Bengisu Münteha Koç

21801974

EEE 321 – 02

14.11.2024

EEE 321 Lab Work 4

Introduction

This report examines the Fourier series expansion of rectified signals to analyze their properties in the time and frequency domains. The Fourier series allows any periodic signal to be represented as a sum of sinusoidal components, each at integer multiples of a fundamental frequency, with specific amplitudes and phases. This decomposition technique is widely used in signal processing, communications, and digital systems to reveal the harmonic structure of signals and provide insights into their frequency content.

The assignment focuses on three types of periodic signals, each with distinct rectification characteristics. The first signal is a rectangular waveform defined over specific intervals within an 18-second period, creating a square-like shape with alternating high and low states. The second signal is a full-wave rectified cosine wave, where all negative values are reflected to positive, resulting in a fully non-negative cosine waveform with double the original frequency. The third signal is a half-wave rectified cosine, which retains only the positive half of each cosine cycle, producing a signal with a periodic "on-off" pattern within each period.

The objectives of the assignment include discretizing these signals with a sampling period of T_s=0.1 seconds, deriving their Fourier series expansions analytically, and examining their spectra. MATLAB was used to synthesize each signal with a varying number of harmonics, N, to evaluate the effect of including additional harmonics on the accuracy of reconstruction. The analysis also explores the Gibbs phenomenon, which is characterized by persistent oscillatory overshoot near discontinuities in signals reconstructed with a finite number of Fourier terms, highlighting a limitation of Fourier series approximations for signals with abrupt transitions.

By analyzing these rectified signals and their Fourier series representations, this report provides insights into the frequency-domain characteristics of rectified waveforms and illustrates the role of harmonic content in signal reconstruction, emphasizing essential Fourier series concepts in signal processing.

Background

The Fourier series is a mathematical tool that allows any periodic signal to be represented as a sum of simpler sinusoidal components: sines and cosines. This decomposition is valuable in signal processing because it reveals the frequency components of a signal, allowing for a detailed analysis of its harmonic content.

For a periodic signal y(t) with period T, the Fourier series expansion expresses y(t) as a sum of sinusoidal terms with frequencies that are integer multiples of a fundamental frequency $\omega_0 = \frac{2\pi}{T}$. The general form of the Fourier series in terms of sines and cosines is given by

$$y(t) = a_0 + \sum_{k=1}^{\infty} (a_k \cos(k\omega_0 t) + b_k \sin(k\omega_0 t))$$

where a_0 represents the DC component or the average value of the signal over one period, and a_k and b_k are the Fourier coefficients for the cosine and sine terms at the kth harmonic of the fundamental frequency ω_0. The terms $\cos(k\omega_0 t)$ and $\sin(k\omega_0 t)$ represent the oscillating components at frequencies that are integer multiples of the fundamental frequency. Each of these components contributes to the signal at a specific frequency, with the Fourier coefficients a_k and b_k quantifying the amplitude of each harmonic.

This standard form of the Fourier series expansion is derived from a more general representation, known as the complex form of the Fourier series. The complex form uses complex exponentials rather than separate sine and cosine terms and provides a unified approach that can represent both real and complex signals. The complex form of the Fourier series expresses a periodic signal y(t) as

$$y(t) = \sum_{k=-\infty}^{\infty} C_k e^{\mathrm{j}k\omega_0 t}$$

where $\omega_0 = \frac{2\pi}{T}$is the fundamental frequency, C_k are the complex Fourier coefficients, and $e^{\mathrm{j}k\omega_0 t}$ represents the complex exponential functions. The Fourier coefficients C_k are calculated by

$$C_k = \frac{1}{T}\int_0^T y(t) e^{-\mathrm{j}k\omega_0 t}\, dt$$

and contain both amplitude and phase information for each harmonic. The complex coefficients C_k are generally complex numbers, meaning they include both a real and an imaginary component, allowing the Fourier series to capture both amplitude and phase shifts in a compact form. For real signals, C_k and C_{-k} are complex conjugates, ensuring that their sum produces a real-valued signal y(t).

Using Euler's formula, $e^{\mathrm{j}k\omega_0 t} = \cos(k\omega_0 t) + \mathrm{j}\sin(k\omega_0 t)$ it is possible to split each complex exponential term into separate cosine and sine terms. The coefficients a_k and b_k in the real form are related to the complex coefficients C_k by $a_k = 2\mathrm{Re}(C_k)$ and $b_k = -2\mathrm{Im}(C_k)$ for positive k, while the DC

component a_0 corresponds to C_0. This transformation from the complex form to the real form results in the expression

$$y(t) = a_0 + \sum_{k=1}^{\infty} (a_k \cos(k\omega_0 t) + b_k \sin(k\omega_0 t))$$

which is particularly useful for analyzing real-valued signals in terms of their separate cosine and sine contributions.

The Fourier coefficients a_0 a_k, and b_k are calculated as follows. The DC component a_0 is given by

$$a_0 = \frac{1}{T} \int_0^T y(t)\ dt$$

which represents the average value of the signal over one period. The cosine coefficients a_k, which capture the amplitude of the cosine components, are given by

$$a_k = \frac{2}{T} \int_0^T y(t) \cos(k\omega_0 t)\ dt$$

and the sine coefficients b_k, which capture the amplitude of the sine components, are given by

$$b_k = \frac{2}{T} \int_0^T y(t) \sin(k\omega_0 t)\ dt$$

For even signals, the sine components b_k are zero, as an even function times an odd function integrates to zero over a symmetric interval. Similarly, for odd signals, the cosine components a_k are zero.

As more terms are included in the Fourier series (increasing the number of harmonics N), the approximation of the signal improves. However, for signals with sharp transitions or discontinuities, the Fourier series approximation exhibits the Gibbs phenomenon, where oscillatory overshoot persists near these discontinuities. This phenomenon limits the precision of Fourier series at points of discontinuity, even as more terms are added. By analyzing signals through their Fourier series, a deeper understanding of both the time-domain shape and frequency-domain characteristics of periodic signals can be achieved.

In this assignment, the Fourier series is applied to analyze rectified signals, including a full-wave rectified cosine, where all negative values are reflected to positive values, effectively doubling the frequency, and a half-wave rectified cosine, which retains only the positive half of each cosine cycle, creating a distinct "on-off" pattern within each period. These rectified signals exhibit unique harmonic structures that can be analyzed using Fourier series, providing insight into how rectification and symmetry affect the frequency content of periodic signals.

Analysis and Solutions

1.

a) First given function is a rectangular waveform which is

$$y_a(t) = \begin{cases} 0, & t \in [0, 7) \\ 8, & t \in [7, 10) \\ 0, & t \in [10, 18) \end{cases}$$

With a period of 18s.

$y_a(t)$ is discretized by using given sampling period of $T_s = 1/10$ s. with the following code block, Code Listing 1, the graph of the rectangular waveform is plotted on MATLAB as shown in Figure 1.

Code Listing 1: Rectangular Waveform Plot

```
% Set sampling period and discrete time range
Ts = 0.1;
n = -40:319;
t = n * Ts;


% Initialize y[n] and set the period of y_a(t)
y = zeros(size(n));
T = 18;


% Calculate y[n] based on the periodic definition of y_a(t)
for i = 1:length(t)
    t_mod = mod(t(i), T);
    if t_mod >= 7 && t_mod < 10
        y(i) = 8;
    else
        y(i) = 0;
    end
end


% Define finer time vector for continuous-style plotting
t_fine = linspace(min(t), max(t), 1000);


% Interpolate y[n] onto the fine time vector
y_continuous = interp1(t, y, t_fine, 'previous');


% Plot the interpolated continuous-style signal
figure;
plot(t_fine, y_continuous, 'LineWidth', 1.5, 'Color', [0, 0.5, 0.8]);
xlabel('Time (s)', 'FontSize', 12);
ylabel('Amplitude y_a(t)', 'FontSize', 12);
```

```
title('Continuous-Time Style Signal y_a(t) with Ts = 0.1 s', 'FontSize', 14);
xlim([min(t_fine), max(t_fine)]);
ylim([-1, 9]);
xticks(-4:2:32);
yticks(0:1:8);
grid on;
set(gca, 'GridLineStyle', '--', 'FontSize', 10);
ax = gca;
ax.Box = 'off';
hold off;
```

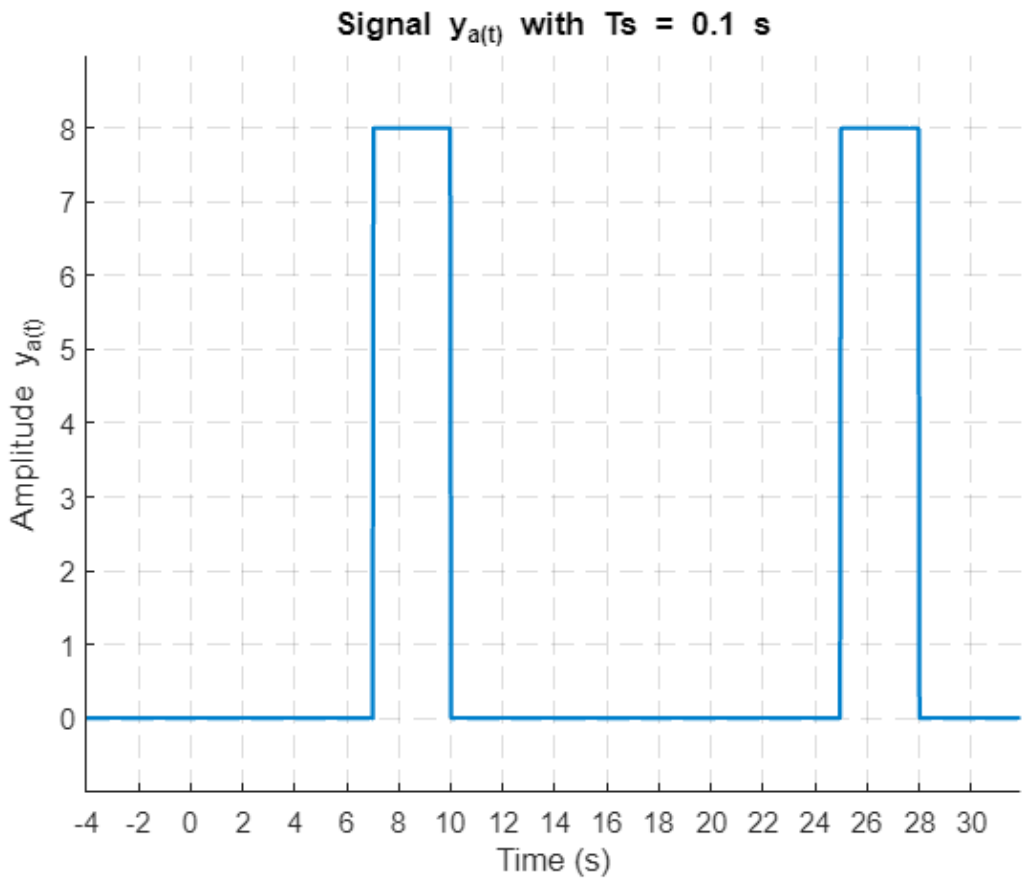

Figure 1: Plot of Rectangular Waveform

b) The Fourier series expansion of periodic waveform is calculated by using the approach mentioned in Background.

