SIGNALS AND SYSTEMS LAB (CSE-301L)

Spring 2024, 4th Semester

Lab Report 01

Submitted by**: Hassan Zaib Jadoon**

Registration Number**: 22PWCSE2144**

Section: **A**

"On my honor, as a student at the University of Engineering and Technology Peshawar, I have neither given nor received unauthorized assistance on this academic work."

Recoverable Signature

X [illegible]

Hassan Zaib Jadpon
Student
Signed by: 506ecb3d-aa56-4353-aa60-5abf648a668d

Signature:

Submitted To: Dr. Safdar Nawaz Khan Marwat

Department of Computer Systems Engineering
University of Engineering and Technology Peshawar

Lab 04

Task 1:

Write a MATLAB function zprint, which takes a complex number and returns it real part,
imaginary part, magnitude, phase in radians, and phase in degrees.
A sample run of program is:

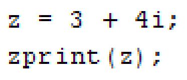
```
>> zprint(z)
Z = X + jY Magnitude Phase Ph(deg)
3 4 5 0.927 53.13
```

Problem Statement:
Develop a MATLAB function **zprint** that accepts a complex number and returns its real part, imaginary part, magnitude, phase in radians, and phase in degrees. The function should display these values in a specified format.

Algorithm:

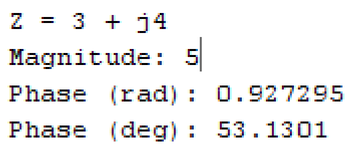
1. Accept a complex number **z** as input to the function **zprint**.
2. Extract the real and imaginary parts of the complex number.
3. Compute the magnitude of **z** using the formula.
4. Compute the phase of **z** in radians using the formula.
5. Convert the phase from radians to degrees.
6. Display the results numerically in the format: "Z = X + jY Magnitude Phase Ph(deg)"

Code:



```
z = 3 + 4i;
zprint(z);
```

Output:



```
Z = 3 + j4
Magnitude: 5
Phase (rad): 0.927295
Phase (deg): 53.1301
```

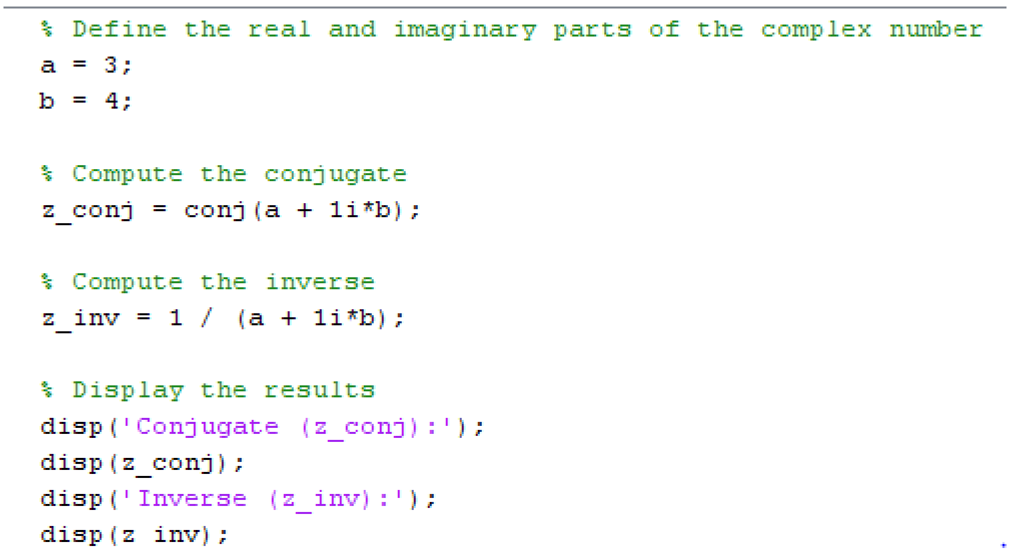
Conclusion:
The MATLAB function **zprint** successfully provides the real part, imaginary part, magnitude, phase in radians, and phase in degrees of a given complex number. It facilitates the numerical analysis of complex numbers, aiding in various signal processing tasks.

Task 2:
Compute the conjugate ź (i.e. z_conj [give variable name]) and the inverse 1/z (i.e. z_inv [give variable name]) for any complex number z. Display the results numerically with zprint.

Problem Statement - Task 02:
Compute the complex conjugate and the inverse for any given complex number z. Display the results numerically using the zprint function.

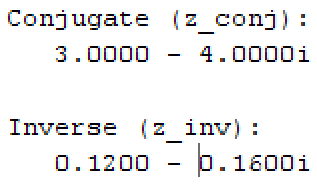
Algorithm:
1. Accept a complex number z as input.
2. Compute the complex conjugate using the formula.
3. Compute the inverse using the formula
4. Display the results numerically using the zprint function.

Code:



```
% Define the real and imaginary parts of the complex number
a = 3;
b = 4;

% Compute the conjugate
z_conj = conj(a + 1i*b);

% Compute the inverse
z_inv = 1 / (a + 1i*b);

% Display the results
disp('Conjugate (z_conj):');
disp(z_conj);
disp('Inverse (z_inv):');
disp(z inv);
```

Output:



```
Conjugate (z_conj):
   3.0000 - 4.0000i

Inverse (z_inv):
   0.1200 - 0.1600i
```

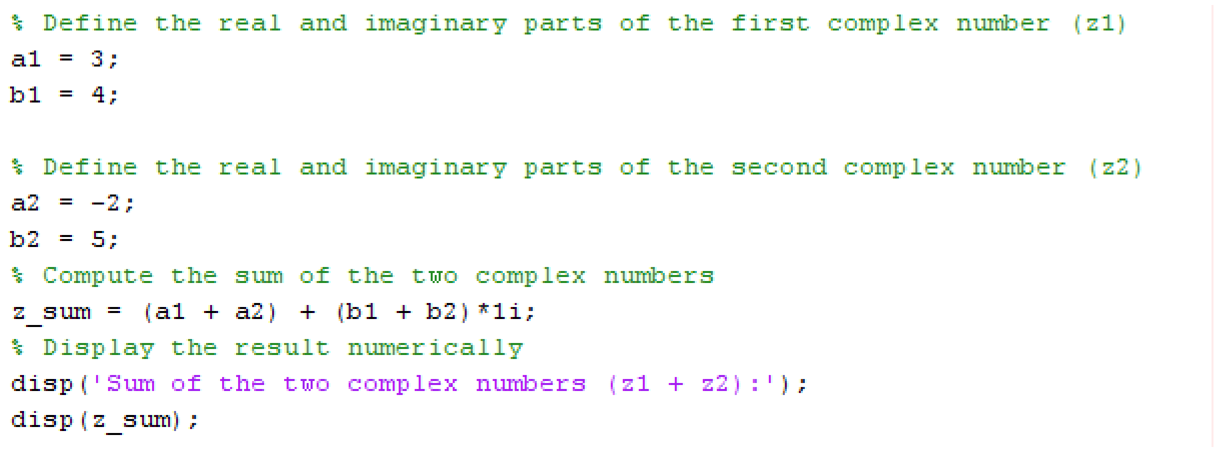
Conclusion:
The tasks of computing the complex conjugate and the inverse of a given complex number have been successfully achieved. The results are displayed numerically with the help of the zprint function, providing insights into the manipulation of complex numbers.

