1. Basic Command:

Data Manipulation:
Loading data: load filename (reads data from a file)
Creating vectors: x = 1:10 (creates a vector from 1 to 10)
Matrix operations: y = x + 2 (adds 2 to each element of x)
Signal generation: sin(2*pi*f*t) (generates a sinusoid with frequency f)
Plotting:
plot(x, y) (plots x-axis vs. y-axis)
stem(x) (plots discrete samples with markers)
subplot(m, n, p) (creates a subplot in a grid)
xlabel, ylabel, title (add labels and title to plots)
Signal Analysis:
fft(x) (computes the Fast Fourier Transform of a signal)
ifft(X) (performs the Inverse Fast Fourier Transform)
filter(b, a, x) (filters a signal with a filter defined by coefficients b and a)
conv(x, h) (computes the convolution of two signals)
General Commands:
clear (clears all variables)
who (lists all variables in the workspace)
help commandName (displays help for a specific command)

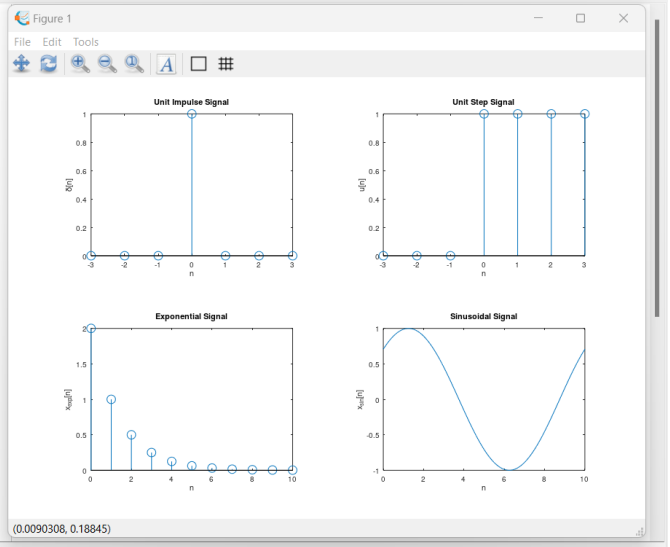
2. Array Declare:

Octave provides various built-in functions to generate specific types of arrays:

- ones(m, n): Creates an m x n matrix filled with ones.
- zeros(m, n): Creates an m x n matrix filled with zeros.
- eye(n): Creates an n x n identity matrix.
- linspace(a, b, n): Creates a vector of n elements linearly spaced between a and b.
- logspace(a, b, n): Creates a vector of n elements logarithmically spaced between a and b.

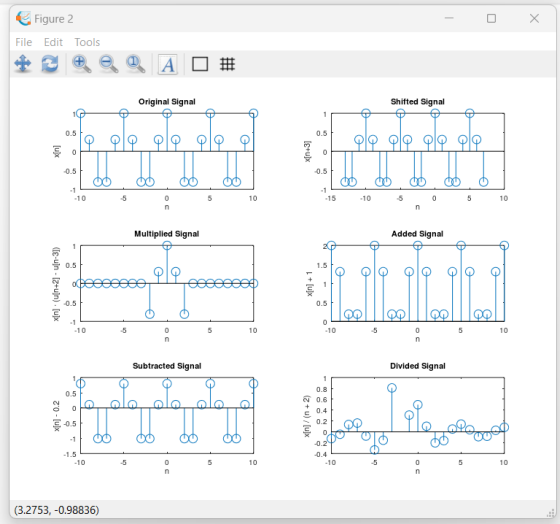
3. Discrete time signal:

```
% Discrete-time signals example

% Unit impulse at n=0
n1 = -3:3;
delta = zeros(size(n1));
delta(n1 == 0) = 1;

% Unit step signal
n2 = -3:3;
u = zeros(size(n2));
u(n2 >= 0) = 1;

% Exponential signal
n3 = 0:10;
A = 2;
alpha = 0.5;
x_exp = A * alpha.^n3;

% Sinusoidal signal
n4 = 0:0.1:10;
A_sin = 1;
omega_sin = 0.2 * pi;
phi_sin = pi/4;
x_sin = A_sin * sin(omega_sin * n4 + phi_sin);

% Plotting
figure;

subplot(2,2,1);
stem(n1, delta);
title('Unit Impulse Signal');
xlabel('n');
ylabel('\delta[n]');

subplot(2,2,2);
stem(n2, u);
title('Unit Step Signal');
xlabel('n');
ylabel('u[n]');

subplot(2,2,3);
stem(n3, x_exp);
title('Exponential Signal');
xlabel('n');
ylabel('x_{exp}[n]');

subplot(2,2,4);
plot(n4, x_sin);
title('Sinusoidal Signal');
xlabel('n');
ylabel('x_{sin}[n]');
```