The fundamental frequency of the waveform is

$$\omega_0 = \frac{2\pi}{T} = \frac{2\pi}{18} = \frac{\pi}{9} \text{ rad/s}$$

DC component a_0 is found as

$$a_0 = \frac{1}{T}\int_0^T y_a(t)\ dt = \frac{1}{18}\int_7^{10} 8\ dt = \frac{1}{18}\cdot 8 \cdot 3 = \frac{4}{3}$$

Fourier coefficients a_k are found as

$$a_k = \frac{1}{18}\int_0^{18} y_a(t) e^{-j\frac{\pi}{9}kt}\ dt$$

$$a_k = \frac{-4j}{\pi k}\left(e^{-j\frac{10\pi}{9}k} - e^{-j\frac{7\pi}{9}k}\right)$$

c) The spectrum of the periodic waveform is plotted with complex amplitude interpretation. Hence the spectrum consists of two separate plots. One is for magnitude spectrum while the other is for phase spectrum.

The magnitude spectrum gives a clear view of the relative strengths of each frequency component. The phase spectrum shows the phase shift of each component in degrees, making it easier to interpret how each harmonic contributes to the waveform.

The code of the plots is given in Code Listing 2. The spectrum of the rectangular waveform is shown in Figure 2.

Code Listing 2: Spectrum of the Periodic Rectangular Waveform

```
% Parameters
T = 18;                          % Period of the signal
omega0 = pi / 9;                 % Fundamental frequency
k_values = -20:20;               % Range of k values for harmonics


% Initialization
frequencies = k_values * omega0;
a_k_magnitude = zeros(size(k_values));
a_k_phase = zeros(size(k_values));


% Calculation of the Fourier series coefficients a_k for each k
for i = 1:length(k_values)
    k = k_values(i);
    if k == 0
        % DC component
        a_k = 4 / 3;
    else
        % Non-zero Fourier coefficients (a_k)
        a_k = (-4j / (pi * k)) * (exp(-1j * 10 * omega0 * k) - exp(-1j * 7 *
omega0 * k));
    end

    a_k_magnitude(i) = abs(a_k);
```

```
    a_k_phase(i) = angle(a_k);
end


% Define finer frequency vector for smooth continuous-style plotting
freq_fine = linspace(min(frequencies), max(frequencies), 1000);


% Interpolate magnitude and phase on the finer frequency vector
magnitude_continuous = interp1(frequencies, a_k_magnitude, freq_fine,
'spline');
phase_continuous = interp1(frequencies, rad2deg(a_k_phase), freq_fine,
'spline');


% Magnitude Plot
figure;
subplot(2,1,1);
plot(freq_fine, magnitude_continuous, 'LineWidth', 1.5, 'Color', [0.8, 0.1,
0.3]);
xlabel('Frequency (rad/s)', 'FontSize', 12);
ylabel('|a_k|', 'FontSize', 12);
title('Magnitude Spectrum of y_a(t)', 'FontSize', 14);
xlim([-7, 7]);
ylim([0, max(a_k_magnitude) + 0.1]);
xticks(-3:0.5:3);
yticks(0:0.2:max(a_k_magnitude));
grid on;
set(gca, 'GridLineStyle', '--', 'FontSize', 10);
ax = gca;
ax.Box = 'off';


% Phase Plot
subplot(2,1,2);
plot(freq_fine, phase_continuous, 'LineWidth', 1.5, 'Color', [0.2, 0.6, 0.8]);
xlabel('Frequency (rad/s)', 'FontSize', 12);
ylabel('Phase \angle a_k (degrees)', 'FontSize', 12);
title('Phase Spectrum of y_a(t)', 'FontSize', 14);
xlim([-8, 8]);
yticks([-180, -90, 0, 90, 180]);
ylim([-200, 200]);
grid on;
set(gca, 'GridLineStyle', '--', 'FontSize', 10);
ax = gca;
ax.Box = 'off';
hold off;
```

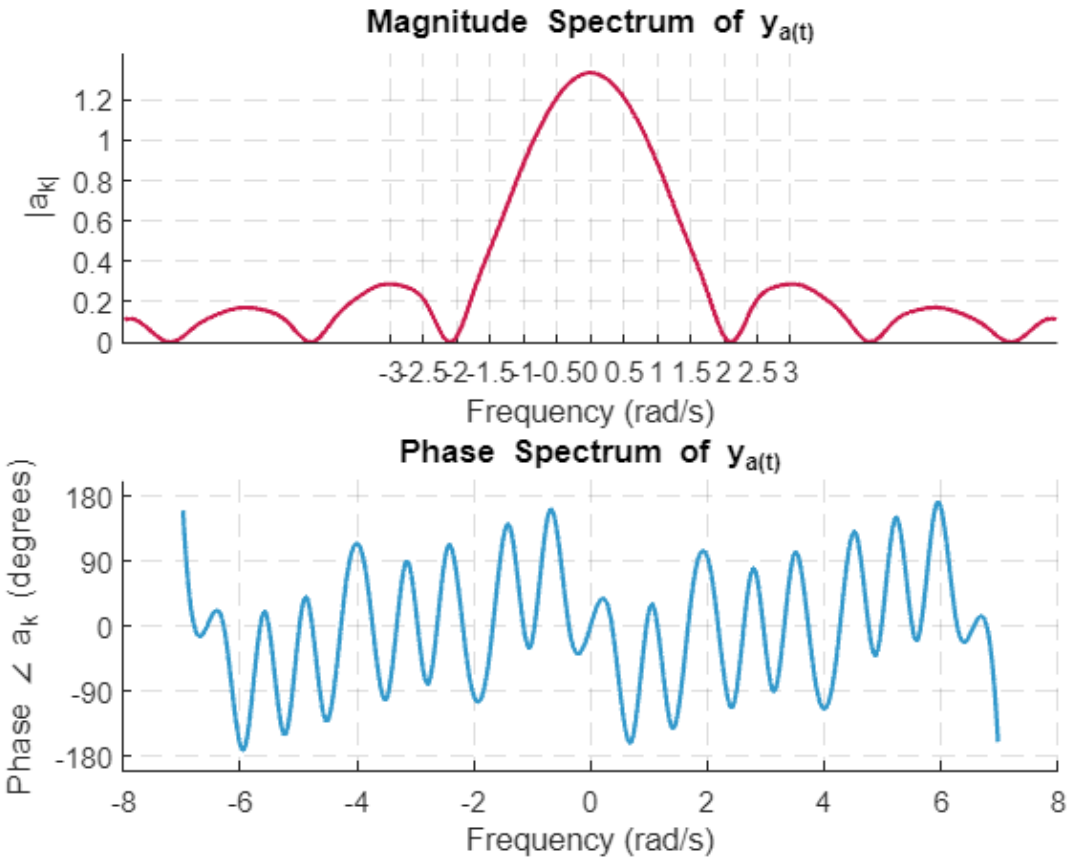

Figure 2: Spectrum Plots of the Rectangular Waveform

d) The given discrete function is

$$z_N[n] = \sum_{k=-N}^{N} a_k e^{j\omega_0 k n T_s}$$

Where a_k's are the Fourier series expansion coefficients that found in b). The code of the plot of this function is given in Code Listing 3. The plot of this function for N = 150 is shown in Figure 3.

Code Listing 3: Discrete Function Plot with Rectangular Waveform FSE Coefficients

```
% Parameters
Ts = 0.1;                       % Sampling period
T = 18;                          % Period of the signal
omega0 = pi / 9;                 % Fundamental frequency
n = -40:319;                     % Range of n values
t = n * Ts;                      % Corresponding time values


% The range of k values for Fourier components
N = 150;                         % Number of terms in Fourier sum
```

```
k_values = -N:N;


% Initialize the signal z_N[n] to 0
z_N = zeros(size(n));


% Compute the Fourier series coefficients and sum terms
for i = 1:length(k_values)
    k = k_values(i);

    if k == 0
        % DC component (a0)
        a_k = 4 / 3;
    else
        % Non-zero Fourier coefficients (a_k)
        a_k = (-4j / (pi * k)) * (exp(-1j * 10 * omega0 * k) - exp(-1j * 7 *
omega0 * k));
    end

    % Sum
    z_N = z_N + a_k * exp(1j * omega0 * k * t);
end


% Define a finer time vector for continuous-style plotting
t_fine = linspace(min(t), max(t), 1000);
z_N_fine = interp1(t, real(z_N), t_fine, 'spline');


% Plot the interpolated continuous-style signal z_N(t)
figure;
plot(t_fine, z_N_fine, 'LineWidth', 1.5, 'Color', [0.2, 0.6, 0.8]);
xlabel('Time (s)', 'FontSize', 12);
ylabel('Amplitude z_N(t)', 'FontSize', 12);
title('Reconstructed Signal z_N(t) for N = 150 (Continuous-Time Style)',
'FontSize', 14);
xlim([min(t_fine), max(t_fine)]);
ylim([-7, 7]);
xticks(-4:2:32);
yticks(-6:2:6);
grid on;
set(gca, 'GridLineStyle', '--', 'FontSize', 10);
ax = gca;
ax.Box = 'off';
hold off;
```