Task 3:
Take two complex numbers, compute z୧ + z୨ and display the results numerically using zprint.

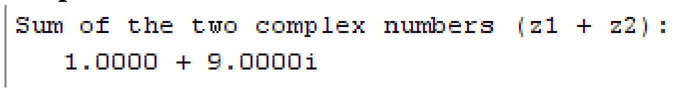
Problem Statement:
Take two complex numbers. Compute the sum and display the results numerically using the zprint function.

Algorithm:

- Accept two complex numbersas inputs.
- Compute the sum
- Display the results numerically using the zprint function.

Code:


```
% Define the real and imaginary parts of the first complex number (z1)
a1 = 3;
b1 = 4;

% Define the real and imaginary parts of the second complex number (z2)
a2 = -2;
b2 = 5;
% Compute the sum of the two complex numbers
z_sum = (a1 + a2) + (b1 + b2)*1i;
% Display the result numerically
disp('Sum of the two complex numbers (z1 + z2):');
disp(z_sum);
```

Output:


```
Sum of the two complex numbers (z1 + z2):
   1.0000 + 9.0000i
```

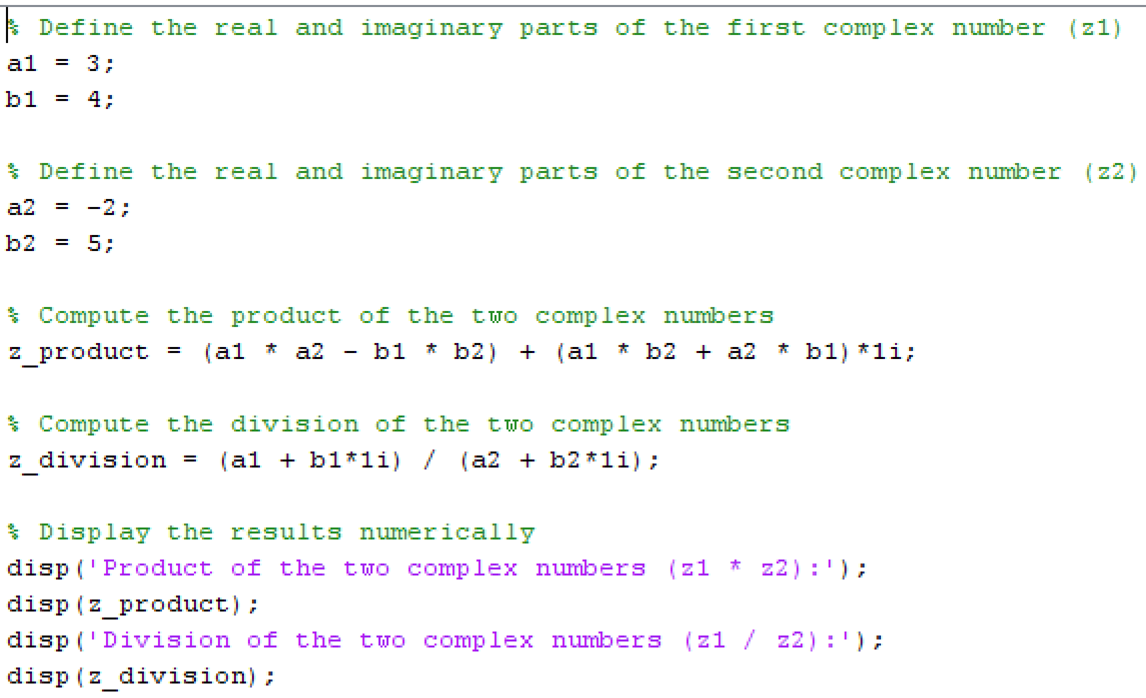
Conclusion:
The task of computing the sum of two complex numbers has been successfully achieved. The numerical results are displayed with the help of the zprint function, providing insights into the addition operation on complex numbers.

Task 04:
Take two complex numbers and compute z1*z2, z1/z2. Use zprint to display the results numerically.

Problem Statement:
Take two complex numbers and compute their product and the division.Use the zprint function to display the results numerically.

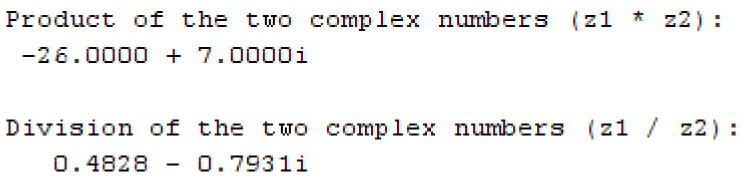
Algorithm:
1. Accept two complex numbers as inputs.
2. Compute their product z.
3. Compute their division
4. Display the results numerically using the zprint function.

Code:



```
% Define the real and imaginary parts of the first complex number (z1)
a1 = 3;
b1 = 4;

% Define the real and imaginary parts of the second complex number (z2)
a2 = -2;
b2 = 5;

% Compute the product of the two complex numbers
z_product = (a1 * a2 - b1 * b2) + (a1 * b2 + a2 * b1)*1i;

% Compute the division of the two complex numbers
z_division = (a1 + b1*1i) / (a2 + b2*1i);

% Display the results numerically
disp('Product of the two complex numbers (z1 * z2):');
disp(z_product);
disp('Division of the two complex numbers (z1 / z2):');
disp(z_division);
```

Output:



```
Product of the two complex numbers (z1 * z2):
 -26.0000 + 7.0000i

Division of the two complex numbers (z1 / z2):
   0.4828 - 0.7931i
```

Conclusion:
The task of computing the product and division of two complex numbers has been successfully achieved. The numerical results are displayed with the help of the zprint function, providing insights into the multiplication and division operations on complex numbers.

Task 05:

Determine the complex conjugate of the exponential signal given in above example and plot its real and imaginary parts.