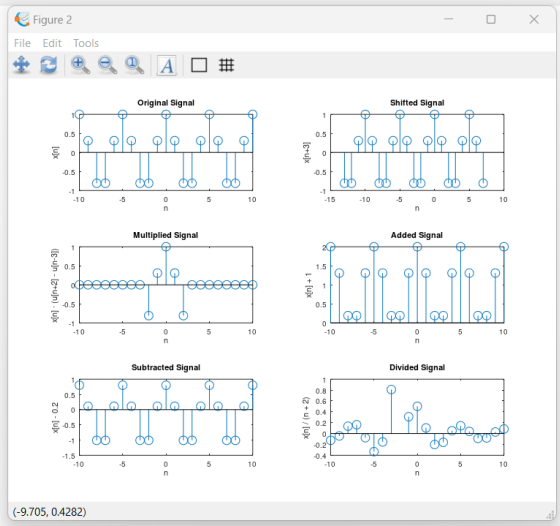


4. Stem & plot, subplot shifting, multiplication, addition, subtraction, and Division.

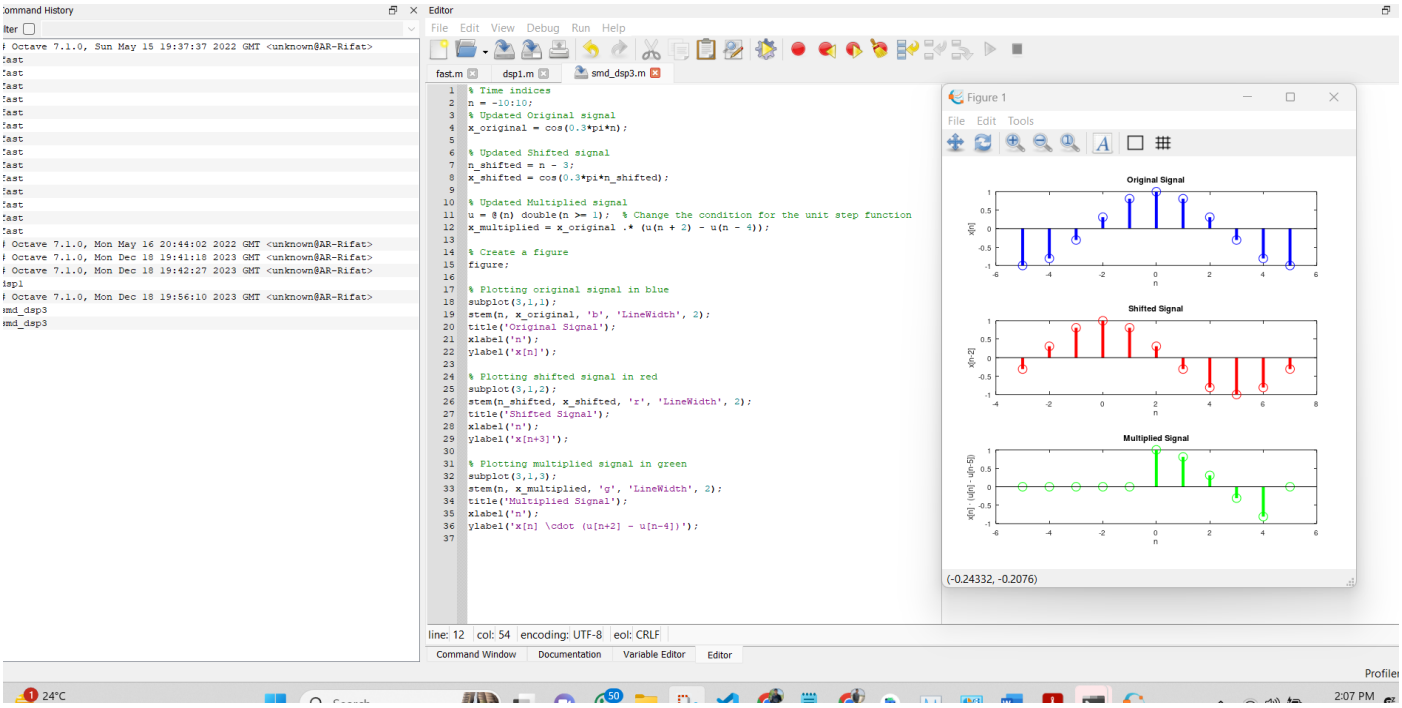
```
%Discrete-time signal operations example

% Time indices
n = -10:10;

% Original signal
x_original = cos(0.4*pi*n);

% Shifting
n_shifted = n - 3;
x_shifted = cos(0.4*pi*n_shifted);

% Multiplication
x_multiplied = x_original .* (u(n + 2) - u(n - 3));

% Addition
x_added = x_original + 1;

% Subtraction
x_subtracted = x_original - 0.2;

% Division
x_divided = x_original ./ (n + 2);

% Plotting
figure;

subplot(3,2,1);
stem(n, x_original);
title('Original Signal');
xlabel('n');
ylabel('x[n]');
```