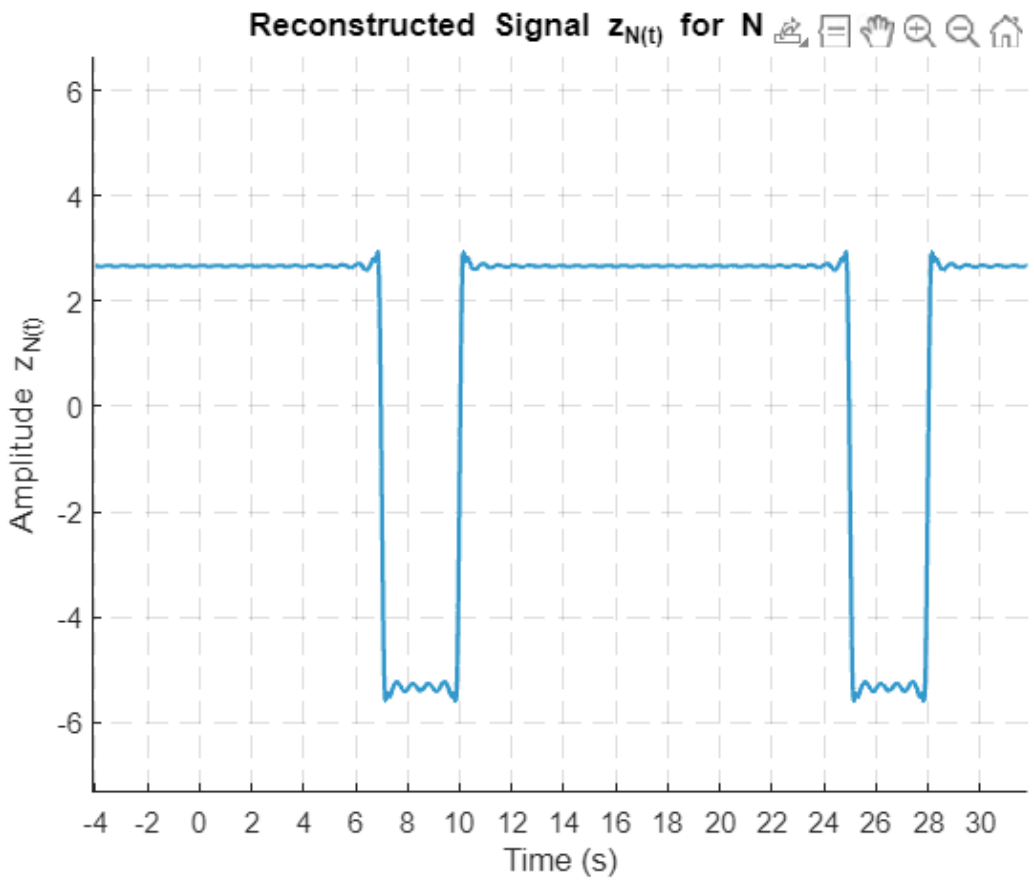

Figure 3: Plot of $z_N[n]$ for N = 150

$y_a(t)$ is a rectangular waveform with sharp transitions between values (discontinuities). The Fourier series approximation $z_N[n]$ on the other hand, tries to approximate this rectangular shape with a finite sum of sinusoids.

The process is repeated with other given values of N with the code given in Code Listing 4.

Code Listing 4: Code of Plot $z_N[n]$ for Various N Values

```
% Parameters
Ts = 0.1;                       % Sampling period
T = 18;                          % Period of the signal
omega0 = pi / 9;                 % Fundamental frequency
n = -40:319;                     % Range of n values
t = n * Ts;                      % Corresponding time values


% Different N values for reconstruction
N_values = [75, 30, 5, 3, 1];


% Loop over each N value and compute z_N[n]
```

```
for idx = 1:length(N_values)
    N = N_values(idx);           % Current value of N
    k_values = -N:N;             % Range of k values for Fourier components


    z_N = zeros(size(n));

    % Compute Fourier series coefficients and sum terms
    for i = 1:length(k_values)
        k = k_values(i);

        if k == 0
            % DC component (a0)
            a_k = 4 / 3;
        else
            % Non-zero Fourier coefficients (a_k)
            a_k = (-4j / (pi * k)) * (exp(-1j * 10 * omega0 * k) - exp(-1j * 7
* omega0 * k));
        end

        % Sum terms for Fourier series reconstruction
        z_N = z_N + a_k * exp(1j * omega0 * k * t);
    end


    % Define a finer time vector for smooth plotting
    t_fine = linspace(min(t), max(t), 1000);
    z_N_fine = interp1(t, real(z_N), t_fine, 'spline');


    % Plot the reconstructed signal z_N(t) for the current N
    figure;
    plot(t_fine, z_N_fine, 'LineWidth', 1.5, 'Color', [0.2, 0.6, 0.8]);
    xlabel('Time (s)', 'FontSize', 12);
    ylabel('Amplitude z_N(t)', 'FontSize', 12);
    title(['Reconstructed Signal z_N(t) for N = ', num2str(N), ' (Continuous-
Time Style)'], 'FontSize', 14);
    xlim([min(t_fine), max(t_fine)]);
    ylim([-9, 9]);
    xticks(-4:2:32);
    yticks(-8:2:8);
    grid on;
    set(gca, 'GridLineStyle', '--', 'FontSize', 10);
    ax = gca;
    ax.Box = 'off';
    hold off;
end
```

The plots of the signal are shown in Figures 4, 5, 6, ,7 and 8.

