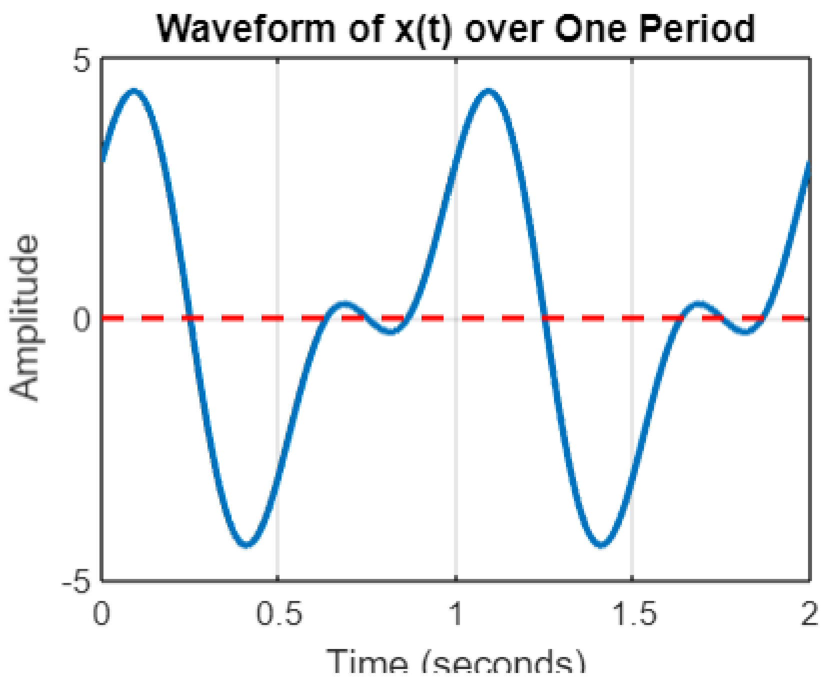
Consider a continuous - time signal x(t)=3cos(2πt) + 2sin(4πt)

1)Sketch the waveform of x(t) over one period

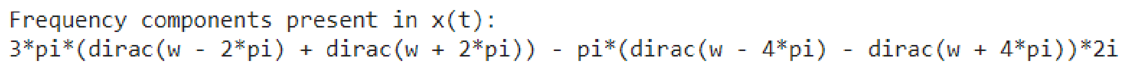
```
% Define the symbolic variable t
syms t

% Define the signal x(t)
xt = 3*cos(2*pi*t) + 2*sin(4*pi*t);

% Set the time values for one period
t_values = linspace(0, 2, 1000);

% Evaluate the signal x(t) at the sampled time points
x_values = double(subs(xt, t, t_values));

% Plot the waveform
figure;
plot(t_values, x_values, 'LineWidth', 2);
title('Waveform of x(t) over One Period');
xlabel('Time (seconds)');
ylabel('Amplitude');
grid on;

% Highlight zero crossings
hold on;
plot([0, 2], [0, 0], '--', 'Color', 'red', 'LineWidth', 1.5);
hold off;
```



The code starts by defining a symbolic variable t using the syms function.
The signal x(t)=3cos(2πt) + 2sin(4πt) is then defined using symbolic variable ‘t’.
The ‘linspace’ function is used to generate 1000 evenly spaced time values between 0 and 2 seconds. This represent one period of the signal.
The ‘subs’ function substitutes the time values t_values into the symbolic expression xt to obtain the corresponding values of the signal at those time points.
The ‘double’ function is used to convert the symbolic results to numerical values.
The code creates a new figure and plots the waveform using the ‘plot’ function.
The “ 'LineWidth', 2” option makes the plot lines thicker for better visibility.
Axis labels and a title are added for clarity.
‘grid on’ adds a grid to the pilot.
The 'hold on' command allows adding multiple plots to the same figure without clearing the existing content.
The code then plots a dashed red line representing the zero axis (y = 0) to highlight the zero crossings of the waveform.
‘hold off’ ends the holds state.