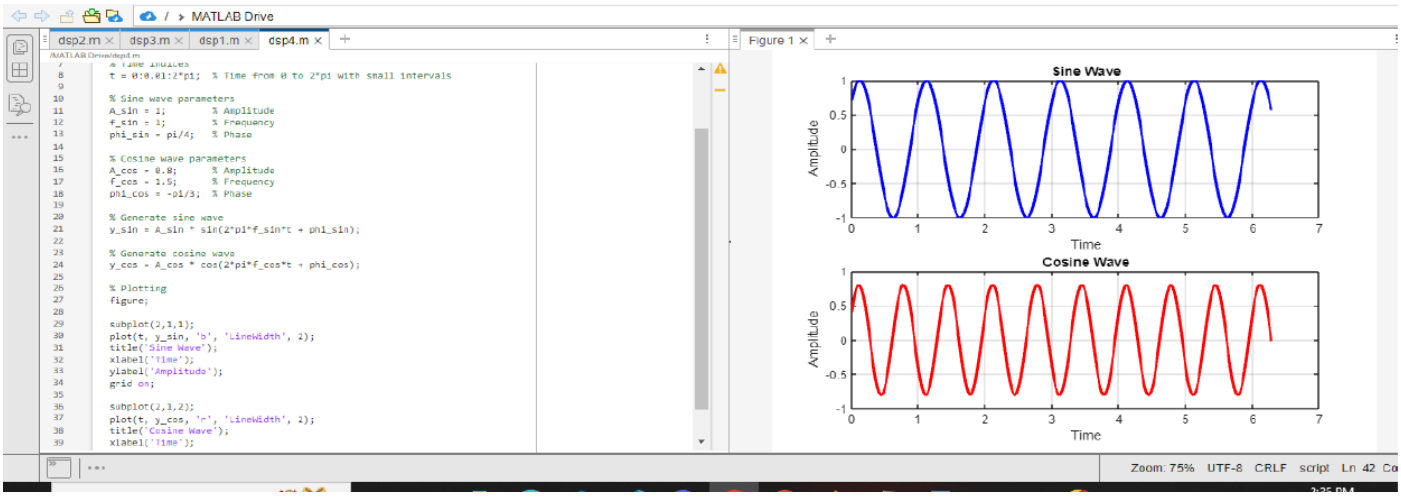

```
xlabel('n');
ylabel('x[n]');

subplot(3,2,2);
stem(n_shifted, x_shifted);
title('Shifted Signal');
xlabel('n');
ylabel('x[n+3]');

subplot(3,2,3);
stem(n, x_multiplied);
title('Multiplied Signal');
xlabel('n');
ylabel('x[n] \cdot (u[n+2] - u[n-3])');

subplot(3,2,4);
stem(n, x_added);
title('Added Signal');
xlabel('n');
ylabel('x[n] + 1');

subplot(3,2,5);
stem(n, x_subtracted);
title('Subtracted Signal');
xlabel('n');
ylabel('x[n] - 0.2');

subplot(3,2,6);
stem(n, x_divided);
title('Divided Signal');
xlabel('n');
ylabel('x[n] / (n + 2)');
```