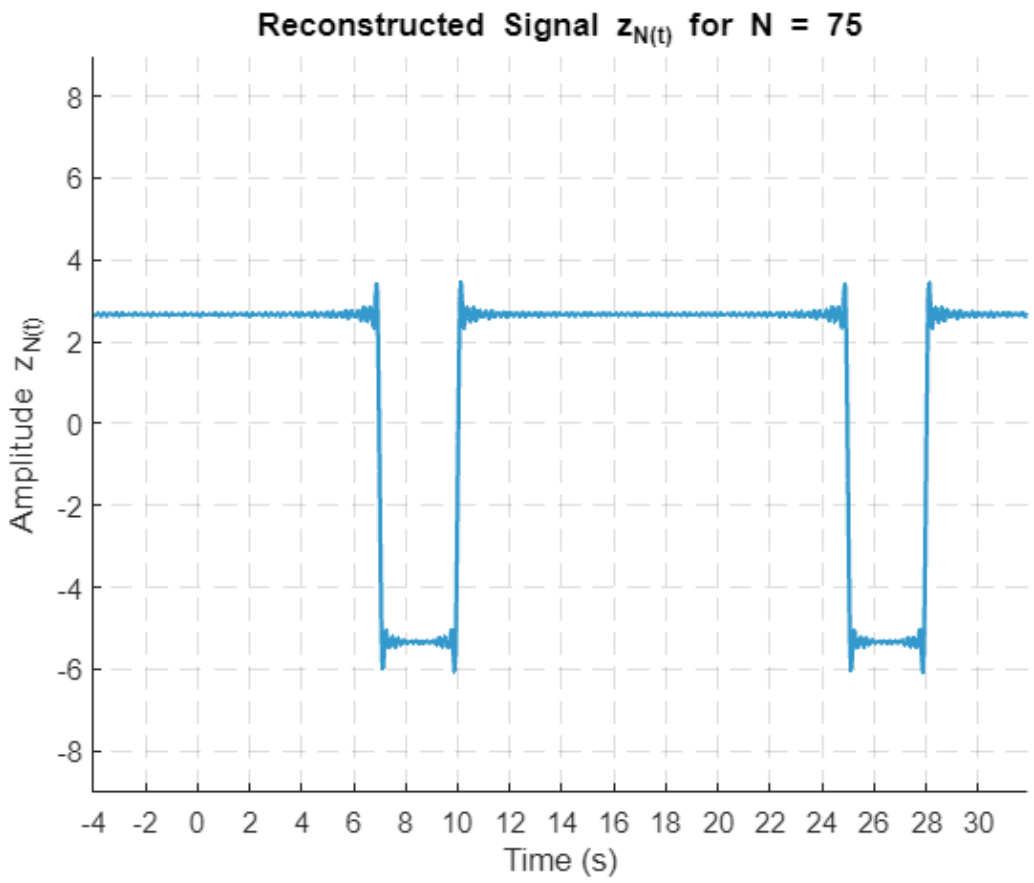

Figure 4: Plot of $z_N[n]$ for N = 75

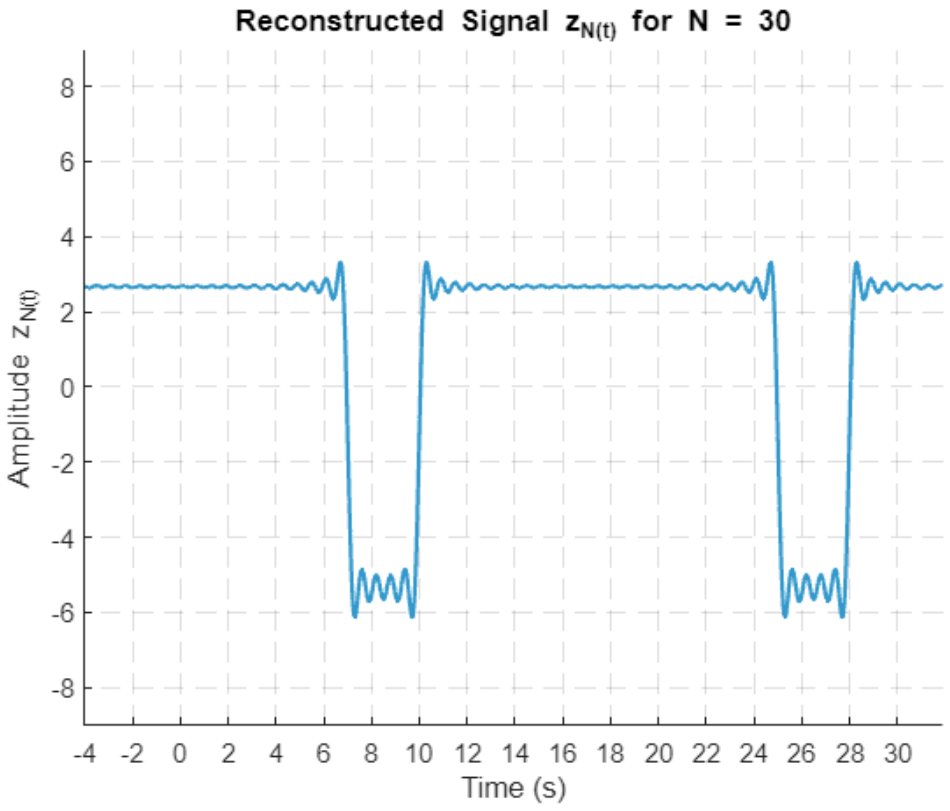

Figure 5: Plot of $z_N[n]$ for N = 30

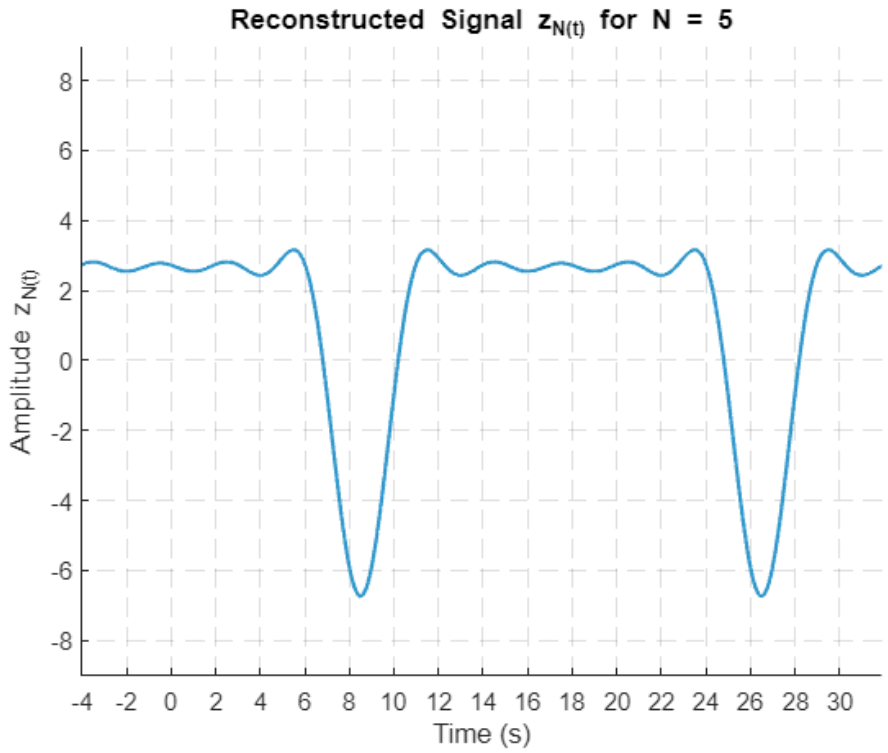

Figure 6: Plot of $z_N[n]$ for N = 5

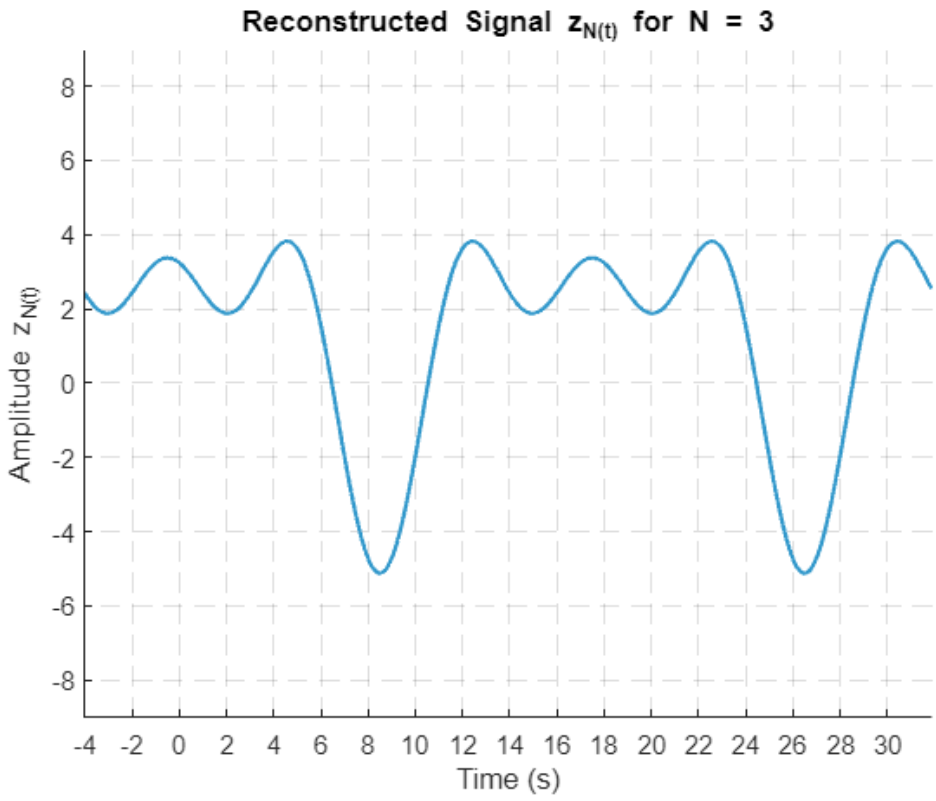

Figure 7: Plot of $z_N[n]$ for N = 3

As the number of Fourier series terms, N, increases, the approximation of $z_N[n]$ to the original signal $y_a(t)$ improve significantly. For small values of N, the approximation is relatively coarse. At these low values, only a few harmonics contribute to the reconstruction, resulting in a signal that captures the general trend of $y_a(t)$ but lacks the precision needed to replicate its sharp transitions. Specifically, the abrupt changes in $y_a(t)$ appear rounded and gradual in the reconstructed signal, which can be attributed to the limited frequency content available when using only a few harmonics. Low-frequency harmonics cannot reproduce sharp edges, so the reconstructed signal appears smoother and less detailed.

As N increases to higher values, such asN = 30, 75, or 150, the quality of the approximation improves considerably. With more harmonics included, the Fourier series can capture finer details and better replicate the sudden transitions in $y_a(t)$ At these higher values of N, the reconstructed signal $z_N[n]$ starts to closely resemble the original waveform, with the high and low sections of the signal becoming more accurate. The overall shape of the signal becomes much clearer, as the additional harmonics allow the Fourier series to more precisely approximate the rectangular nature of, $y_a(t)$.

However, even with high values of N the Gibbs phenomenon remains evident near the discontinuities in $y_a(t)$. The Gibbs phenomenon is characterized by oscillatory overshoots near sharp transitions, which cannot be eliminated by simply adding more harmonics. As N increases, these oscillations become more localized around the discontinuities, meaning they are confined to a narrower region near the sharp edges. Nevertheless, the overshoot itself persists, stabilizing around a fixed percentage of the jump at the transition. This behavior is a fundamental property of Fourier series when approximating signals with abrupt changes, and it highlights the limitations of Fourier series in perfectly reconstructing signals with sharp discontinuities.

In conclusion, increasing N enhances the approximation of $z_N[n]$to $y_a(t)$. yielding a closer match in most regions of the signal. However, the Gibbs phenomenon remains present near discontinuities, indicating a persistent overshoot that cannot be fully resolved through additional harmonics. This analysis demonstrates the strength of Fourier series in approximating periodic signals, while also illustrating the inherent challenges of reconstructing signals with sharp transitions.

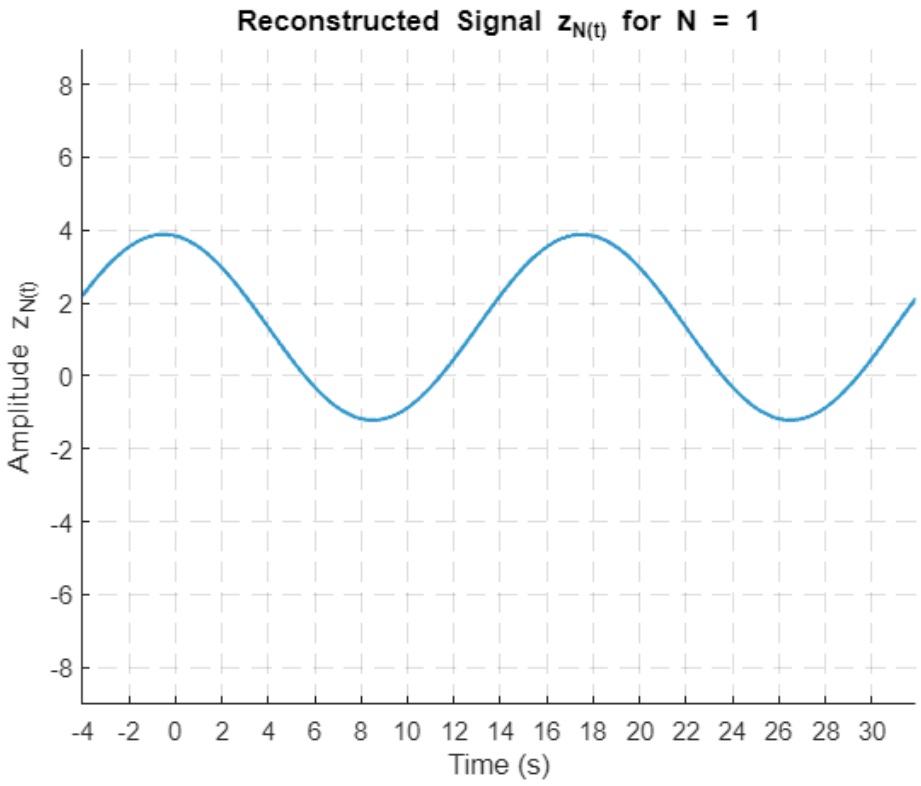

Figure 8: Plot of $z_N[n]$ for N = 1

e) To plot the required harmonics of the rectangular waveform the MATLAB code is given in Code Listing 5 and the plots are shown in Figure 9.