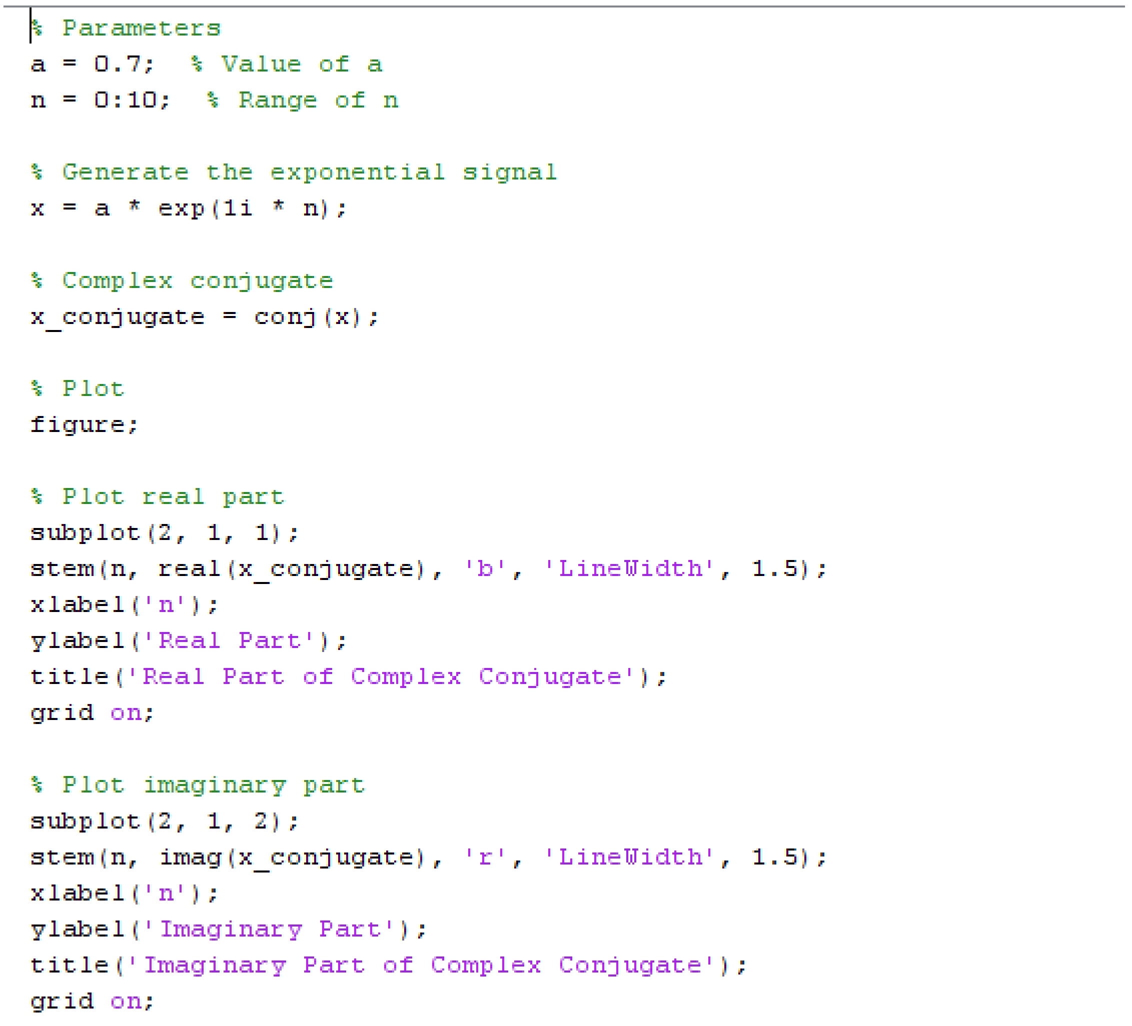
Problem Statement:

Determine the complex conjugate of an exponential signal given in the above example and plot its real and imaginary parts.

Algorithm:

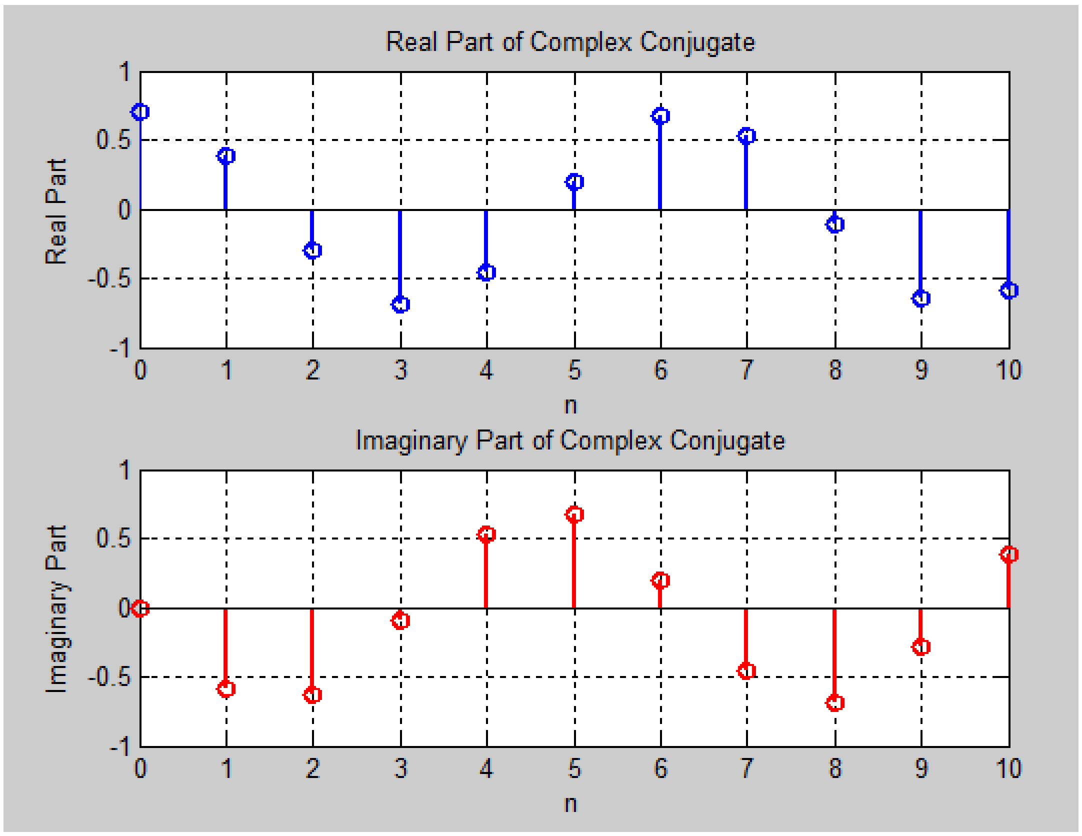
1. Determine the exponential signal.
2. Calculate its complex conjugate.
3. Plot the real and imaginary parts of the conjugate signal.