5. Signal color change field:

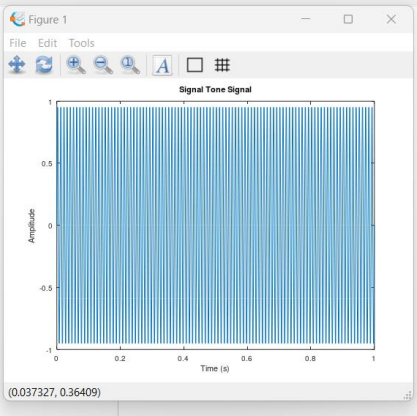


6. Sin Wave and Cos Wave:

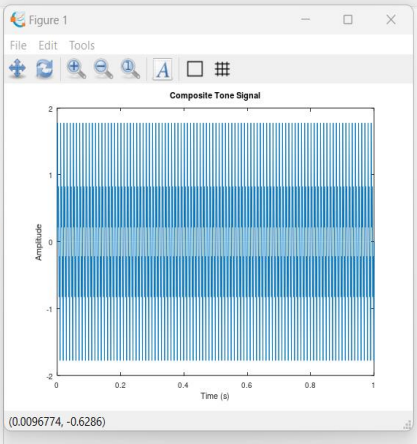


7. Single and Composite tone signal:

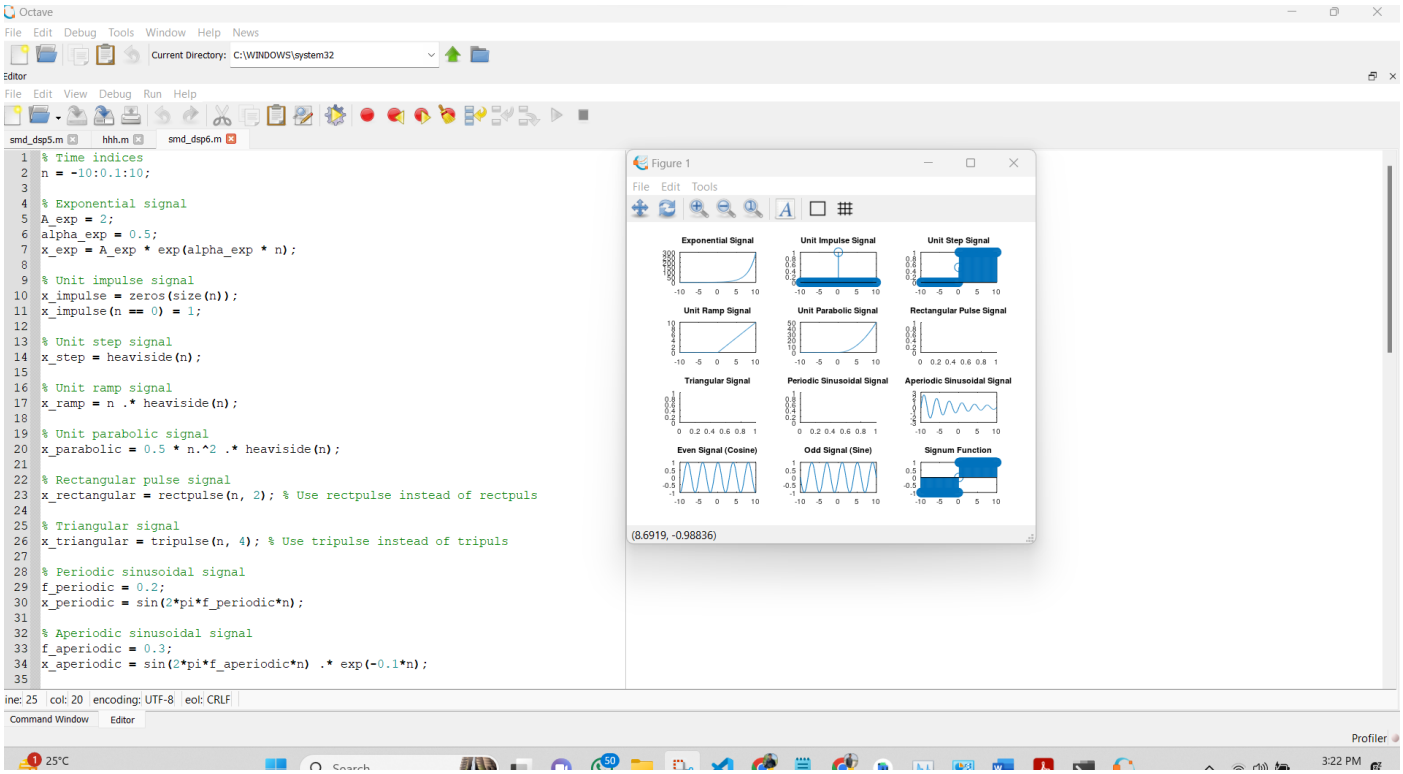
```
% Define frequencies and amplitudes for individual tones
frequency = 100;
amplitude = 1;
time_duration = 1;

%time vetor
t = 0:1/1000:time_duration;

% Generate Signal tone signal
signal_tone = amplitude * sin(2*pi*frequency*t);

% Plot the composite signal
plot(t, signal_tone);
xlabel('Time (s)');
ylabel('Amplitude');
title('Signal Tone Signal');
```