Code Listing 5: Plot of Harmonics of Rectangular Waveform

```
% Parameters
Ts = 0.1;                       % Sampling period
T = 18;                          % Period of the signal
omega0 = pi / 9;                 % Fundamental frequency
n = -40:319;                     % Range of n values
t = n * Ts;                      % Corresponding time values


% Initialize harmonics
harmonic_0 = zeros(size(n));  % Zeroth harmonic (DC component)
harmonic_1 = zeros(size(n));  % First harmonic
harmonic_2 = zeros(size(n));  % Second harmonic
harmonic_3 = zeros(size(n));  % Third harmonic


% Zeroth Harmonic (DC Component)
a_0 = 4 / 3; % DC component
harmonic_0 = a_0 * ones(size(n)); % Constant value


% First Harmonic (k = ±1)
```

```
a_1 = (-4j / pi) * (exp(-1j * 10 * omega0) - exp(-1j * 7 * omega0));
harmonic_1 = real(a_1 * exp(1j * omega0 * t) + conj(a_1) * exp(-1j * omega0 *
t));


% Second Harmonic (k = ±2)
a_2 = (-2j / pi) * (exp(-1j * 2 * 10 * omega0) - exp(-1j * 2 * 7 * omega0));
harmonic_2 = real(a_2 * exp(1j * 2 * omega0 * t) + conj(a_2) * exp(-1j * 2 *
omega0 * t));


% Third Harmonic (k = ±3)
a_3 = (-4j / (3 * pi)) * (exp(-1j * 3 * 10 * omega0) - exp(-1j * 3 * 7 *
omega0));
harmonic_3 = real(a_3 * exp(1j * 3 * omega0 * t) + conj(a_3) * exp(-1j * 3 *
omega0 * t));


% Define finer time vector for smooth plots
t_fine = linspace(min(t), max(t), 1000);


% Interpolate each harmonic for smooth plotting
harmonic_0_fine = interp1(t, harmonic_0, t_fine, 'spline');
harmonic_1_fine = interp1(t, harmonic_1, t_fine, 'spline');
harmonic_2_fine = interp1(t, harmonic_2, t_fine, 'spline');
harmonic_3_fine = interp1(t, harmonic_3, t_fine, 'spline');


% Plotting each harmonic
figure;


% Plot zeroth harmonic (DC component)
subplot(4,1,1);
plot(t_fine, harmonic_0_fine, 'LineWidth', 1.5, 'Color', [0.1, 0.5, 0.8]);
xlabel('Time (s)', 'FontSize', 10);
ylabel('Amplitude', 'FontSize', 10);
title('Zeroth Harmonic (DC Component)', 'FontSize', 12);
grid on;
set(gca, 'GridLineStyle', '--', 'FontSize', 10);
xlim([min(t_fine), max(t_fine)]);


% Plot first harmonic
subplot(4,1,2);
plot(t_fine, harmonic_1_fine, 'LineWidth', 1.5, 'Color', [0.8, 0.1, 0.3]);
xlabel('Time (s)', 'FontSize', 10);
ylabel('Amplitude', 'FontSize', 10);
title('First Harmonic', 'FontSize', 12);
grid on;
set(gca, 'GridLineStyle', '--', 'FontSize', 10);
```

```
xlim([min(t_fine), max(t_fine)]);


% Plot second harmonic
subplot(4,1,3);
plot(t_fine, harmonic_2_fine, 'LineWidth', 1.5, 'Color', [0.2, 0.6, 0.2]);
xlabel('Time (s)', 'FontSize', 10);
ylabel('Amplitude', 'FontSize', 10);
title('Second Harmonic', 'FontSize', 12);
grid on;
set(gca, 'GridLineStyle', '--', 'FontSize', 10);
xlim([min(t_fine), max(t_fine)]);


% Plot third harmonic
subplot(4,1,4);
plot(t_fine, harmonic_3_fine, 'LineWidth', 1.5, 'Color', [0.5, 0.2, 0.8]);
xlabel('Time (s)', 'FontSize', 10);
ylabel('Amplitude', 'FontSize', 10);
title('Third Harmonic', 'FontSize', 12);
grid on;
set(gca, 'GridLineStyle', '--', 'FontSize', 10);
xlim([min(t_fine), max(t_fine)]);


for i = 1:4
    ax = subplot(4,1,i);
    ax.Box = 'off';  % Turn off the box/frame around each plot
end
```

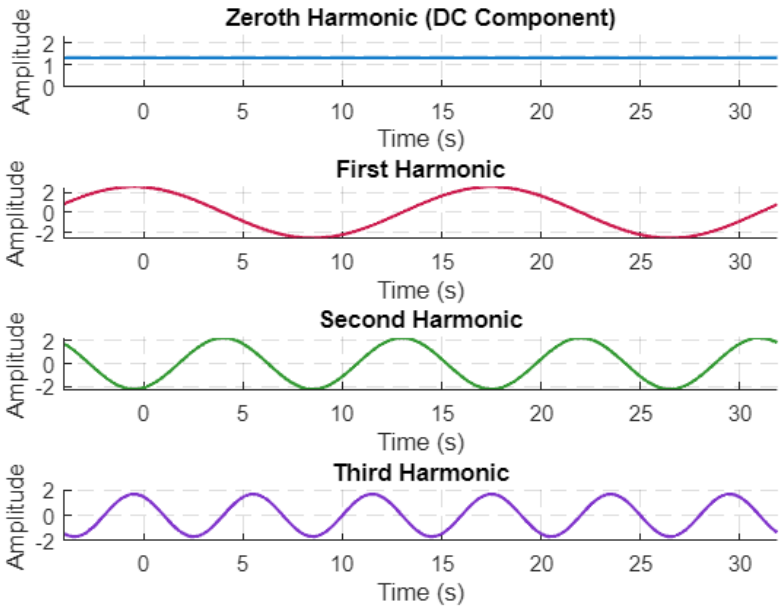

Figure 9: Plots of the Harmonics of the Rectangular Wave

2. The given function for this question is

$$y_a(t) = \left|5\cos\left(\frac{\pi}{9}t\right)\right|.$$

Which is a full-wave rectifier.

A full-wave rectifier is an electronic circuit that converts an alternating current (AC) signal into a direct current (DC) signal by allowing both the positive and negative halves of the AC waveform to contribute to the output. In a typical AC waveform, the signal oscillates above and below zero, creating positive and negative cycles. A full-wave rectifier "rectifies" this signal by flipping the negative half of each cycle to be positive, resulting in a waveform that is entirely above zero.

This process is achieved by inverting the negative portion of the signal so that both halves of the original AC cycle are combined as positive output. Full-wave rectification is commonly used in power supplies to convert AC power from the main supply into DC power for electronic devices. Compared to a half-wave rectifier, which only allows the positive half of the AC cycle to pass through, a full-wave rectifier provides a smoother and more efficient DC output since it uses the entire waveform. This smooth output reduces the ripple effect in the rectified signal, which can be further smoothed with additional filtering components if needed.

In mathematical terms, a full-wave rectified signal can be represented as the absolute value of a sinusoidal function. For example, a full-wave rectified cosine wave is expressed as the absolute value of the cosine function, where all negative values of the cosine wave are converted to positive, resulting in a waveform that oscillates only in the positive range.

a) The plot of this signal is shown in Figure 10. The code of this part is given in Code Listing A.1.

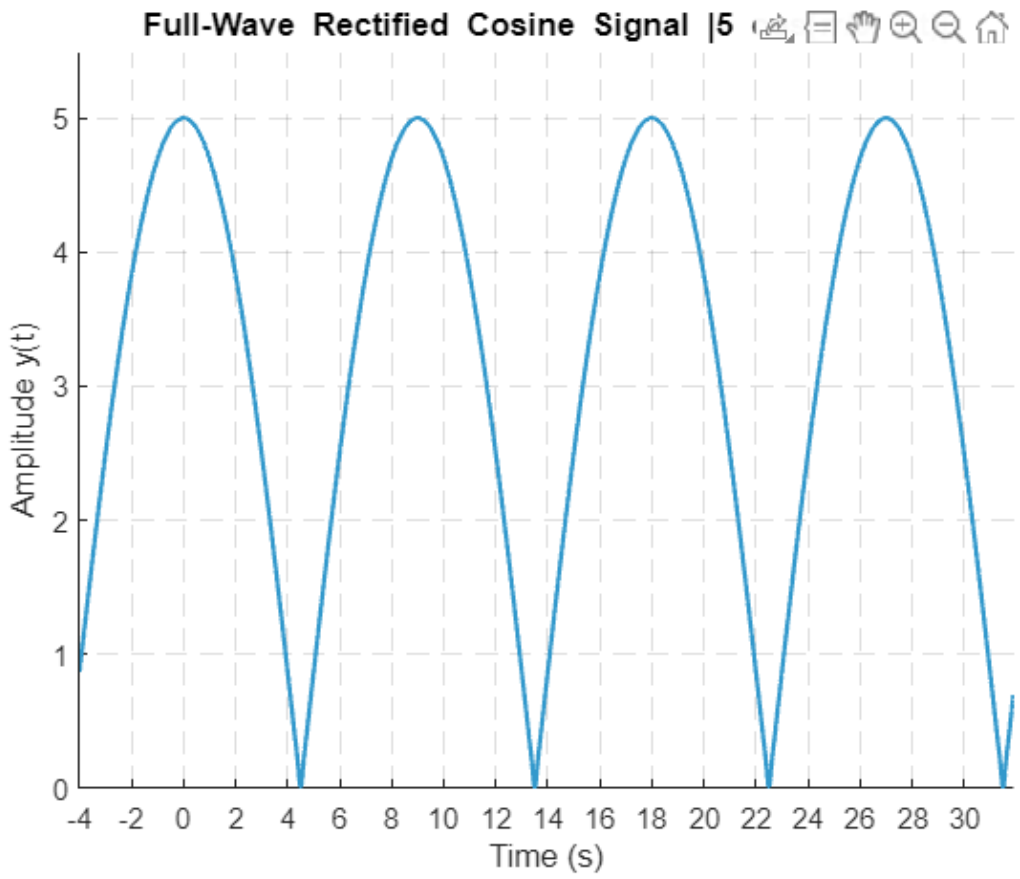

Figure 10: Full-Wave Rectifier Plot

b) The Fourier series expansion of full-wave rectifier is calculated by using the approach mentioned in Background.