2)Determine the frequency components present in x(t)

```
% Define the symbolic variable t
syms t

% Define the continuous-time signal x(t)
x_t = 3*cos(2*pi*t) + 2*sin(4*pi*t);

% Compute the Fourier transform of x(t)
X_f = fourier(x_t);

% Display the frequency components
disp('Frequency components present in x(t):');
disp(X_f);
```

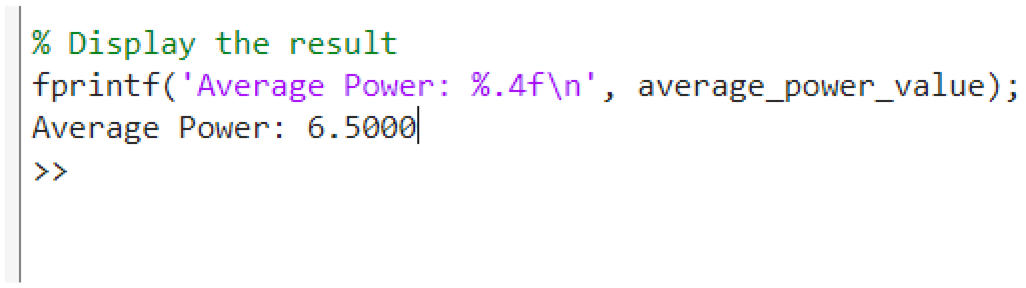


```
Frequency components present in x(t):
3*pi*(dirac(w - 2*pi) + dirac(w + 2*pi)) - pi*(dirac(w - 4*pi) - dirac(w + 4*pi))*2i
```

This line creates a symbolic variable t using the syms function. Symbolic variables are used for symbolic mathematics in MATLAB.
This line defines the continuous-time signal x(t)=3cos(2πt) + 2sin(4πt) using the symbolic variable t.
The ‘fourier’ function is used to compute the Fourier transform of the signal x(t) with respect to the variable t.
The result Xf is a symbolic expression representing the frequency domain representation of the signal.
This section displays a message and then prints the symbolic expression Xf, which represents the frequency components present in the signal x(t).

3)Compute the average power of x(t) over one period.

```
% Define the symbolic variable t
syms t

% Define the signal x(t)
xt = 3*cos(2*pi*t) + 2*sin(4*pi*t);

% Define the period T
T = 2;

% Define the power expression
power_expression = xt^2;

% Integrate the power expression over one period
average_power = (1/T) * int(power_expression, 0, T);

% Evaluate the numerical result
average_power_value = double(average_power);

% Display the result
fprintf('Average Power: %.4f\n', average_power_value);
```



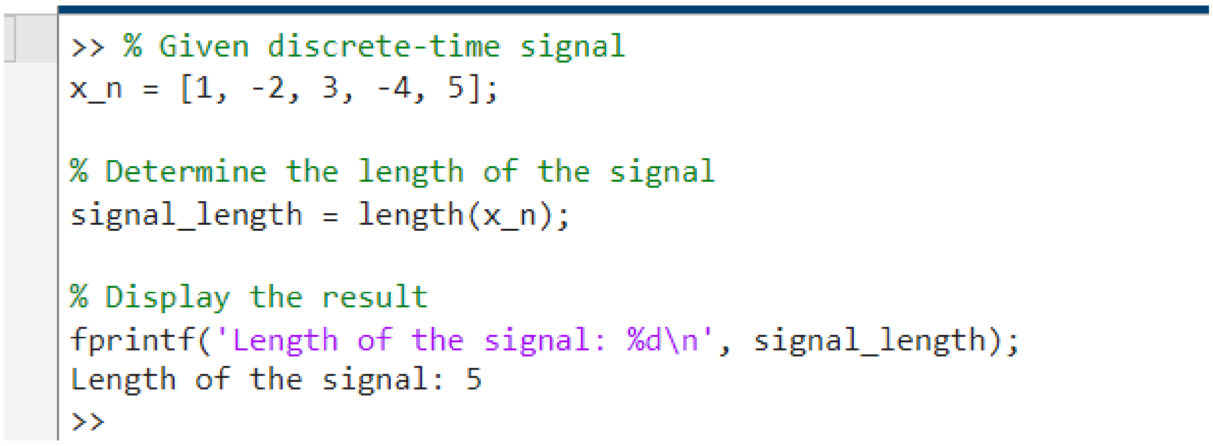
```
% Display the result
fprintf('Average Power: %.4f\n', average_power_value);
Average Power: 6.5000
>>
```

This line creates a symbolic variable t using the ‘syms’ function for symbolic calculations.
The continuous-time signal x(t) is defined using the symbolic variable t and a combination of 'cosine' and 'sine' functions.
The variable T is set to the period of the signal. In this case, the period is set to 22 seconds.
The power expression is defined as the square of the signal x(t).
The power expression is integrated over one period (from 0 to T) to calculate the average power.
The ‘double’ function is used to convert the symbolic result to a numerical value.
The average power value is displayed with four decimal places.