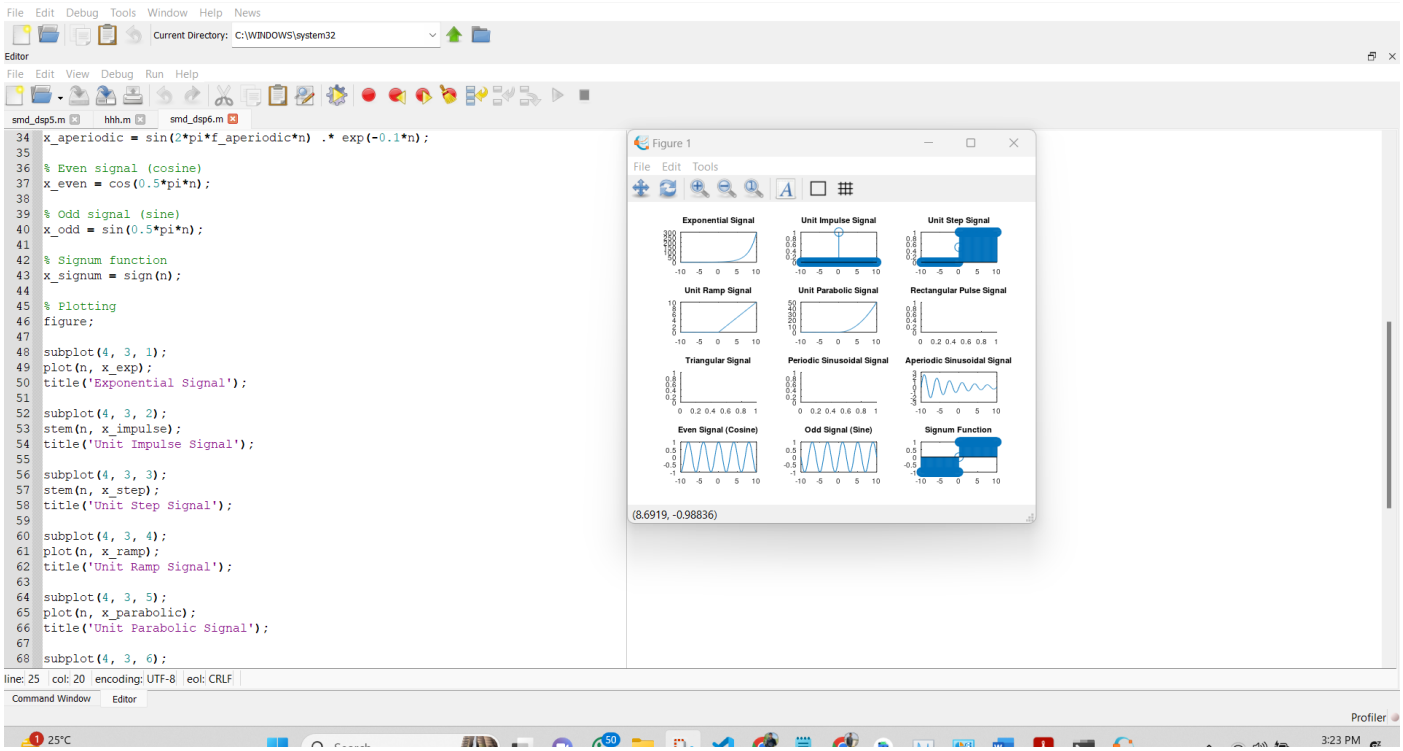

```
% Define frequencies and amplitudes for individual tones
frequencies = [100, 200, 300];
amplitudes = [1, 0.5, 0.75];
time_duration = 1;

%time vetor
t = 0:1/1000:time_duration;

% Initialize composite signal
composite_tone = zeros(size(t));

% Add contributions from each single tone
for i = 1:length(frequencies)
    composite_tone = composite_tone + amplitudes(i) * sin(2*pi*frequencies(i)*t);
end

% Plot the composite signal
plot(t, composite_tone);
xlabel('Time (s)');
ylabel('Amplitude');
title('Composite Tone Signal');
```



8. Exponential Signal, Impulse Signal, Unit Step Signal, Unit Ramp Signal, Unit Parabolic Signal, Rectangular Signal, Triangular Signal, Periodic Signal, Aperiodic Signal, Even & Odd Signals, Signum function.

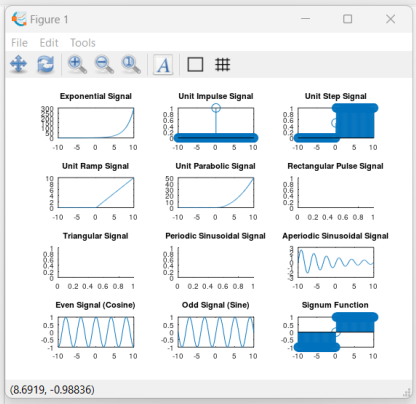


```
% Time indices
n = -10:0.1:10;

% Exponential signal
A_exp = 2;
alpha_exp = 0.5;
x_exp = A_exp * exp(alpha_exp * n);

% Unit impulse signal
x_impulse = zeros(size(n));
x_impulse(n == 0) = 1;

% Unit step signal
x_step = heaviside(n);

% Unit ramp signal
x_ramp = n .* heaviside(n);

% Unit parabolic signal
x_parabolic = 0.5 * n.^2 .* heaviside(n);

% Rectangular pulse signal
x_rectangular = rectpulse(n, 2); % Use rectpulse instead of rectpuls

% Triangular signal
x_triangular = tripulse(n, 4); % Use tripulse instead of tripuls

% Periodic sinusoidal signal
f_periodic = 0.2;
x_periodic = sin(2*pi*f_periodic*n);

% Aperiodic sinusoidal signal
f_aperiodic = 0.3;
x_aperiodic = sin(2*pi*f_aperiodic*n) .* exp(-0.1*n);
```