The fundamental frequency of the waveform is

$$\omega_0 = \frac{2\pi}{T} = \frac{2\pi}{9} \text{ rad/s}$$

DC component a_0 is found as

$$a_0 = \frac{1}{T}\int_0^T y_a(t)\ dt = \frac{2}{9}\int_0^{4.5} 5\cos(\frac{\pi}{9}t)\, dt = \frac{10}{\pi}$$

Fourier coefficients a_k are found as

$$a_k = \frac{2}{9}\int_0^9 \left|5\cos\left(\frac{\pi}{9}t\right)\right| \cos\left(\frac{2\pi k}{9}t\right)\ dt$$

$$a_k = \frac{20}{9}\int_0^{4.5} \frac{1}{2}\left(\cos\left(\frac{\pi(1-2k)}{9}t\right) + \cos\left(\frac{\pi(1+2k)}{9}t\right)\right) dt$$

$$a_k = \frac{10}{9}\left[\frac{9}{\pi(1-2k)}\sin\left(\frac{\pi(1-2k)}{9}t\right) + \frac{9}{\pi(1+2k)}\sin\left(\frac{\pi(1+2k)}{9}t\right)\right]$$

c) The plot of the spectrum of this signal is shown in Figure 11. The code is listed in Code Listing A.2.

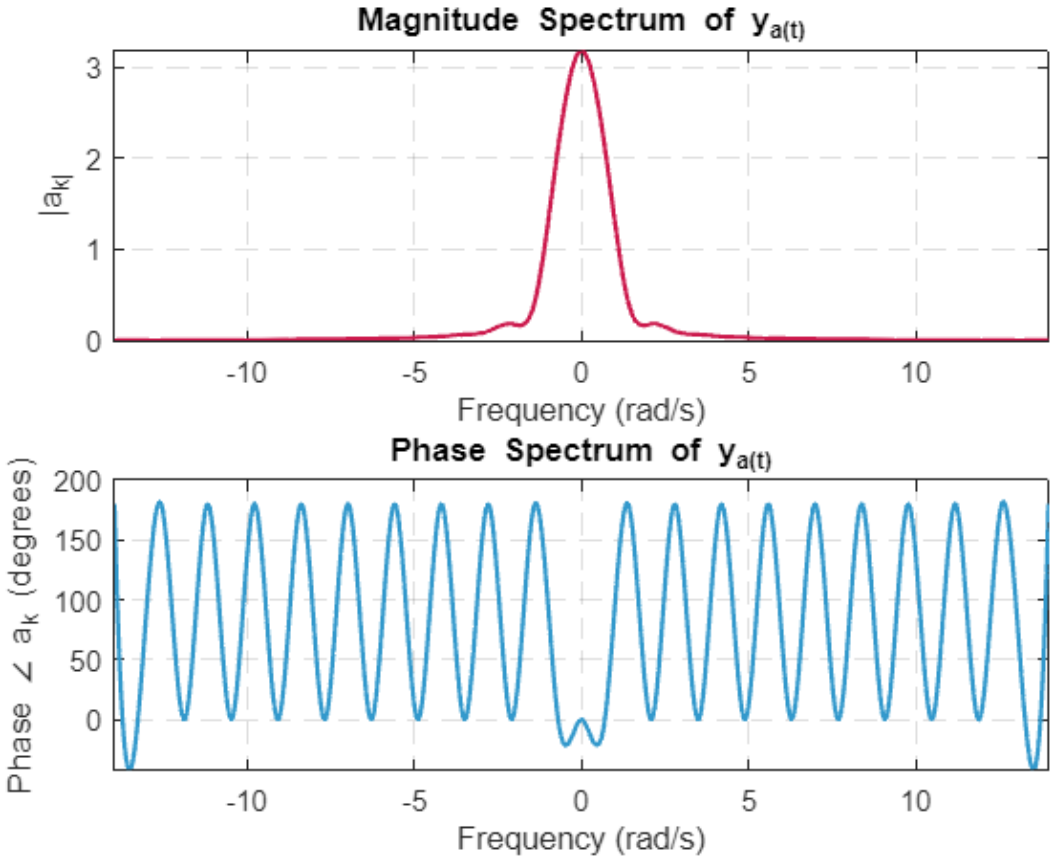

Figure 11: The Spectrum Plot of the Full-Wave Rectifier

d) The plot of $z_N[n]$ with FSE coefficients of the full wave rectifier is shown in Figure 12, and its code is given in Code Listing A.3.

Plots are shown in Figure 12, 13, 14, 15, 16 and 17.