Code:



```
% Parameters
a = 0.7;  % Value of a
n = 0:10;  % Range of n

% Generate the exponential signal
x = a * exp(1i * n);

% Complex conjugate
x_conjugate = conj(x);

% Plot
figure;

% Plot real part
subplot(2, 1, 1);
stem(n, real(x_conjugate), 'b', 'LineWidth', 1.5);
xlabel('n');
ylabel('Real Part');
title('Real Part of Complex Conjugate');
grid on;

% Plot imaginary part
subplot(2, 1, 2);
stem(n, imag(x_conjugate), 'r', 'LineWidth', 1.5);
xlabel('n');
ylabel('Imaginary Part');
title('Imaginary Part of Complex Conjugate');
grid on;
```

Output:



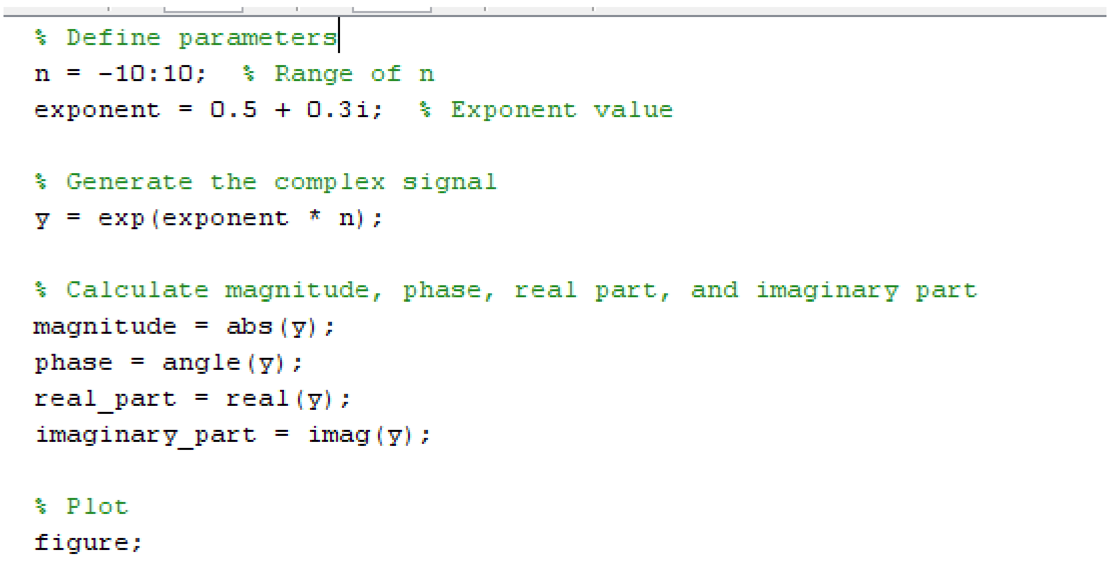
Conclusion:
The complex conjugate of the exponential signal has been successfully determined, and its real and imaginary parts have been plotted. This visualization aids in understanding the behavior of the signal and its conjugate.

Task 06:
Generate the complex valued signal for- 10 ≤ n ≤ 10 and plot its magnitude, phase, the real part, and the imaginary part in separate subplots.

Problem Statement:
Generate a complex-valued signal and plot its magnitude, phase, real part, and imaginary part in separate subplots.

Algorithm:
1. Generate the complex-valued signal.
2. Calculate its magnitude, phase, real part, and imaginary part.
3. Plot these components in separate subplots.

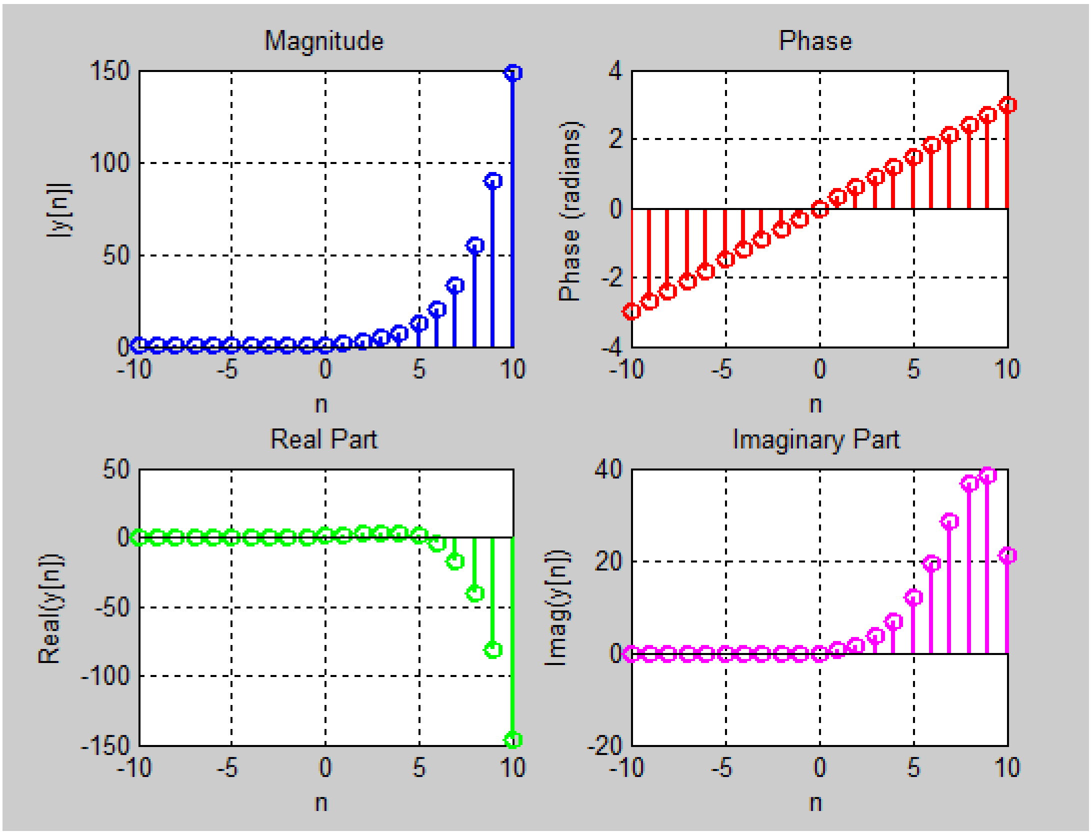
Code:



```
% Define parameters
n = -10:10;  % Range of n
exponent = 0.5 + 0.3i;  % Exponent value

% Generate the complex signal
y = exp(exponent * n);

% Calculate magnitude, phase, real part, and imaginary part
magnitude = abs(y);
phase = angle(y);
real_part = real(y);
imaginary_part = imag(y);

% Plot
figure;
```

```
% Magnitude subplot
subplot(2, 2, 1);
stem(n, magnitude, 'b', 'LineWidth', 1.5);
xlabel('n');
ylabel('|y[n]|');
title('Magnitude');
grid on;

% Phase subplot
subplot(2, 2, 2);
stem(n, phase, 'r', 'LineWidth', 1.5);
xlabel('n');
ylabel('Phase (radians)');
title('Phase');
grid on;

% Real part subplot
subplot(2, 2, 3);
stem(n, real_part, 'g', 'LineWidth', 1.5);
xlabel('n');
ylabel('Real(y[n])');
title('Real Part');
grid on;

% Imaginary part subplot
subplot(2, 2, 4);
stem(n, imaginary_part, 'm', 'LineWidth', 1.5);
xlabel('n');
ylabel('Imag(y[n])');
title('Imaginary Part');
grid on;
```

Output:



Conclusion:
The complex-valued signal has been successfully generated and analyzed. Its magnitude, phase, real part, and imaginary part have been plotted separately, providing a comprehensive understanding of the signal's characteristics.

Task 07:
Generate a real-exponential x = an for a = 0.7 and n ranging from 0-10. Find the discrete time as well as the continuous time versions of this signal. Plot the two signals on the same graph (by holding both the graphs).
Repeat the same program with value of a = 1.3.