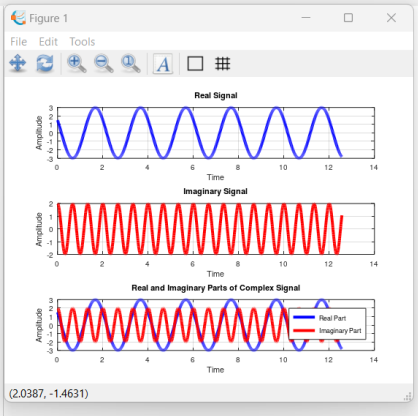
```
x_aperiodic = sin(2*pi*f_aperiodic*n) .* exp(-0.1*n);

% Even signal (cosine)
x_even = cos(0.5*pi*n);

% Odd signal (sine)
x_odd = sin(0.5*pi*n);

% Signum function
x_signum = sign(n);

% Plotting
figure;

subplot(4, 3, 1);
plot(n, x_exp);
title('Exponential Signal');

subplot(4, 3, 2);
stem(n, x_impulse);
title('Unit Impulse Signal');

subplot(4, 3, 3);
stem(n, x_step);
title('Unit Step Signal');

subplot(4, 3, 4);
plot(n, x_ramp);
title('Unit Ramp Signal');

subplot(4, 3, 5);
plot(n, x_parabolic);
title('Unit Parabolic Signal');

subplot(4, 3, 6);
```

```
plot(n, x_ramp);
title('Unit Ramp Signal');

subplot(4, 3, 5);
plot(n, x_parabolic);
title('Unit Parabolic Signal');

subplot(4, 3, 6);
stem(n, x_rectangular);
title('Rectangular Pulse Signal');

subplot(4, 3, 7);
plot(n, x_triangular);
title('Triangular Signal');

subplot(4, 3, 8);
plot(n, x_periodic);
title('Periodic Sinusoidal Signal');

subplot(4, 3, 9);
plot(n, x_aperiodic);
title('Aperiodic Sinusoidal Signal');

subplot(4, 3, 10);
plot(n, x_even);
title('Even Signal (Cosine)');

subplot(4, 3, 11);
plot(n, x_odd);
title('Odd Signal (Sine)');

subplot(4, 3, 12);
stem(n, x_signum);
title('Signum Function');
```

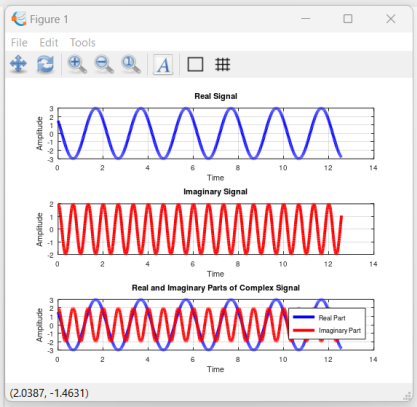


9. Real Signal, Imaginary Signal & Complex Signal:

```
% Time indices
t = 0:0.01:4*pi;  % Increase the range of time

% Real signal
A_real = 3;  % Increase the amplitude
f_real = 0.5;  % Decrease the frequency
phi_real = pi/3;  % Change the phase
x_real = A_real * cos(2*pi*f_real*t + phi_real);

% Imaginary signal
A_imag = 2;
f_imag = 1.5;  % Change the frequency
phi_imag = pi/2;  % Change the phase
x_imag = A_imag * sin(2*pi*f_imag*t + phi_imag);

% Complex signal
x_complex = x_real + 1i * x_imag;

% Plotting
figure;

% Plotting real signal
subplot(3, 1, 1);
plot(t, x_real, 'b', 'LineWidth', 2);
title('Real Signal');
xlabel('Time');
```

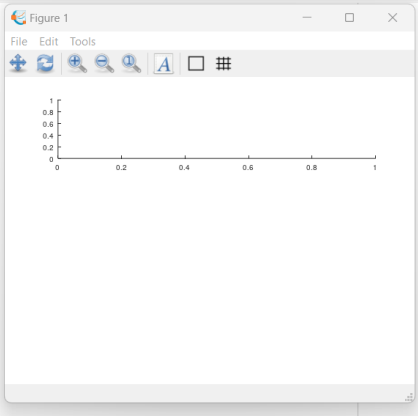


```
% Plotting real signal
subplot(3, 1, 1);
plot(t, x_real, 'b', 'LineWidth', 2);
title('Real Signal');
xlabel('Time');
ylabel('Amplitude');
grid on;

% Plotting imaginary signal
subplot(3, 1, 2);
plot(t, x_imag, 'r', 'LineWidth', 2);
title('Imaginary Signal');
xlabel('Time');
ylabel('Amplitude');
grid on;

% Plotting complex signal
subplot(3, 1, 3);
plot(t, real(x_complex), 'b', 'LineWidth', 2); hold on;
plot(t, imag(x_complex), 'r', 'LineWidth', 2);
title('Real and Imaginary Parts of Complex Signal');
xlabel('Time');
ylabel('Amplitude');
legend('Real Part', 'Imaginary Part');
grid on;
```