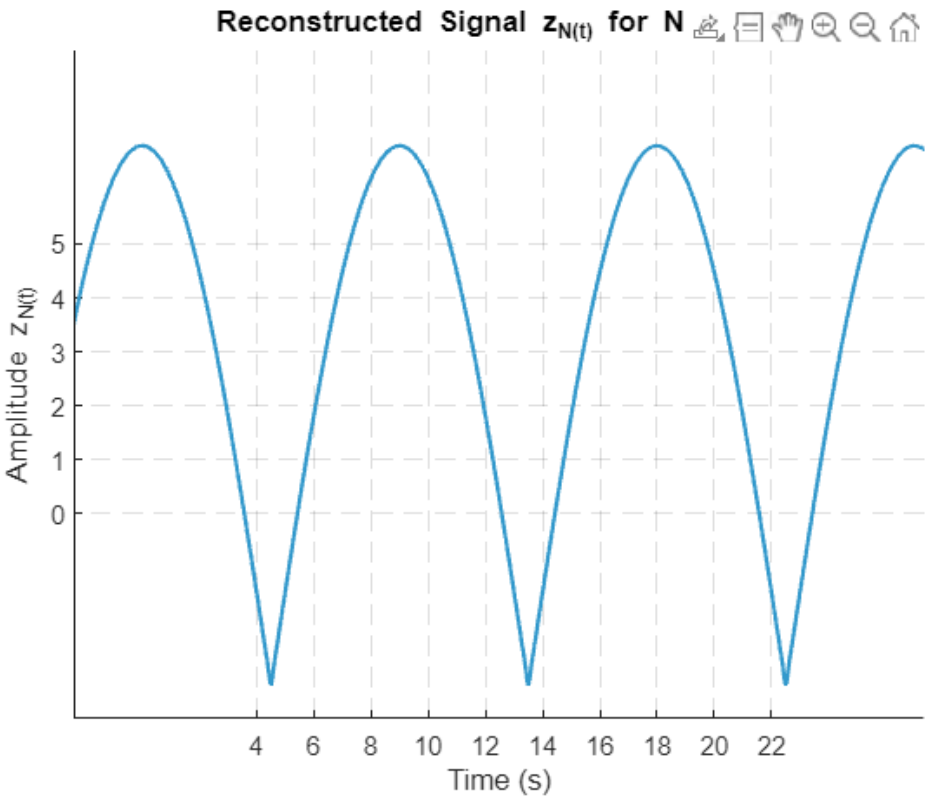

Figure 12: Plot of $z_N[n]$ for N = 150

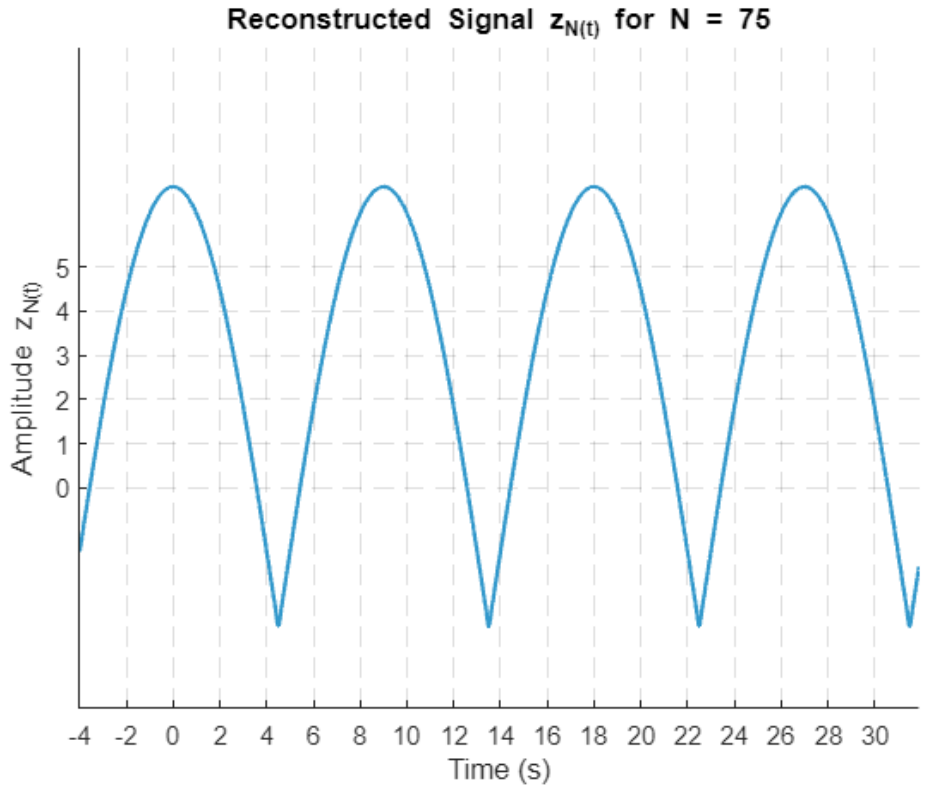

Figure 13: Plot of $z_N[n]$ for N = 75

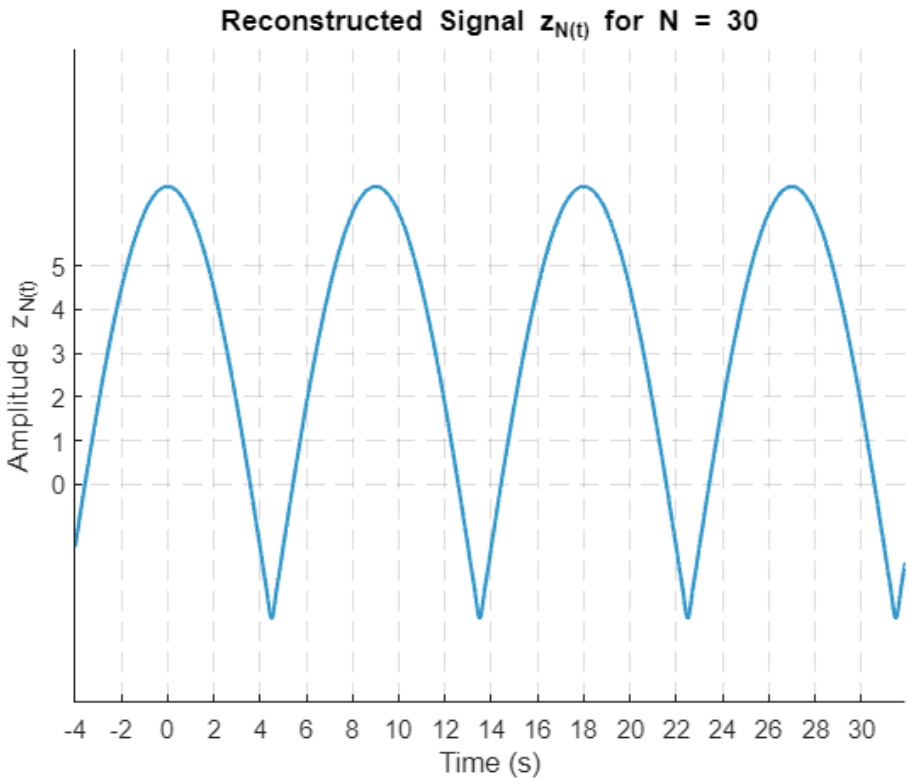

Figure 14: Plot of $z_N[n]$ for N = 30

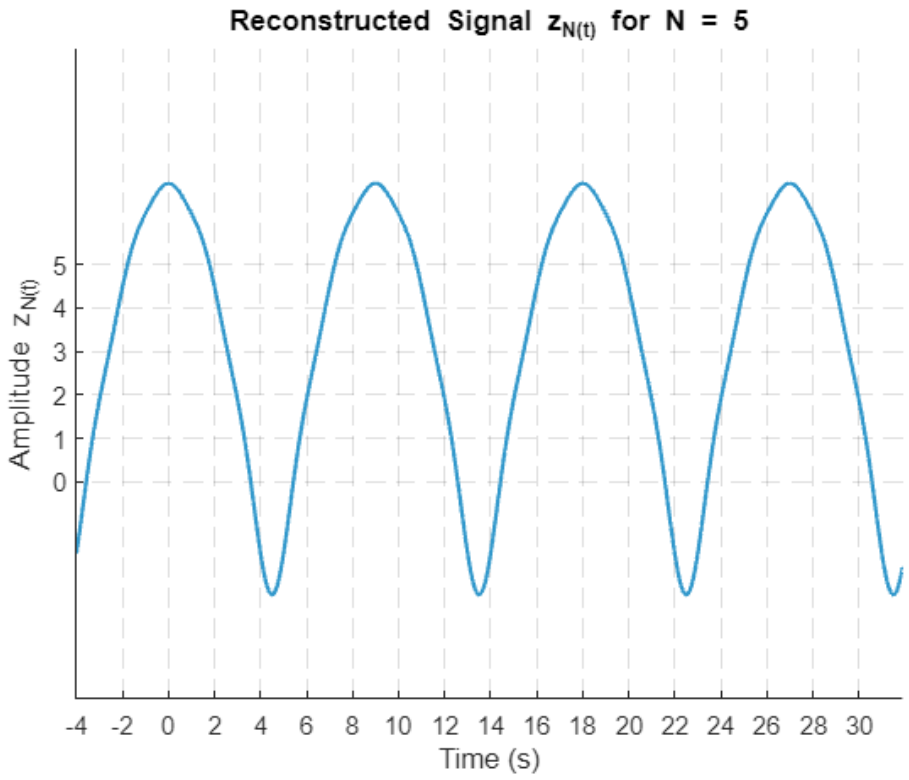

Figure 15: Plot of $z_N[n]$ for N = 5

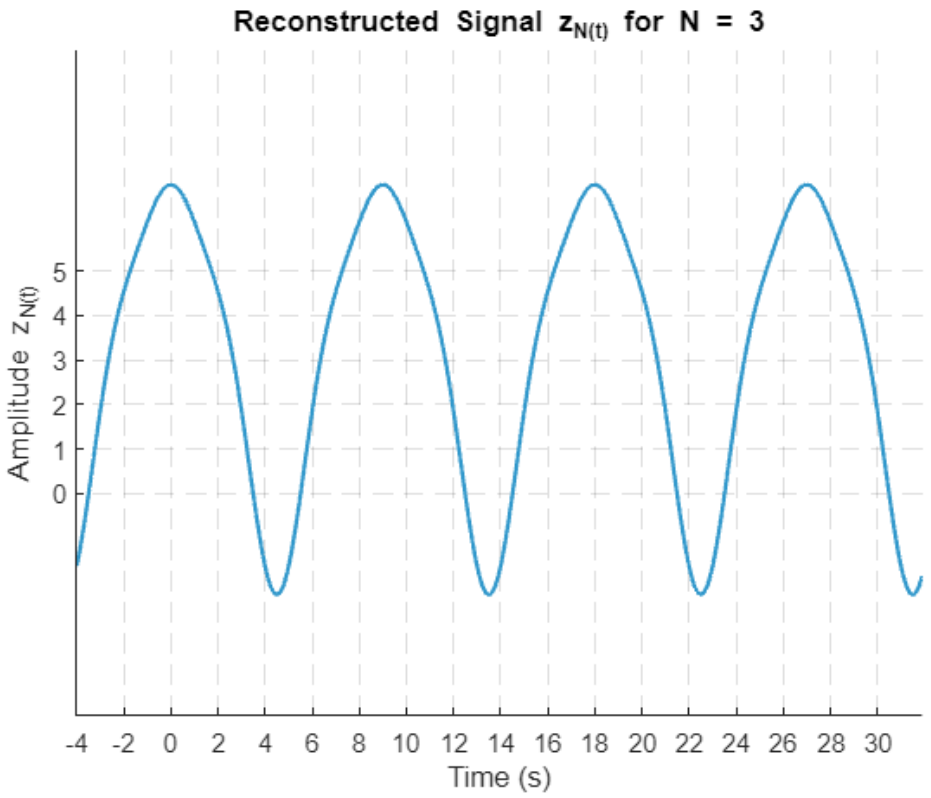

Figure 16: Plot of $z_N[n]$ for N = 3

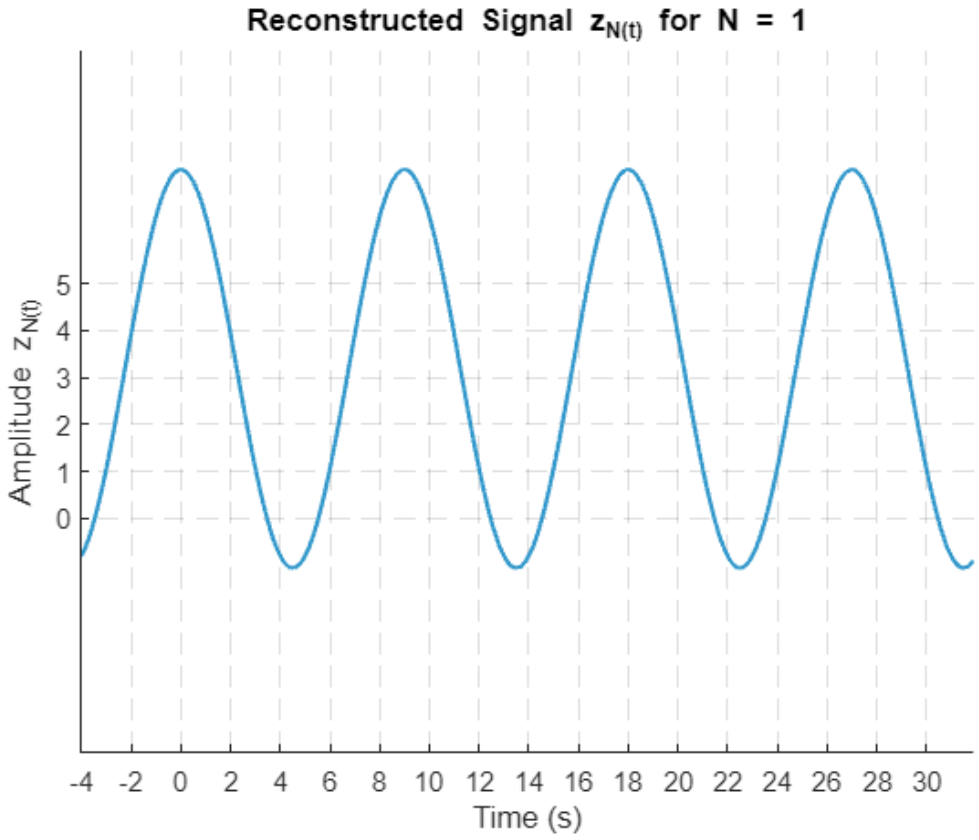

Figure 17: Plot of $z_N[n]$ for N = 1

The results are like the previous ones.

e) Plots of the harmonics of the full-wave rectifier are shown in Figure 18, and it code is given in Code Listing A.4.