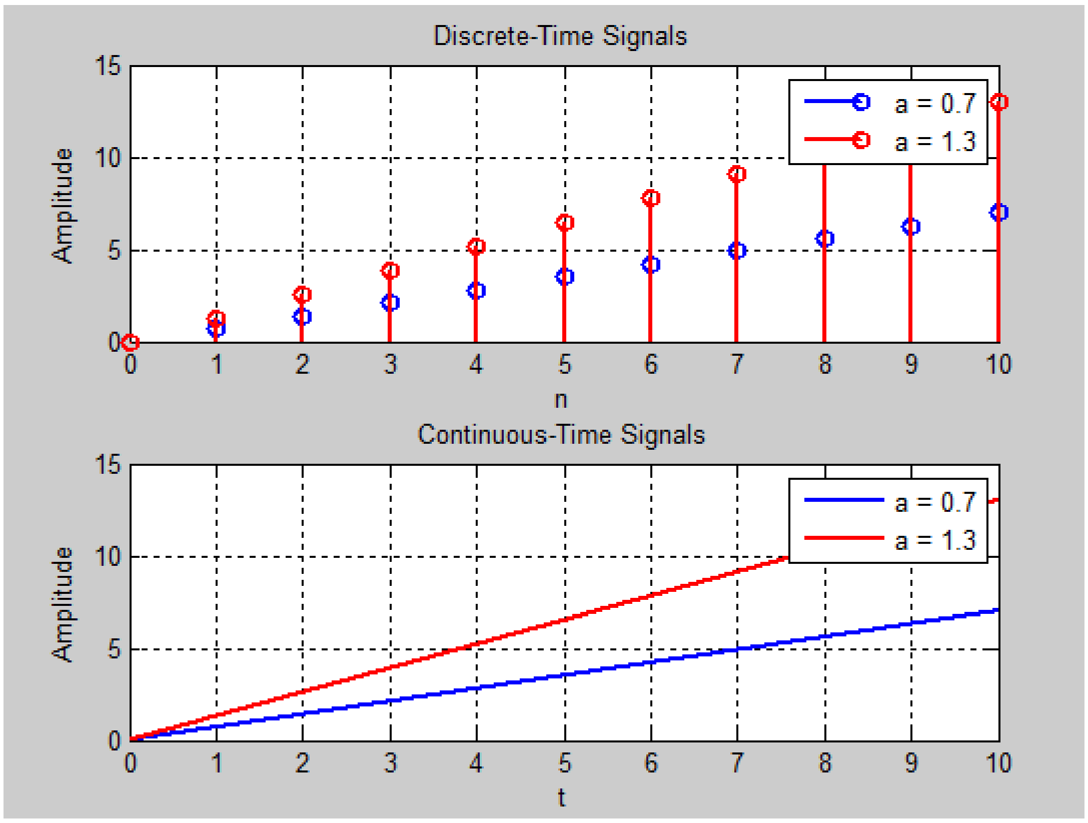
Problem Statement:
Generate a real-exponential signal for given values of *a* and *n*. Find its discrete-time and continuous-time versions and plot both signals on the same graph.

Algorithm:
1. Generate the real-exponential signal *x*[*n*] for given values of *a* and *n*.
2. Find its discrete-time and continuous-time versions.
3. Plot both signals on the same graph.

Code:

```
a_values = [0.7, 1.3];  % Different values of a
n = 0:10;  % Range of n
% Generate discrete-time signals
x_discrete_0_7 = a_values(1) * n;
x_discrete_1_3 = a_values(2) * n;
% Continuous-time versions
% For a = 0.7
t_continuous_0_7 = linspace(0, 10, 1000);
x_continuous_0_7 = a_values(1) * t_continuous_0_7;
% For a = 1.3
t_continuous_1_3 = linspace(0, 10, 1000);
x_continuous_1_3 = a_values(2) * t_continuous_1_3;
% Plot
figure;
% Plot discrete-time signals
subplot(2, 1, 1);
stem(n, x_discrete_0_7, 'b', 'LineWidth', 1.5);
hold on;
stem(n, x_discrete_1_3, 'r', 'LineWidth', 1.5);
xlabel('n');
ylabel('Amplitude');
title('Discrete-Time Signals');
legend('a = 0.7', 'a = 1.3');
grid on;
% Plot continuous-time signals
subplot(2, 1, 2);
plot(t_continuous_0_7, x_continuous_0_7, 'b', 'LineWidth', 1.5);
hold on;
plot(t_continuous_1_3, x_continuous_1_3, 'r', 'LineWidth', 1.5);
xlabel('t');
ylabel('Amplitude');
title('Continuous-Time Signals');
legend('a = 0.7', 'a = 1.3');
grid on;
```

Output:



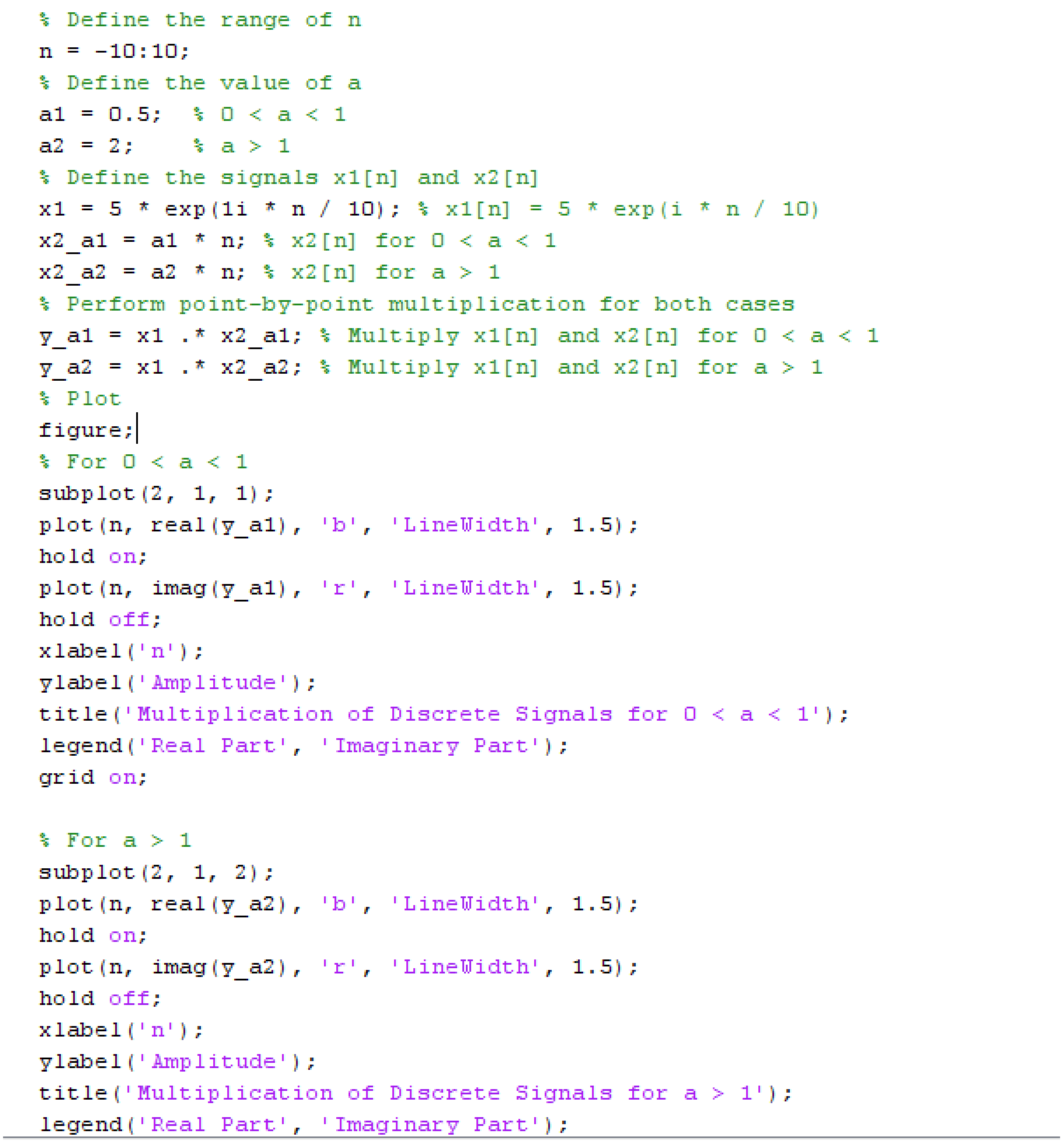
Conclusion:
The real-exponential signal has been successfully generated and plotted in both discrete-time and continuous-time domains. Plotting both signals on the same graph allows for comparison and visualization of their behavior.