Given the discrete - time signal x[n] ={1,-2,3,-4,5}:

1)Determine the length of the signal.

```
% Given discrete-time signal
x_n = [1, -2, 3, -4, 5];

% Determine the length of the signal
signal_length = length(x_n);

% Display the result
fprintf('Length of the signal: %d\n', signal_length);
```

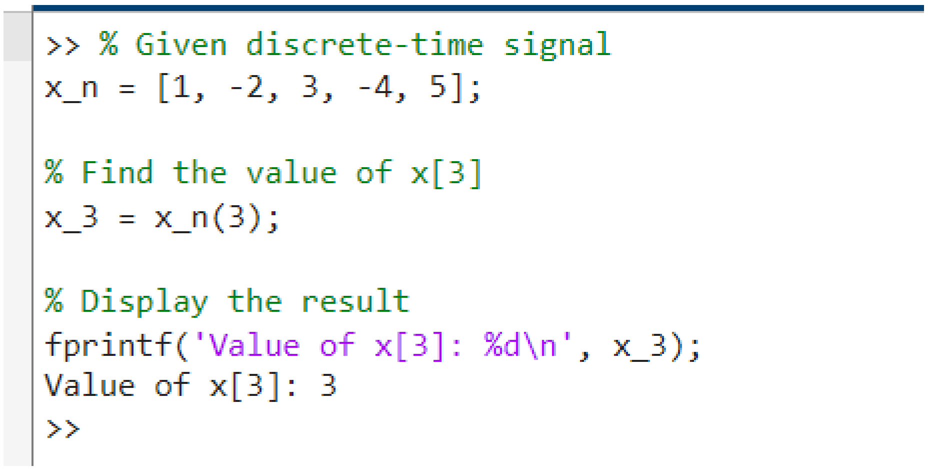


```
>> % Given discrete-time signal
x_n = [1, -2, 3, -4, 5];

% Determine the length of the signal
signal_length = length(x_n);

% Display the result
fprintf('Length of the signal: %d\n', signal_length);
Length of the signal: 5
>>
```

In this script, 'x_n' is the given discrete-time signal, and the 'length' function is used to find the number of elements in the signal, which corresponds to its length. The result is then displayed using 'fprintf'.

2)Find the value of x[3].

```
% Given discrete-time signal
x_n = [1, -2, 3, -4, 5];

% Find the value of x[3]
x_3 = x_n(3);

% Display the result
fprintf('Value of x[3]: %d\n', x_3);
```

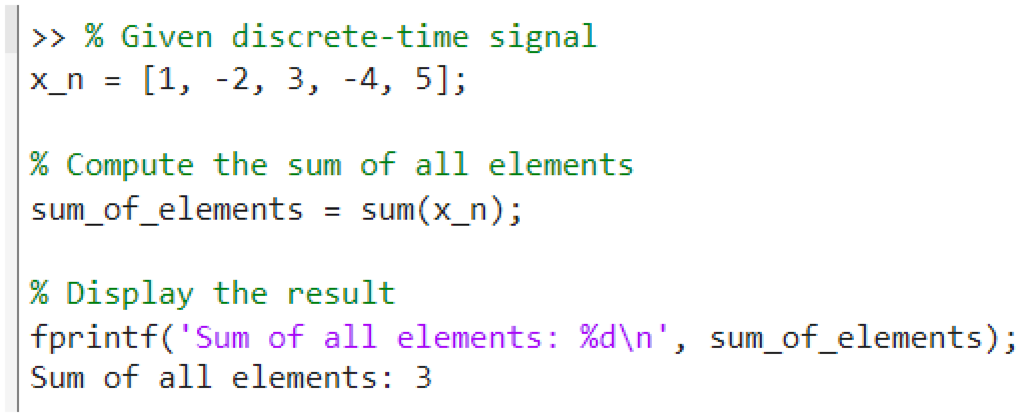


```
>> % Given discrete-time signal
x_n = [1, -2, 3, -4, 5];

% Find the value of x[3]
x_3 = x_n(3);

% Display the result
fprintf('Value of x[3]: %d\n', x_3);
Value of x[3]: 3
>>
```

