# DIGITAL SIGNAL PROCESSING – EC5010 LABORATORY SESSION 1

# DIGITAL SIGNAL PROCESSING THEORY AND APPLICATION

WIMALASOORIYA G.H.N.P.D.

2022/E/039

GROUP CG03

SEMESTER 05

28 APRIL 2025

PART1: SAMPLING, TIME DOMAIN & FREQUENCY DOMAIN REPRESENTATION

Generate the digital signal, x[n], obtained in your pre-lab.

$$y[n]=0.2+x(n) \quad (1)$$

Plot y[n] scaling the x-axis to ms.

# PART 01

1.)

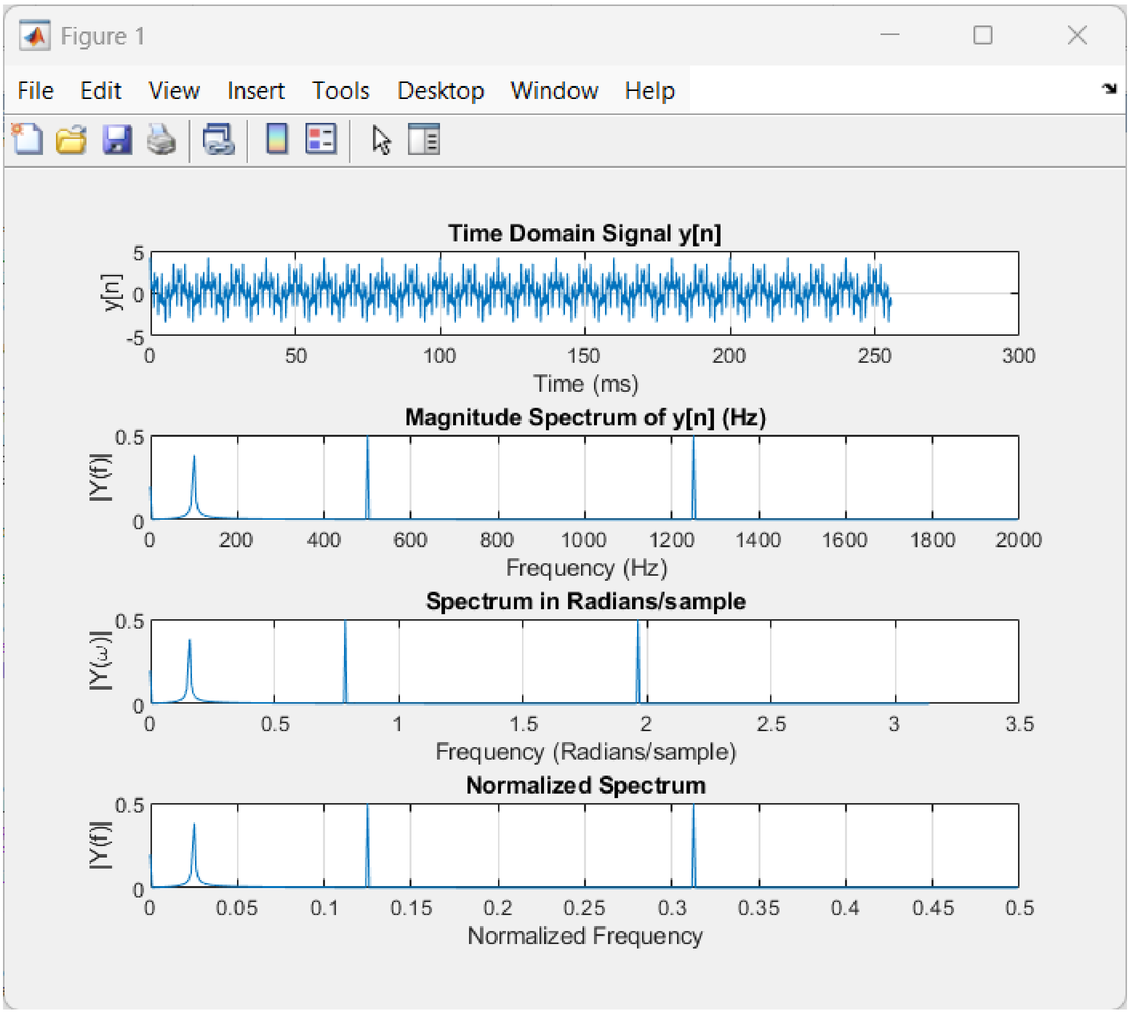
## CODE

Editor - C:\Users\dcrea\OneDrive - University of Jaffna\5th sem\EC5011 - Digital Signal Proc

part1.m

```
%Sampling info
fs = 4000;
T = 1/fs;
N = 1024;
t = (0:N-1)*T;
%signal definition
x = cos(2*pi*100*t)+cos(2*pi*500*t)+cos(2*pi*2000*t)+cos(2*pi*2750*t);
y = 0.2 + x;

%----- Frequency Domain (FFT)----
Y = fft(y);
Y_mag = abs(Y)/N;
Y_half = Y_mag(1:N/2);
f = (0:N/2-1)*(fs/N);
w = 2*pi*f/fs;
f_norm = f/fs;

%-------Subplots-------
figure;
%------ Time Domain Plot-----