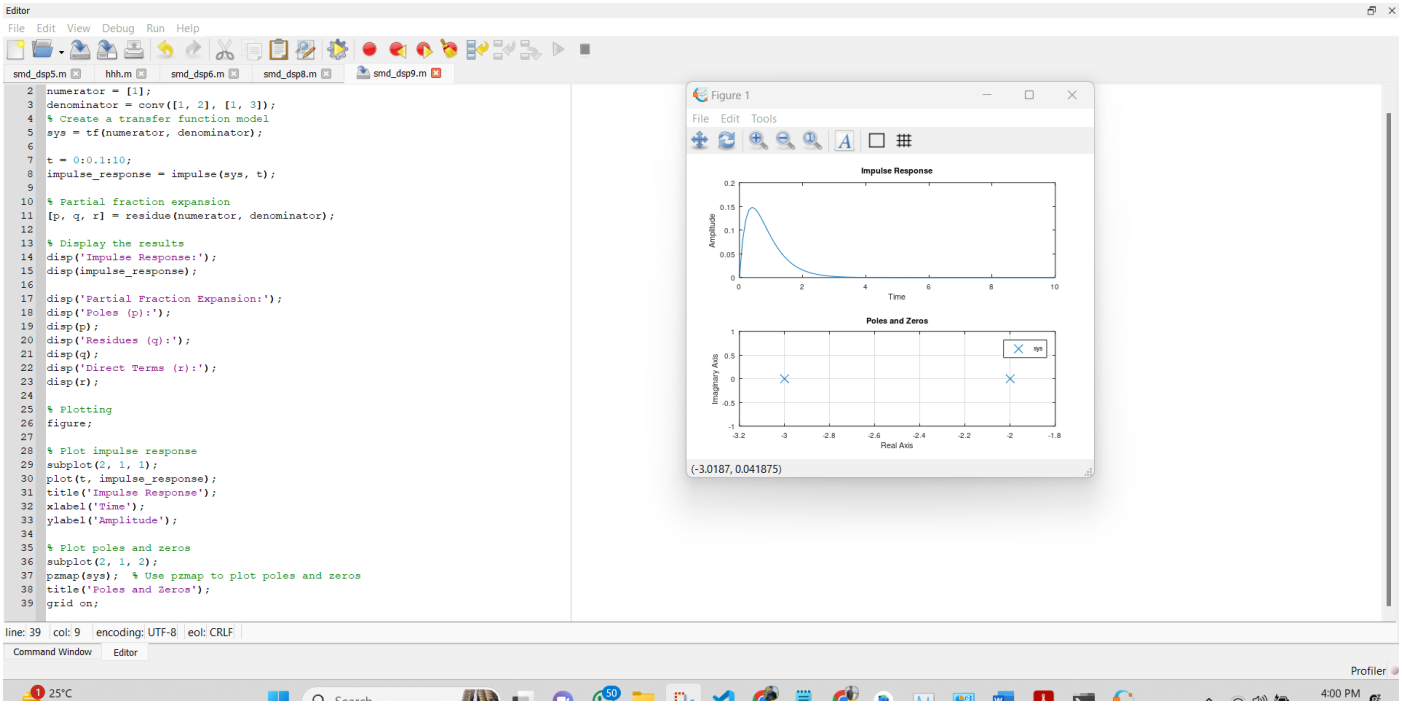


10. Convolution:

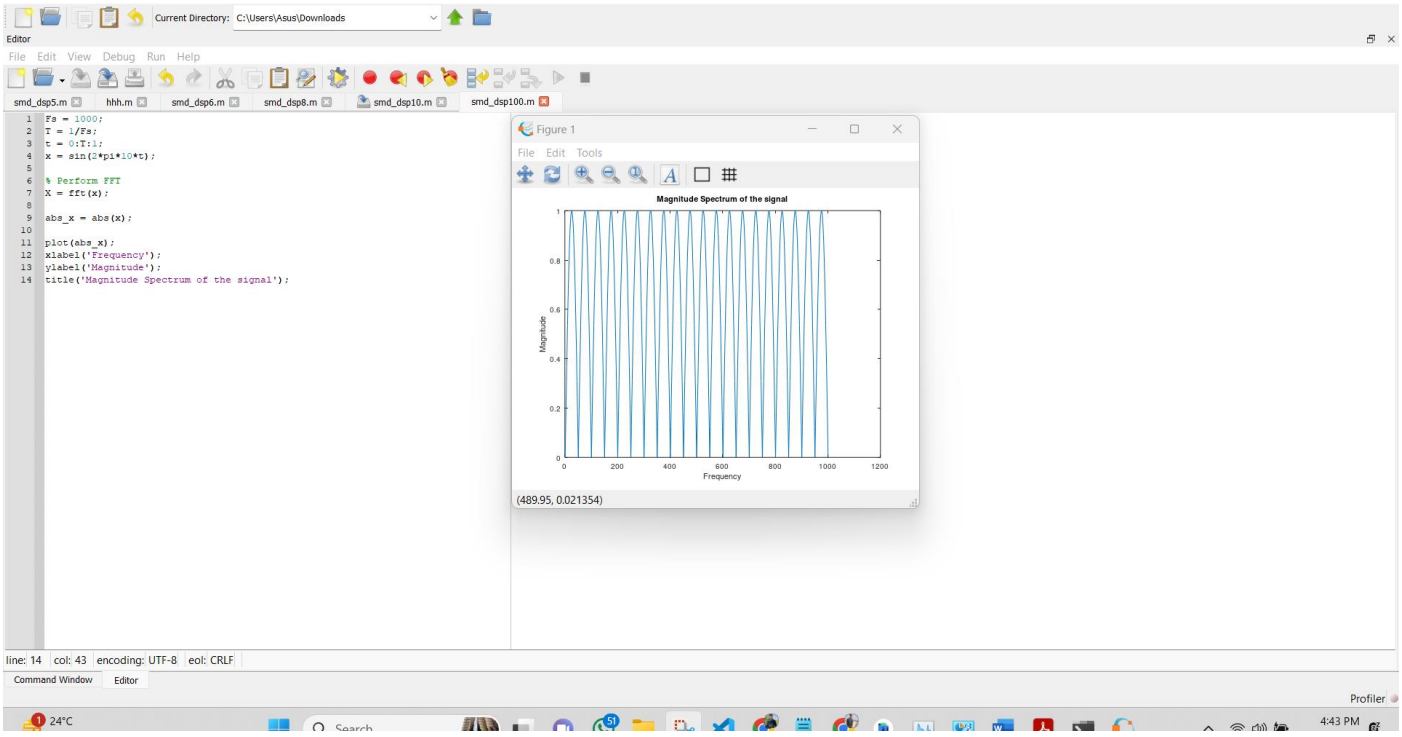
```
n1 = 0:4;
n2 = 0:2;
x1 = [1, 2, 1];
x2 = [1, -1];
% Convolution
y = conv(x1, x2);

% Plotting
figure;
subplot(3, 1, 1);
stem(n1, x1, 'b', 'LineWidth', 2);
title('Signal 1');
xlabel('n');
ylabel('x_1[n]');
grid on;
```