In MATLAB, indexing starts from 1, so 'x_n(3)' accesses the element at the third position in the array, which corresponds to [3]x[3]. The result is then displayed using 'fprintf'.

3) Compute the sum of all elements in the signal.

```
% Given discrete-time signal
x_n = [1, -2, 3, -4, 5];

% Compute the sum of all elements
sum_of_elements = sum(x_n);

% Display the result
fprintf('Sum of all elements: %d\n', sum_of_elements);
```

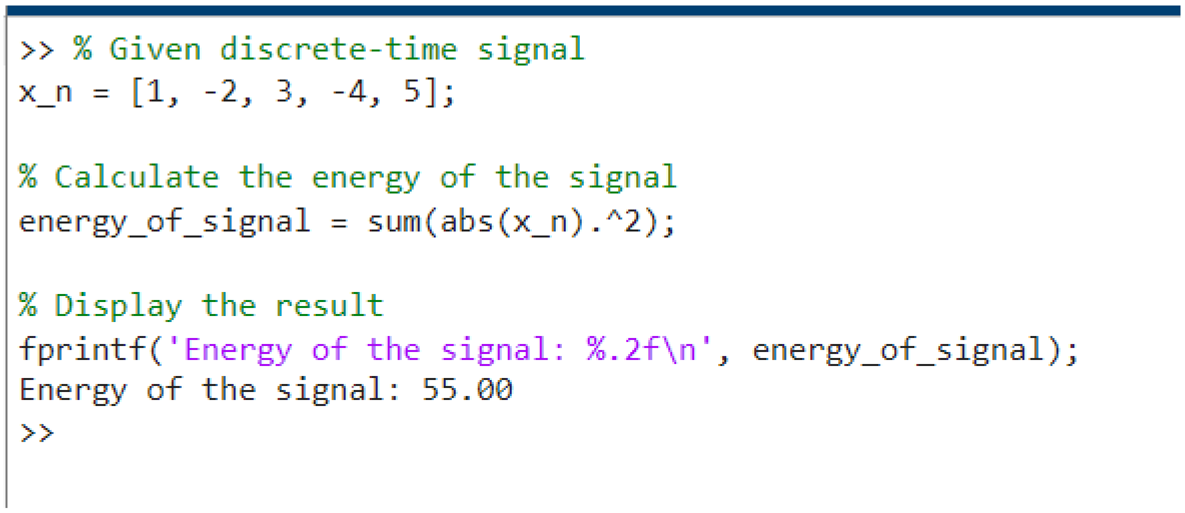


```
>> % Given discrete-time signal
x_n = [1, -2, 3, -4, 5];

% Compute the sum of all elements
sum_of_elements = sum(x_n);

% Display the result
fprintf('Sum of all elements: %d\n', sum_of_elements);
Sum of all elements: 3
```

The 'sum' function adds up all the elements in the array 'x_n', and the result is displayed using 'fprintf'.

4)Calculate the energy of the signal.

```
% Given discrete-time signal
x_n = [1, -2, 3, -4, 5];

% Calculate the energy of the signal
energy_of_signal = sum(abs(x_n).^2);

% Display the result
fprintf('Energy of the signal: %.2f\n', energy_of_signal);
```



```
>> % Given discrete-time signal
x_n = [1, -2, 3, -4, 5];

% Calculate the energy of the signal
energy_of_signal = sum(abs(x_n).^2);

% Display the result
fprintf('Energy of the signal: %.2f\n', energy_of_signal);
Energy of the signal: 55.00
>>
```

In this code, 'abs(x_n).^2' calculates the squared magnitude of each element in the signal, and 'sum' adds up all these squared values. The result is then displayed using 'fprintf'.