subplot(4,1,1);
plot(t*1000, y);
xlabel('Time (ms)');
ylabel('y[n]');
title('Time Domain Signal y[n]');
grid on;

%plot in Hz
subplot(4,1,2)
plot(f,Y_half);
xlabel('Frequency (Hz)');
ylabel('|Y(f)|');
title('Magnitude Spectrum of y[n] (Hz)');
grid on;

% Frequency Spectrum (Radians/sample)
subplot(4,1,3);
plot(w, Y_half);
xlabel('Frequency (Radians/sample)');
ylabel('|Y(\omega)|');
title('Spectrum in Radians/sample');
grid on;

% Frequency Spectrum (Normalized)
subplot(4,1,4);
plot(f_norm, Y_half);
xlabel('Normalized Frequency');
ylabel('|Y(f)|');
title('Normalized Spectrum');
grid on;
```

## OUTPUT




2.)

The DC value = 0.2

3.)

$f_a = | f - k \cdot f_s |$ such that $f_a < f_s/2$

$f_a = |2750 - 4000| = 1250$ Hz

* Aliased frequency component = 1250 Hz

4.)

**CODE**

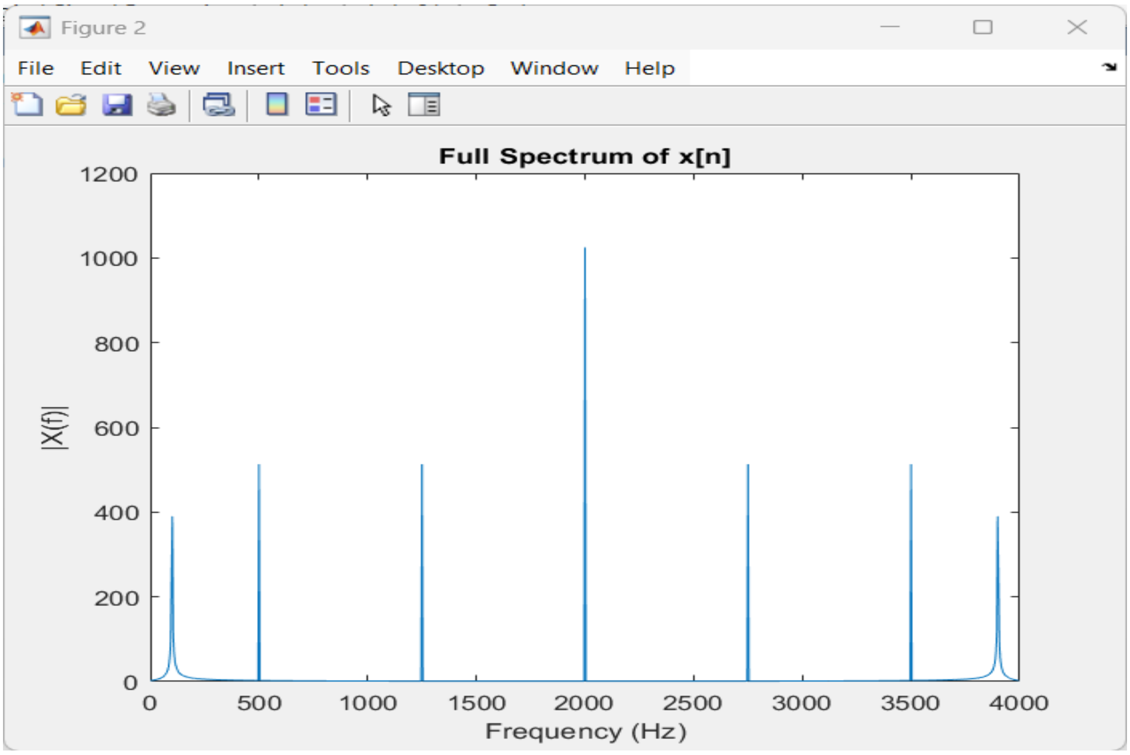
Editor - C:\Users\dcrea\OneDrive - University of Jaffna\5th sem\EC5011 - Digital Signal Processing