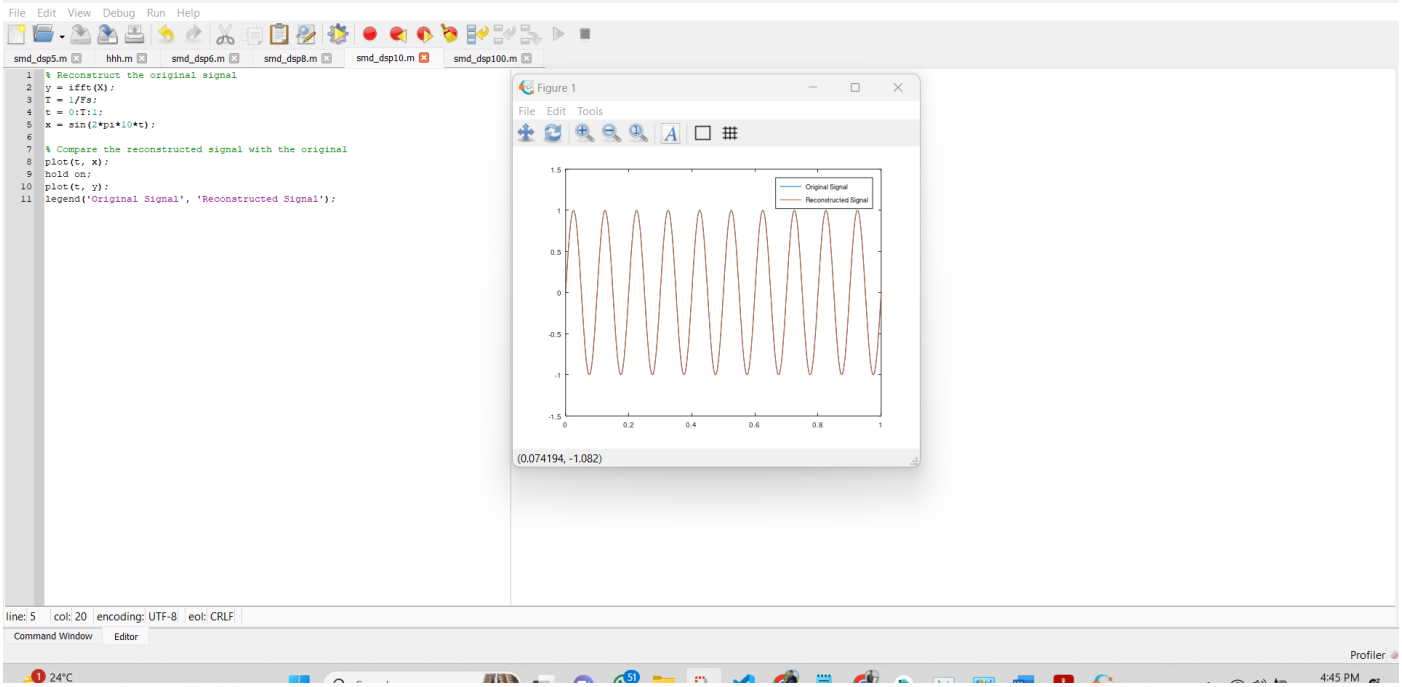


11. Partial fraction expansion impulse response of an LTI system:



12. Fourier & Inverse Fourier:





13. Z-transforms & Inverse Z-transform: