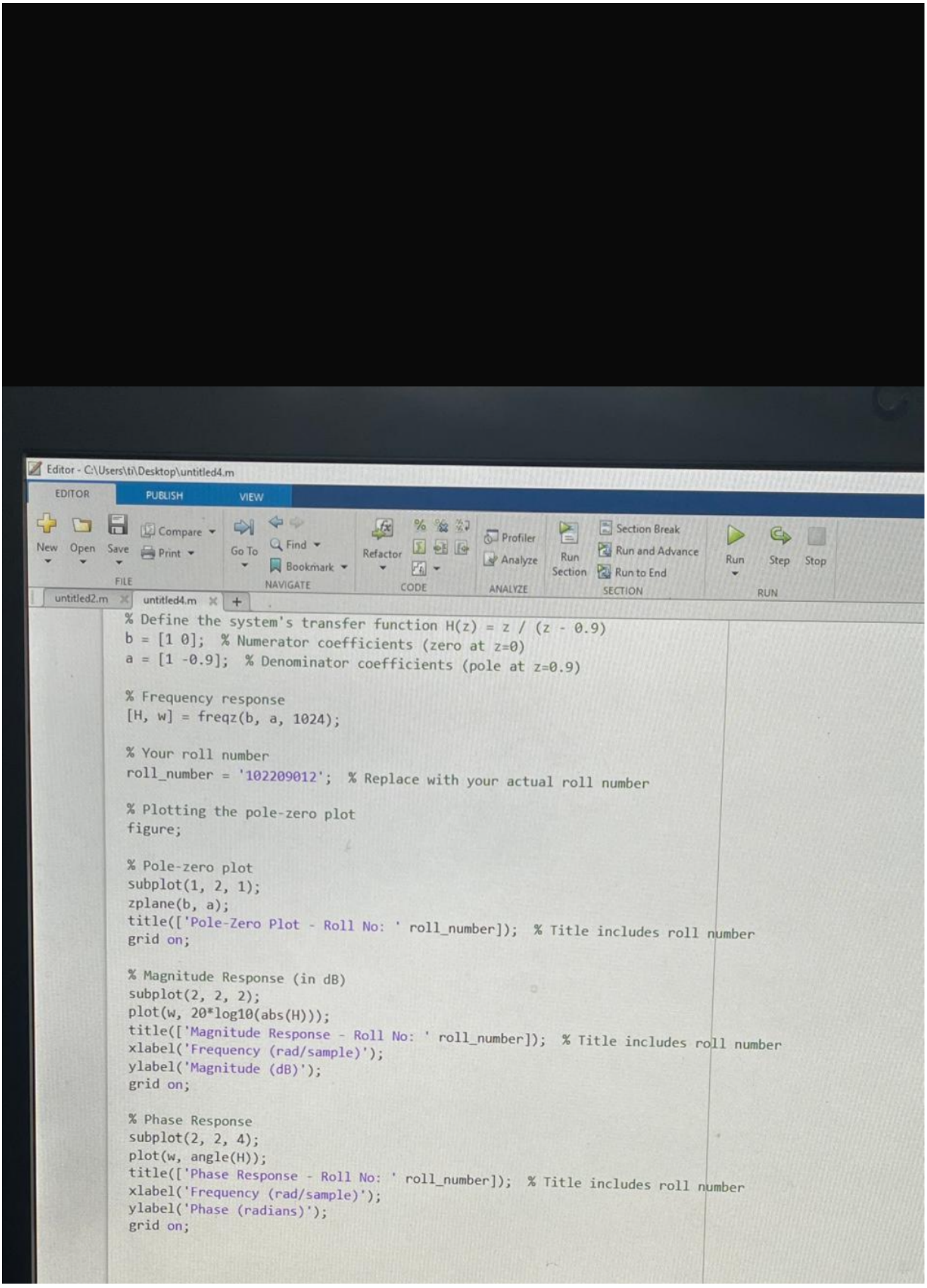
```
clc
clear all
close all
x=[1 2];
h=[1 2 4 6 5];
lh=length(h);
lx=length(x);
c=conv(x,h);
lc=lx+lh-1;
lcc=max(lx,lh);
cnew=[];
% divide the linear convolution in two parts
c1=c(1:lcc);
c2=c(lcc+1:end);
for g=1:lcc
if g>lc-lcc
cnew(g)=c(g);
else
cnew(g)=c1(g)+c2(g);
end
end
```

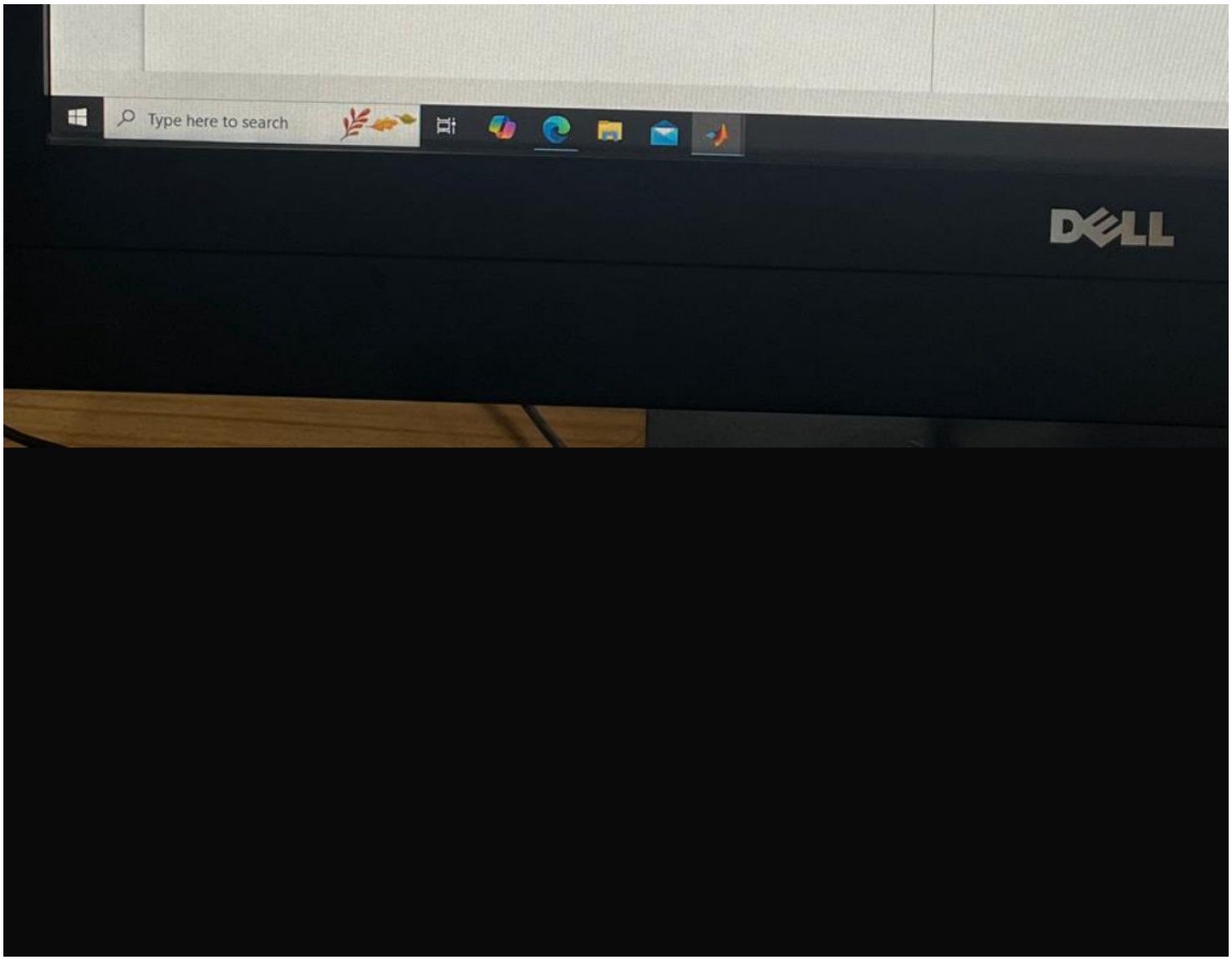
```
verify=cconv(x,h,lcc);
```
d) circular convolution using DFT-IDFT method
```
clc
clear all
close all
x=[1 2];
h=[1 2 4 6 5];
lh=length(h);
lx=length(x);
N=max(lh,lx);
if lh==lx
x1=x;
y1=y;
else if lx>lh
k=lx-lh;
x1=x;
h1=[h,zeros(1,k)];
else
k=lh-lx;
x1=[x,zeros(1,k)];
h1=h;
end
end
% Compute DFT of each sequence
n=0:N-1;
k=0:N-1;
twidf=exp(-j×((2×pi)/N)×n'*k);
Xw=x1×twidf;
Hw=h1×twidf;
CC=(Xw××Hw)×/N;
invtwidf=exp(j×((2×pi)/N)×n'*k);
cc=CC*invtwidf
check1=(ifft(fft(x1).*fft(h1)))
check2=cconv(x1,h1,N)
```
10. Write a program to determine the time domain and frequency domain response of windows.
```
clc
clear all
close all
M = 45;
n = 0:M-1;
% time domain windows
```

```
rect = ones(1,M);
han = 0.5-0.5×cos(2×pi×n/(M-1));
ham = 0.54 - 0.46 * cos(2*pi*n/(M-1));
blac = 0.42 - 0.5 * cos(2*pi*n/(M-1)) + 0.08 * cos(4*pi*n/(M-
1));
% compute frequecny response of windows
N=1024;
faxis= 0:1/N : (N-1)/N;
fftrect=fft(rect,N);
ffthan = fft(han,N)
```