part1.m

part1_4.m

```
%Plot Info
N = length(x);
X = abs(fft(x));
F = (0:N-1)*(fs/N);

% Plot full frequency spectrum
figure;
plot(F, X);
title('Full Spectrum of x[n]');
xlabel('Frequency (Hz)');
ylabel('|X(f)|');
xlim([0 fs]);  % Plot full 0 to fs
```

**OUTPUT**



5.)

# CODE

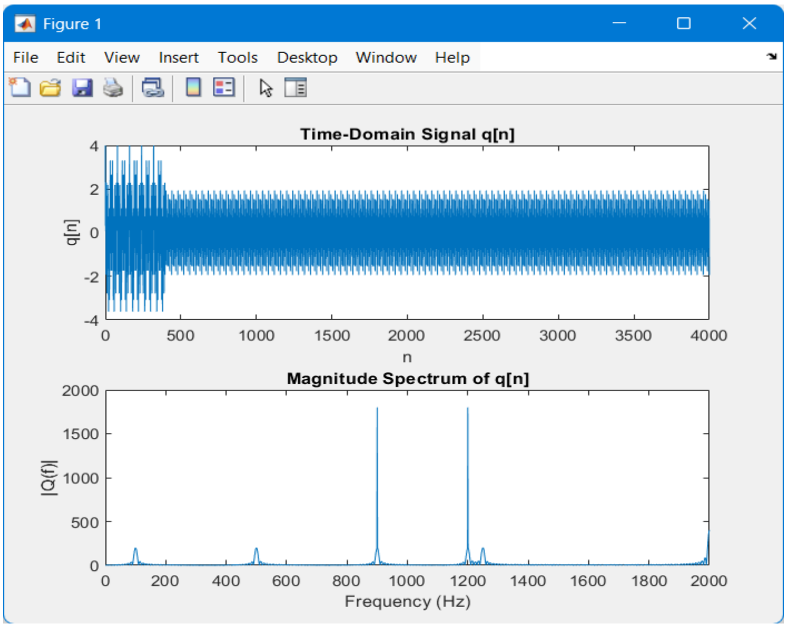
Editor - C:\Users\dcrea\OneDrive - University of Jaffna\5th sem\EC5011 - Digital Signal Processing\Labs\Lab_01\Code\pa

part1.m

part1_5.m

```
% Sampling setup
fs = 4000;          % Sampling frequency
n_x = 0:399;        % 400 samples for x[n]
n_p = 0:3599;       % 3600 samples for p[n]

% x[n]: same as in Q1
x = cos(2*pi*100*n_x/fs) + cos(2*pi*500*n_x/fs) + ...
    cos(2*pi*2000*n_x/fs) + cos(2*pi*2750*n_x/fs);

% p[n]: two sinusoids at 900 Hz and 1200 Hz
p = sin(2*pi*900*n_p/fs) + sin(2*pi*1200*n_p/fs);

% Construct q[n]: x followed by p
q = [x, p];                 % 4000 samples total
n_q = 0:length(q)-1;

% Plot time domain
figure;
subplot(2,1,1);
plot(n_q, q);
xlabel('n'); ylabel('q[n]');
title('Time-Domain Signal q[n]');

% Frequency domain
Q = abs(fft(q));
f = (0:length(q)-1)*fs/length(q);
subplot(2,1,2);
plot(f, Q);
xlabel('Frequency (Hz)'); ylabel('|Q(f)|');
title('Magnitude Spectrum of q[n]');
xlim([0 fs/2]);  % Plot up to Nyquist
```

# OUTPUT



# PART 02

## CODE

Editor - C:\Users\dcrea\OneDrive - University of Jaffna\5th sem\EC5011 - Digital Signal Processing\Labs\Lab_01\Code\part2.m

part... part... part... gen...

```matlab
% === PART 2: Audio Signal Time and Frequency Analysis ===

% 1. Read audio file
[x, fs] = audioread('jaffna.wav');
x = x(:,1);  % Use only one channel if stereo

% 2. Plot time-domain waveform
t = (0:length(x)-1)/fs;
figure;
subplot(2,1,1);
plot(t*1000, x);
xlabel('Time (ms)');
ylabel('Amplitude');
title('Time-Domain Signal');

% 3. Plot frequency spectrum
X = fft(x);
N = length(X);
f = (0:N-1)*fs/N;

subplot(2,1,2);
plot(f, abs(X));
xlabel('Frequency (Hz)');
ylabel('|X(f)|');
title('Magnitude Spectrum');
xlim([0 fs/2]);

% 4. Segment audio into words (manually identified regions)
% You can update these indices after zooming into the plot
word1 = x(9000:15000);     % Example segment for word 1
word2 = x(16000:22000);    % Example segment for word 2
word3 = x(23000:28000);    % Example segment for word 3

% 5. Play the segmented words
disp('Playing Word 1...');
sound(word1, fs);
pause(length(word1)/fs + 0.5);

disp('Playing Word 2...');
sound(word2, fs);
pause(length(word2)/fs + 0.5);

disp('Playing Word 3...');
sound(word3, fs);
pause(length(word3)/fs + 0.5);

% 6. Save the words to new WAV files
audiowrite('word1.wav', word1, fs);
audiowrite('word2.wav', word2, fs);
audiowrite('word3.wav', word3, fs);

disp('Words saved as word1.wav, word2.wav, and word3.wav');
```

# OUTPUT

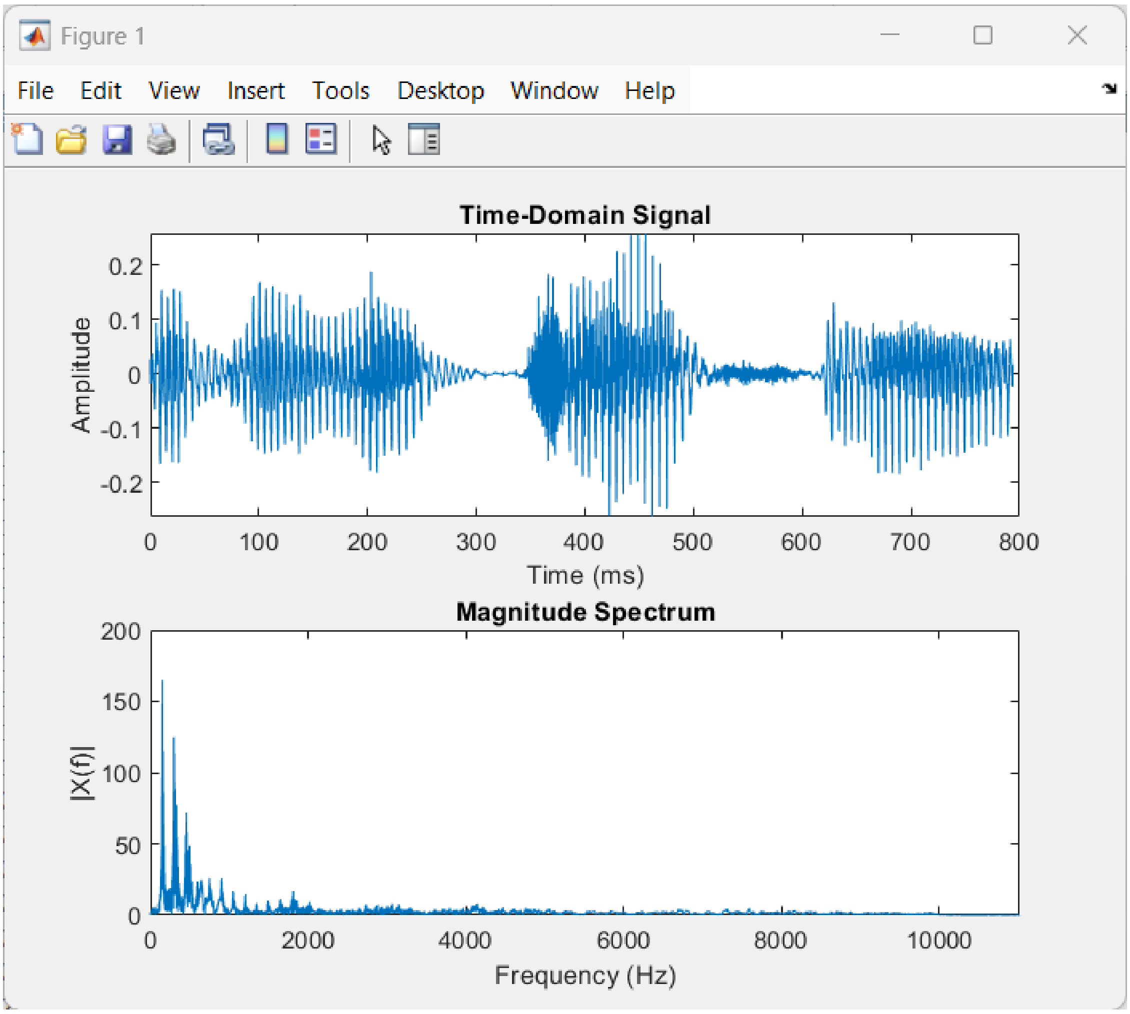


Figure 1
File Edit View Insert Tools Desktop Window Help
Time-Domain Signal
Amplitude
0.2
0.1
0
-0.1
-0.2
0 100 200 300 400 500 600 700 800
Time (ms)
Magnitude Spectrum
|X(f)|
200
150
100
50
0
0 2000 4000 6000 8000 10000
Frequency (Hz)

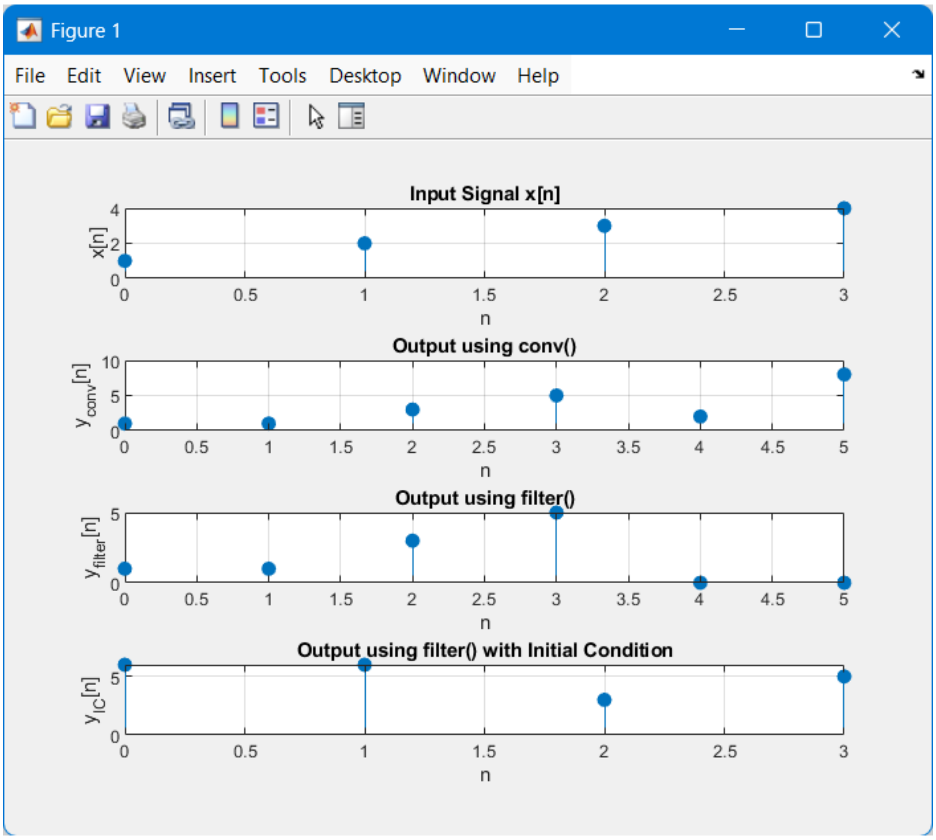
# PART 03

## CODE

Editor - C:\Users\dcrea\OneDrive - University of Jaffna\5th sem\EC5011 - Digital Signal Processing\Labs\Lab_01\Code\part_03

```matlab
% === PART 3: Convolution and Filtering in LTI Systems ===

% Define input signal x[n] and impulse response h[n]
x = [1 2 3 4];            % Example signal x[n]
h = [1 -1 2];             % Example LTI system's impulse response h[n]

% (1) Output using convolution
y_conv = conv(x, h);      % Convolution output

% (2) Output using filter
y_filter = filter(h, 1, x);  % a = 1 (FIR filter), equivalent to convolution

% (3) Compare conv vs filter (needs zero-padding filter output)
y_filter_padded = [y_filter, zeros(1, length(y_conv) - length(y_filter))];

% (4) Explore initial and final conditions
% We'll start with non-zero initial condition
zi = [5 5];  % Initial condition (length = length(h)-1)
[y_with_ic, zf] = filter(h, 1, x, zi);  % Filter with initial condition

% Re-run the same filter starting from final condition
x2 = [0 0 0 0];  % Next input
[y_final_continuation, ~] = filter(h, 1, x2, zf);  % Output using final state

% (5) Plotting all outputs
n1 = 0:length(x)-1;
n2 = 0:length(y_conv)-1;

figure;

subplot(4,1,1);
stem(n1, x, 'filled');
title('Input Signal x[n]');
xlabel('n'); ylabel('x[n]'); grid on;

subplot(4,1,2);
stem(n2, y_conv, 'filled');
title('Output using conv()');
xlabel('n'); ylabel('y_{conv}[n]'); grid on;

subplot(4,1,3);
stem(n2, y_filter_padded, 'filled');
title('Output using filter()');
xlabel('n'); ylabel('y_{filter}[n]'); grid on;

subplot(4,1,4);
stem(0:length(y_with_ic)-1, y_with_ic, 'filled');
title('Output using filter() with Initial Condition');
xlabel('n'); ylabel('y_{IC}[n]'); grid on;

% (6) Explanation:
% y_conv and y_filter match only if length and zero-padding are considered.
% y_with_ic starts higher due to initial condition.
% zf holds the final state of internal delay elements.

% (7) Manual Convolution (step-by-step)
x_pad = [x, zeros(1, length(h)-1)];
h_pad = [h, zeros(1, length(x)-1)];
y_manual = zeros(1, length(x)+length(h)-1);

figure;
for n = 1:length(y_manual)
    for k = 1:length(x)
        if (n-k+1 > 0 && n-k+1 <= length(h))
            y_manual(n) = y_manual(n) + x(k)*h(n-k+1);
        end
    end
    % Plot intermediate output after each step
    subplot(length(y_manual),1,n);
    stem(0:length(y_manual)-1, y_manual, 'filled');
    title(['Manual Convolution Step ' num2str(n)]);
    ylim([min(y_manual)-1, max(y_manual)+1]);
end
```

## OUTPUT




0.6 )

- Convolution output is longer (length = $L_x + L_h - 1$) and shows full filter effect.
- Filter output is same size as x(n)x(n)x(n); assumes causal FIR system.
- Filter with final condition is used when processing a signal in chunks.

0.7 )

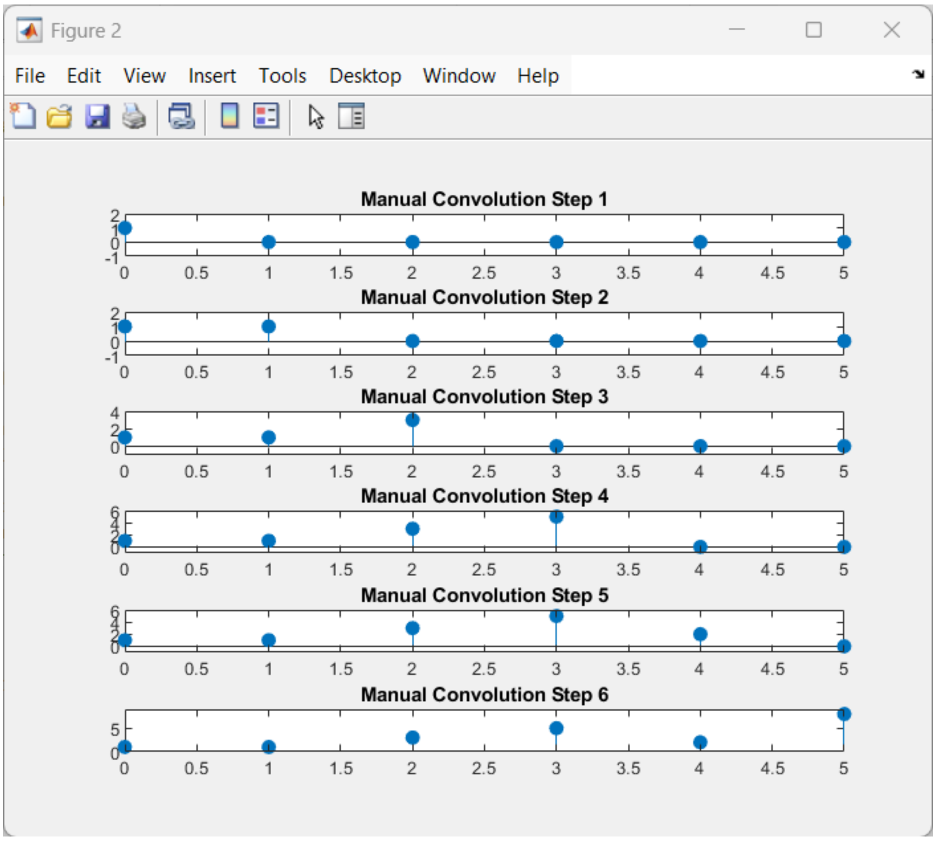


Figure 2
File Edit View Insert Tools Desktop Window Help
Manual Convolution Step 1
Manual Convolution Step 2
Manual Convolution Step 3
Manual Convolution Step 4
Manual Convolution Step 5
Manual Convolution Step 6