Task 08:
Multiply the two discrete signals, use point-by-point multiplication of the two signals). Plot the real as well as the exponential parts for 0 < a < 1 and a > 1.

Problem Statement:
Multiply two discrete signals using point-by-point multiplication. Plot the real and imaginary parts for different ranges of *a*.

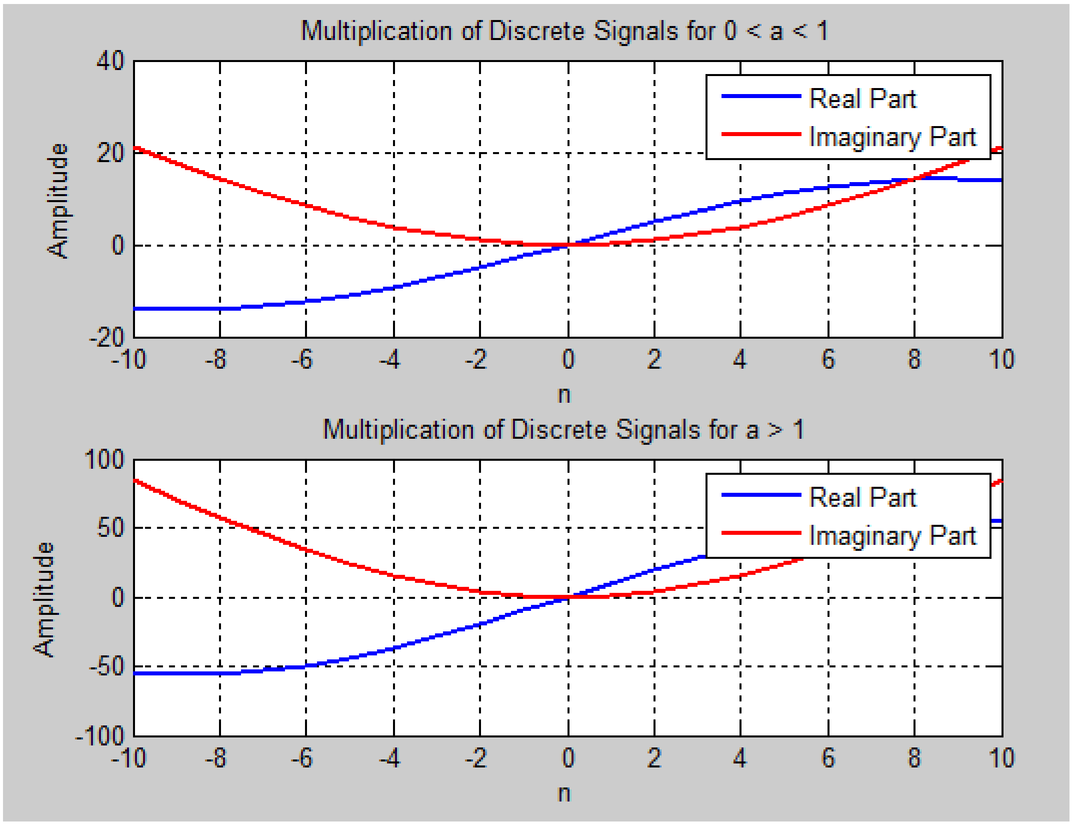
Algorithm - Task 08:
1. Multiply two discrete signals using point-by-point multiplication.
2. Plot the real and imaginary parts of the result for different ranges of *a*.

Code:



```matlab
% Define the range of n
n = -10:10;
% Define the value of a
a1 = 0.5;  % 0 < a < 1
a2 = 2;    % a > 1
% Define the signals x1[n] and x2[n]
x1 = 5 * exp(1i * n / 10); % x1[n] = 5 * exp(i * n / 10)
x2_a1 = a1 * n; % x2[n] for 0 < a < 1
x2_a2 = a2 * n; % x2[n] for a > 1
% Perform point-by-point multiplication for both cases
y_a1 = x1 .* x2_a1; % Multiply x1[n] and x2[n] for 0 < a < 1
y_a2 = x1 .* x2_a2; % Multiply x1[n] and x2[n] for a > 1
% Plot
figure;
% For 0 < a < 1
subplot(2, 1, 1);
plot(n, real(y_a1), 'b', 'LineWidth', 1.5);
hold on;
plot(n, imag(y_a1), 'r', 'LineWidth', 1.5);
hold off;
xlabel('n');
ylabel('Amplitude');
title('Multiplication of Discrete Signals for 0 < a < 1');
legend('Real Part', 'Imaginary Part');
grid on;

% For a > 1
subplot(2, 1, 2);
plot(n, real(y_a2), 'b', 'LineWidth', 1.5);
hold on;
plot(n, imag(y_a2), 'r', 'LineWidth', 1.5);
hold off;
xlabel('n');
ylabel('Amplitude');
title('Multiplication of Discrete Signals for a > 1');
legend('Real Part', 'Imaginary Part');
```

Output:

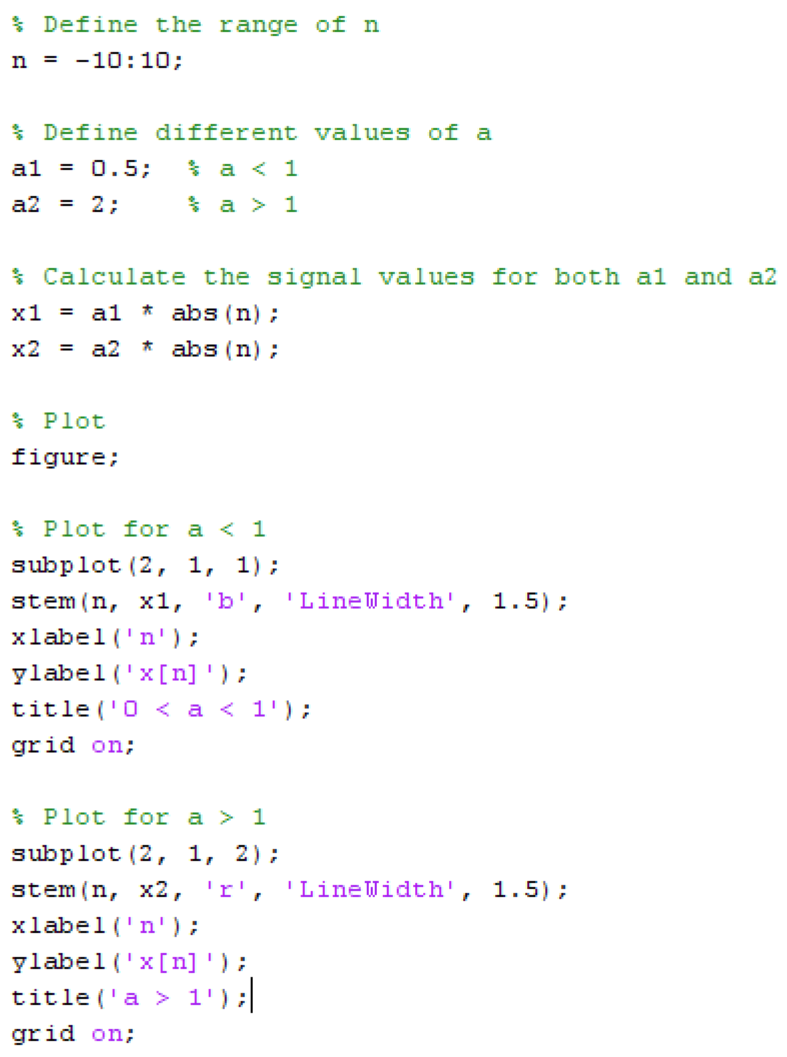


Conclusion:
The point-by-point multiplication of two discrete signals has been successfully performed, and their real and imaginary parts have been plotted for different ranges of *a*. This visualization helps in understanding the effects of multiplication on the signals.

Task 09:
Plot the discrete signal for n ranging from -10 to 10. Draw two subplots for 0 < a <1 and a > 1.

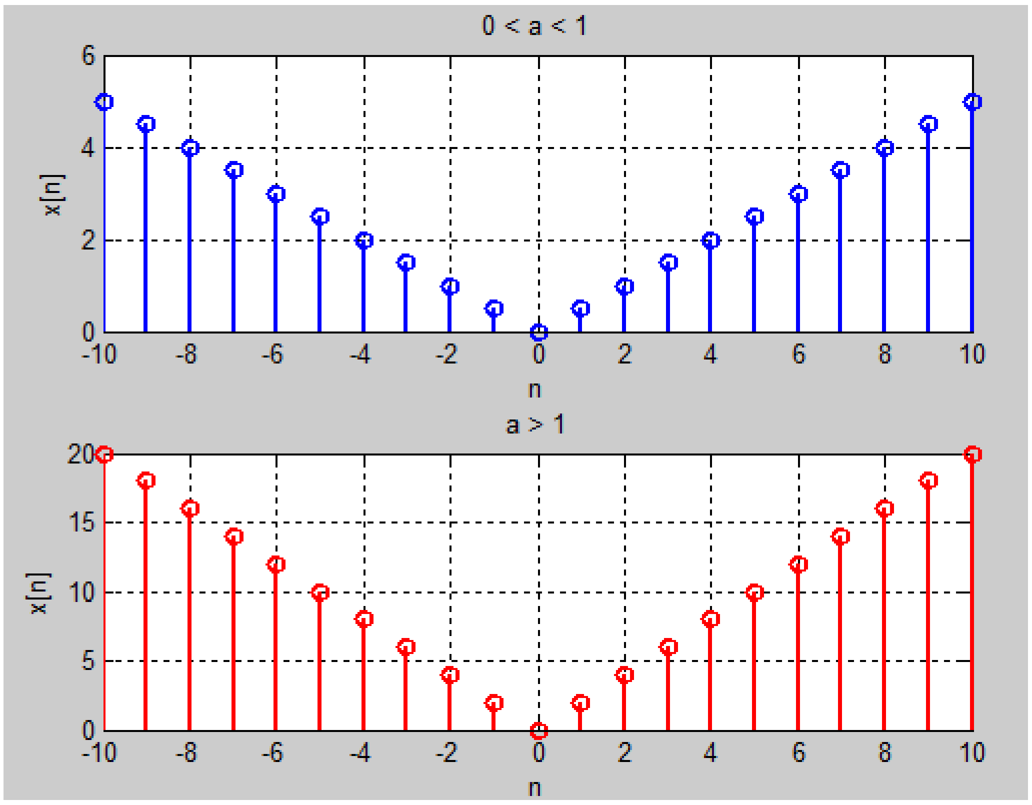
Problem Statement:
Plot the discrete signal for different ranges of *a*.
Algorithm:
1. Plot the discrete signal for different ranges of *a*.
2. Draw subplots.

Code:



```
% Define the range of n
n = -10:10;

% Define different values of a
a1 = 0.5;  % a < 1
a2 = 2;    % a > 1

% Calculate the signal values for both a1 and a2
x1 = a1 * abs(n);
x2 = a2 * abs(n);

% Plot
figure;

% Plot for a < 1
subplot(2, 1, 1);
stem(n, x1, 'b', 'LineWidth', 1.5);
xlabel('n');
ylabel('x[n]');
title('0 < a < 1');
grid on;

% Plot for a > 1
subplot(2, 1, 2);
stem(n, x2, 'r', 'LineWidth', 1.5);
xlabel('n');
ylabel('x[n]');
title('a > 1');
grid on;
```

Output:



Conclusion:
The discrete signal has been successfully plotted for different ranges of *a*. Subplots have been drawn, providing a clear visualization of the signal's behavior in different scenarios.