New
Open
Save
Print
FILE
Go To
Find
Bookmark
NAVIGATE
Refactor
CODE
Analyze
ANALYZE
Run Section
Run and Advance
Run to End
SECTION
Run
Step
RUN
untitled2.m
untitled4.m
exp7.m
% Your roll number
roll_number = '102209012'; % Replace with your actual roll number
% Plot magnitude of manual DFT
subplot(2, 2, 1);
stem(0:N-1, abs(X_manual), 'filled');
title(['Manual DFT Magnitude - Roll No: ' roll_number]);
xlabel('k');
ylabel('|X[k]|');
grid on;
% Plot magnitude of FFT DFT
subplot(2, 2, 2);
stem(0:N-1, abs(X_fft), 'filled');
title(['FFT DFT Magnitude - Roll No: ' roll_number]);
xlabel('k');
ylabel('|X[k]|');
grid on;
% Plot phase of manual DFT
subplot(2, 2, 3);
stem(0:N-1, angle(X_manual), 'filled');
title(['Manual DFT Phase - Roll No: ' roll_number]);
xlabel('k');
ylabel('angle(X[k])');
grid on;
% Plot phase of FFT DFT
subplot(2, 2, 4);
stem(0:N-1, angle(X_fft), 'filled');
title(['FFT DFT Phase - Roll No: ' roll_number]);
xlabel('k');
ylabel('angle(X[k])');
grid on;


Type here to search


Editor - C:\Users\ti\Desktop\untitled4.m
EDITOR
PUBLISH
VIEW
New
Open
Save
Compare
Print
FILE
Go To
Find
Bookmark
NAVIGATE
Refactor
CODE
Profiler
Analyze
ANALYZE
Run Section
Section Break
Run and Advance
Run to End
SECTION
Run
Step
Stop
RUN
untitled2.m
untitled4.m
% Define the system's transfer function H(z) = z / (z - 0.9)
b = [1 0]; % Numerator coefficients (zero at z=0)
a = [1 -0.9]; % Denominator coefficients (pole at z=0.9)
% Frequency response
[H, w] = freqz(b, a, 1024);
% Your roll number
roll_number = '102209012'; % Replace with your actual roll number
% Plotting the pole-zero plot
figure;
% Pole-zero plot
subplot(1, 2, 1);
zplane(b, a);
title(['Pole-Zero Plot - Roll No: ' roll_number]); % Title includes roll number
grid on;
% Magnitude Response (in dB)
subplot(2, 2, 2);
plot(w, 20*log10(abs(H)));
title(['Magnitude Response - Roll No: ' roll_number]); % Title includes roll number
xlabel('Frequency (rad/sample)');
ylabel('Magnitude (dB)');
grid on;
% Phase Response
subplot(2, 2, 4);
plot(w, angle(H));
title(['Phase Response - Roll No: ' roll_number]); % Title includes roll number
xlabel('Frequency (rad/sample)');
ylabel('Phase (radians)');
grid on;


Type here to search
DELL