Task 10:

- **Generate the signal x(t) for A=3, π =- 0.4, and ω = 2π(1250). Take a range for t that will cover 2 or 3 periods.**
- **Plot the real part versus t and the imaginary part versus t. Use subplot(2,1,i) to put both plots in the same window.**
- **Verify that the real and imaginary parts are sinusoids and that they have the correct frequency, phase, and amplitude.**

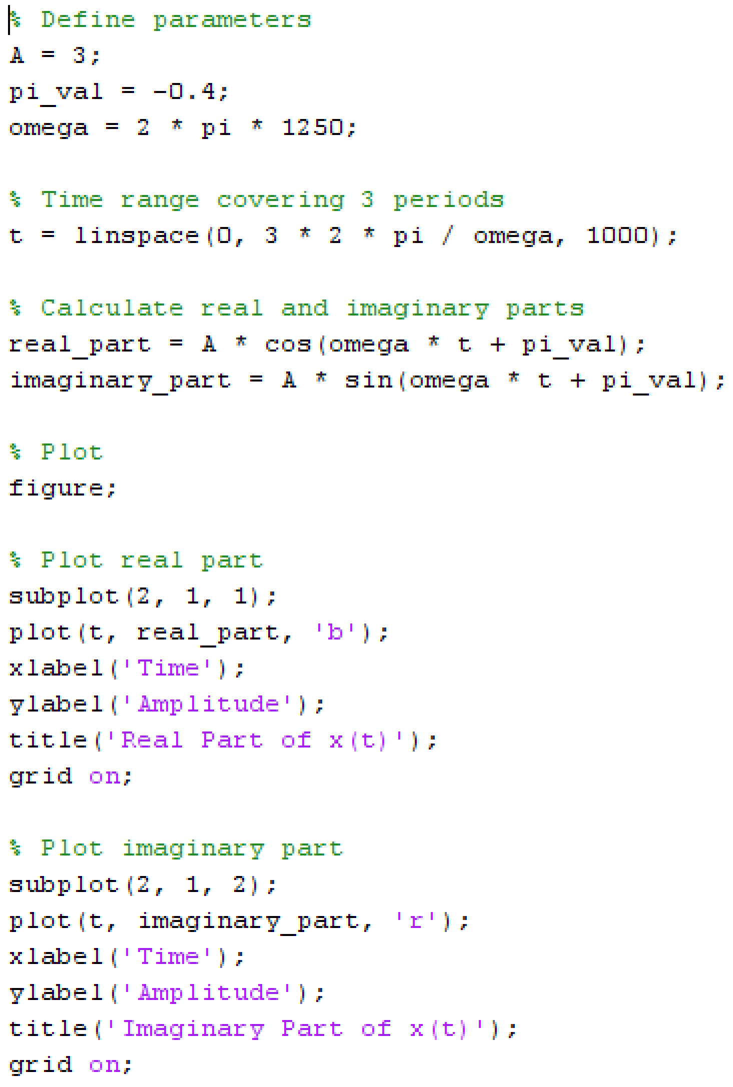
Problem Statement:
Generate the signal $x(t)=Aej(\omega t+\phi)$ for given parameters A, π, and ω. Take a range for t that will cover 2 or 3 periods. Plot the real part versus t and the imaginary part versus t. Use subplot(2,1,i) to put both plots in the same window. Verify that the real and imaginary parts are sinusoids and that they have the correct frequency, phase, and amplitude.

Algorithm:

1. Define the parameters A, π, and ω.
2. Define the time range t that will cover 2 or 3 periods of the signal.
3. Calculate the real and imaginary parts of the signal $x(t)=Aej(\omega t+\phi)$.
4. Plot the real part versus t and the imaginary part versus t using subplot(2,1,i) to put both plots in the same window.

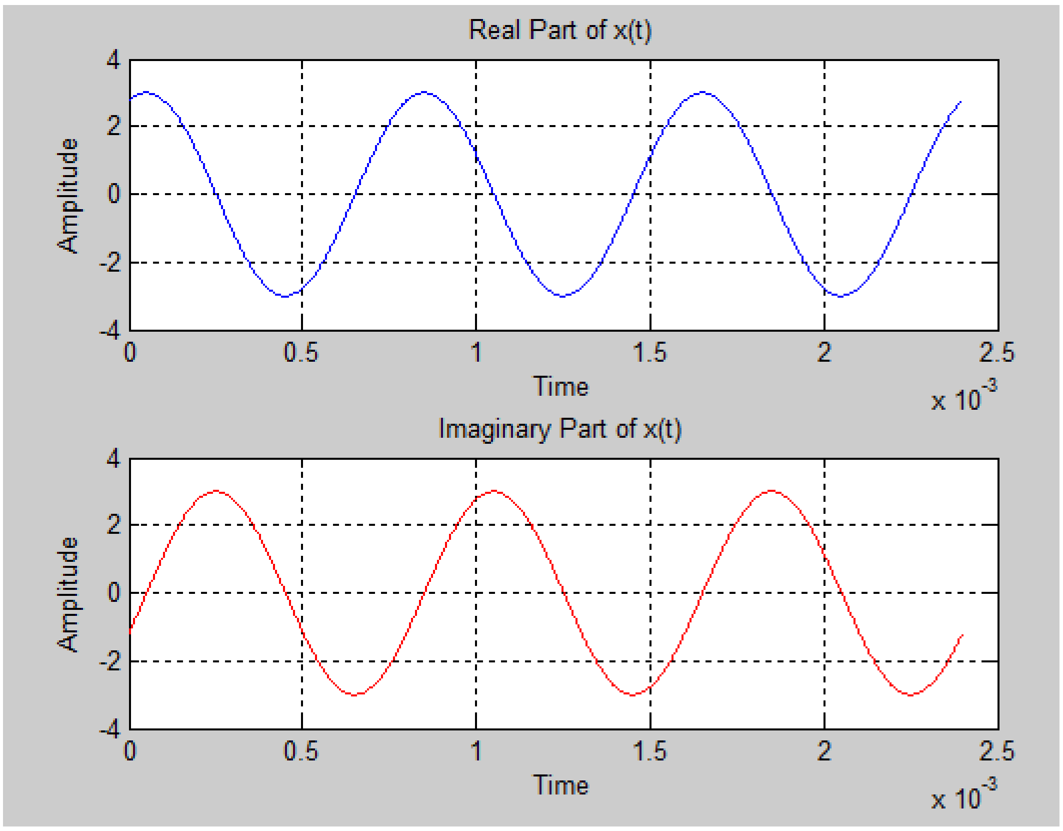
5. Verify that the real and imaginary parts are sinusoids with the correct frequency, phase, and amplitude.

Code:



```
% Define parameters
A = 3;
pi_val = -0.4;
omega = 2 * pi * 1250;

% Time range covering 3 periods
t = linspace(0, 3 * 2 * pi / omega, 1000);

% Calculate real and imaginary parts
real_part = A * cos(omega * t + pi_val);
imaginary_part = A * sin(omega * t + pi_val);

% Plot
figure;

% Plot real part
subplot(2, 1, 1);
plot(t, real_part, 'b');
xlabel('Time');
ylabel('Amplitude');
title('Real Part of x(t)');
grid on;

% Plot imaginary part
subplot(2, 1, 2);
plot(t, imaginary_part, 'r');
xlabel('Time');
ylabel('Amplitude');
title('Imaginary Part of x(t)');
grid on;
```

Output:



Conclusion:
The signal $x(t)=Aej(\omega t+\phi)$ has been successfully generated and plotted. The real and imaginary parts have been plotted versus time t in the same window using subplots, confirming that they are sinusoidal with the correct frequency, phase, and amplitude. This analysis verifies the characteristics of the signal and its components.