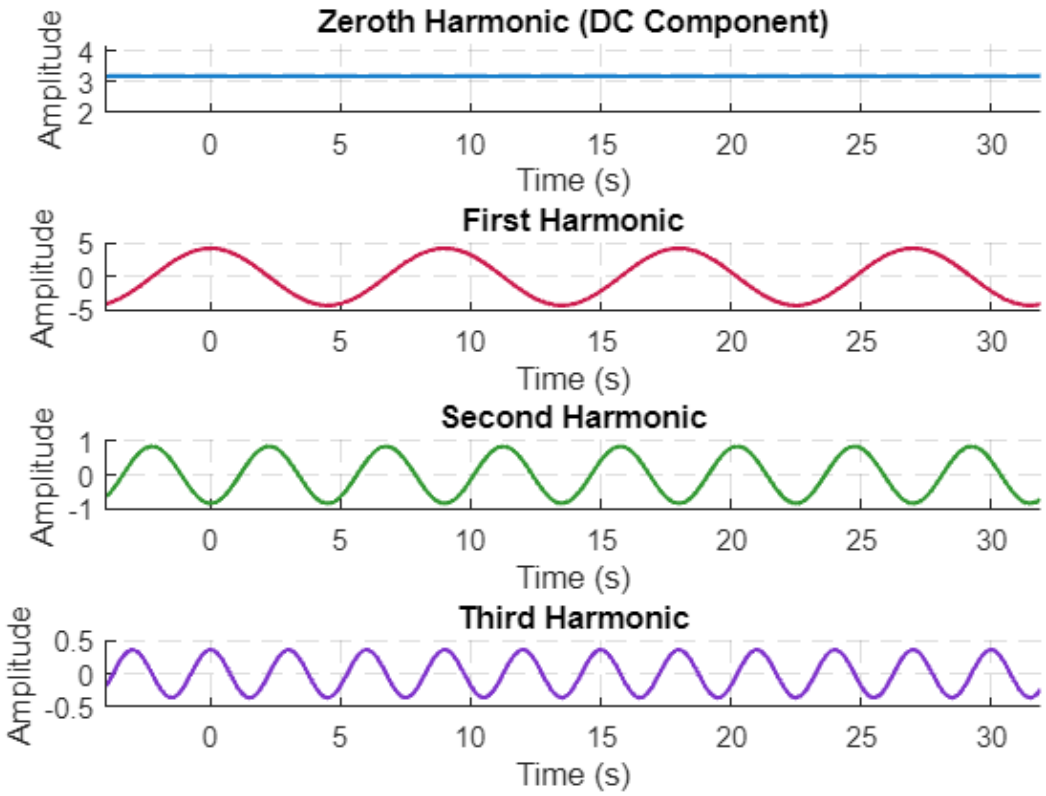

Figure 18: Plots of the Harmonics of the Full-Wave Rectifier

3. The signal given for this part is

$$y_a(t) = \begin{cases} \left|5\cos\left(\frac{\pi}{9}t\right)\right| & t \in [-4.5, 4.5)\,\mathrm{s}\,. \\ 0 \quad t \in [4.5, 13.5)\mathrm{s}. \end{cases}$$

Which is a half wave rectifier.

A half-wave rectifier is an electronic circuit that converts an alternating current (AC) signal into a direct current (DC) signal by allowing only one half of each AC cycle to pass through. In an AC signal, the current oscillates between positive and negative values, creating a waveform that alternates above and below zero. A half-wave rectifier blocks the negative portion of each cycle, allowing only the positive half to reach the output.

In operation, the rectifier uses a diode to conduct current only during the positive half-cycle of the AC waveform. When the input voltage is positive, the diode allows current to pass through, creating a positive output. When the input voltage becomes negative, the diode blocks the current, resulting in zero output during that portion of the cycle.

The result is a waveform that has gaps where the negative half-cycles of the original AC signal used to be. This produces a pulsating DC signal that still has significant ripple, as it only includes half of the original AC wave. To smooth out the resulting DC signal, additional filtering components, such as capacitors, are often used in practical applications.

Half-wave rectifiers are commonly used in low-power applications and basic power supplies, although they are less efficient than full-wave rectifiers. Because they discard half of the input wave, they deliver lower output power and are generally not suitable for applications that require a stable, continuous DC output.

a) The plot of the half wave rectifier is shown in Figure 19 and its code is listed in Code Listing A.5.

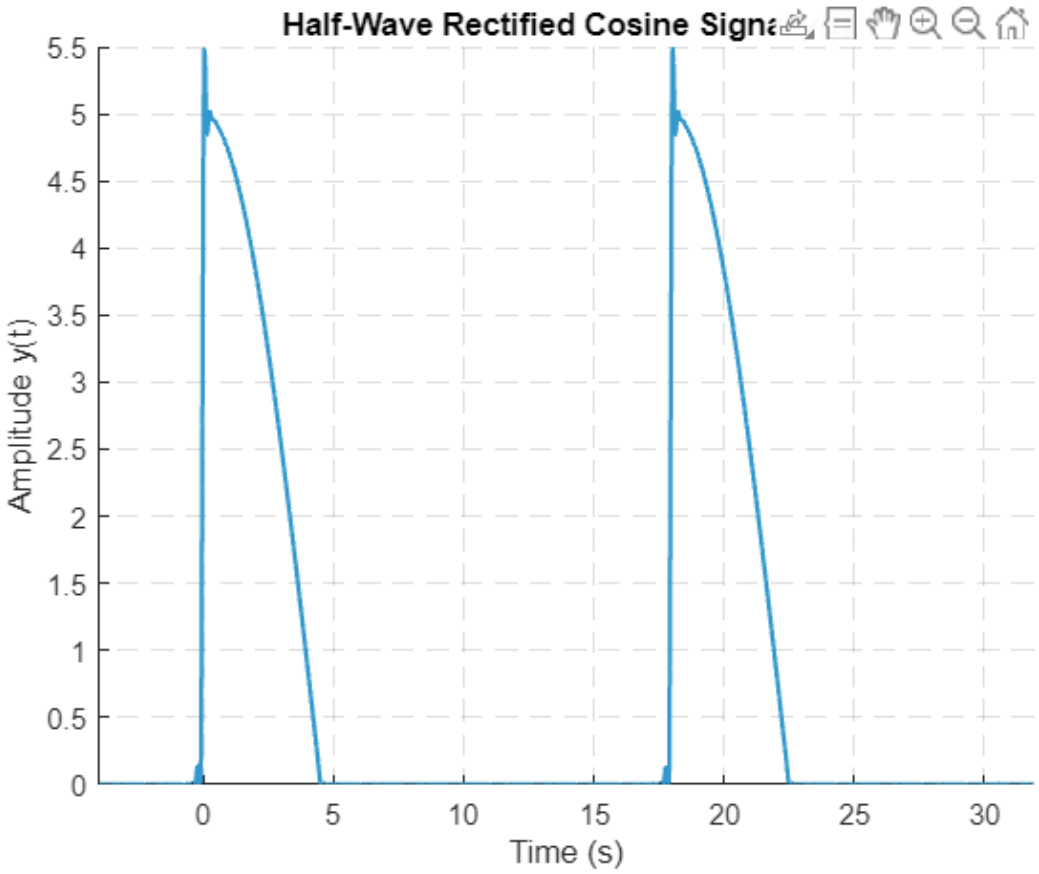

Figure 19: Plot of the Half-Wave Rectifier

b) The Fourier series expansion of half-wave rectifier is calculated by using the approach mentioned in Background.

The fundamental frequency of the waveform is

$$\omega_0 = \frac{2\pi}{T} = \frac{\pi}{9} \text{ rad/s}$$

DC component a_0 is found as

$$a_0 = \frac{1}{18} \cdot 2 \int_0^{4.5} 5 \cos\left(\frac{\pi}{9} t\right) \, dt$$

$$= \frac{10}{18} \int_0^{4.5} 5 \cos\left(\frac{\pi}{9} t\right) \, dt$$

$$a_0 = \frac{10}{18} \cdot \frac{9}{\pi} = \frac{90}{18\pi} = \frac{\pi}{5}$$

Fourier coefficients a_k are found as

$$a_k = \frac{2 \cdot 5}{18} \int_0^{4.5} \cos\left(\frac{\pi}{9} t\right) \cos\left(\frac{\pi k}{9} t\right) \, dt$$

$$= \frac{10}{18} \int_0^{4.5} \cos\left(\frac{\pi}{9} t\right) \cos\left(\frac{\pi k}{9} t\right) \, dt$$

$$a_k = \frac{5}{2\pi} \left[\frac{\sin\left(\frac{\pi(1-k)}{2}\right)}{1-k} + \frac{\sin\left(\frac{\pi(1+k)}{2}\right)}{1+k} \right]$$

c) The plots of the harmonics of the half-wave rectifier are shown in Figure 20 and its code is given in Code Listing A.6.

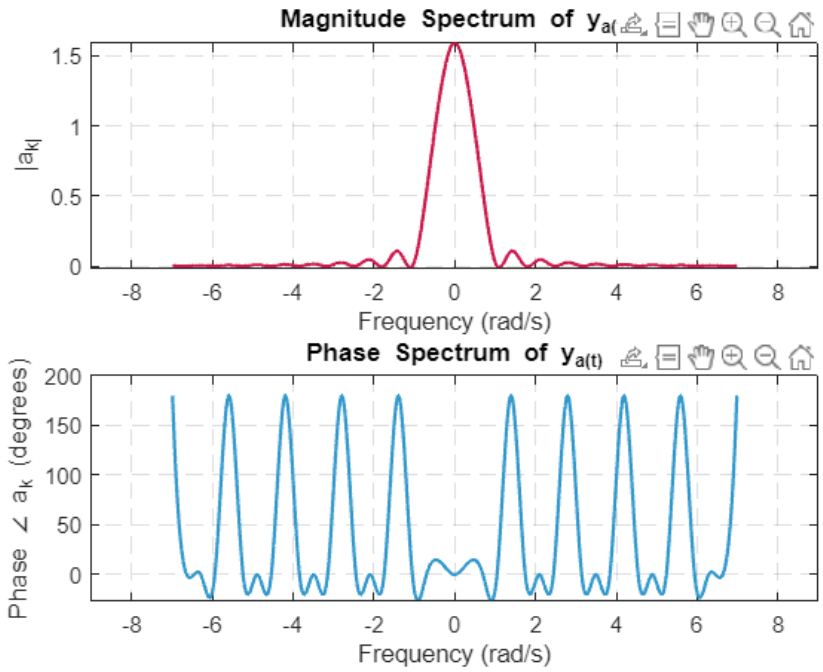

Figure 20: The Plots of the Spectrum of Half-Wave Rectifier

d) Plots of the $z_N[n]$ are shown in Figures 21, 22, 23, 24, 25, and 26. Its code is listed in Code Listing A.7.

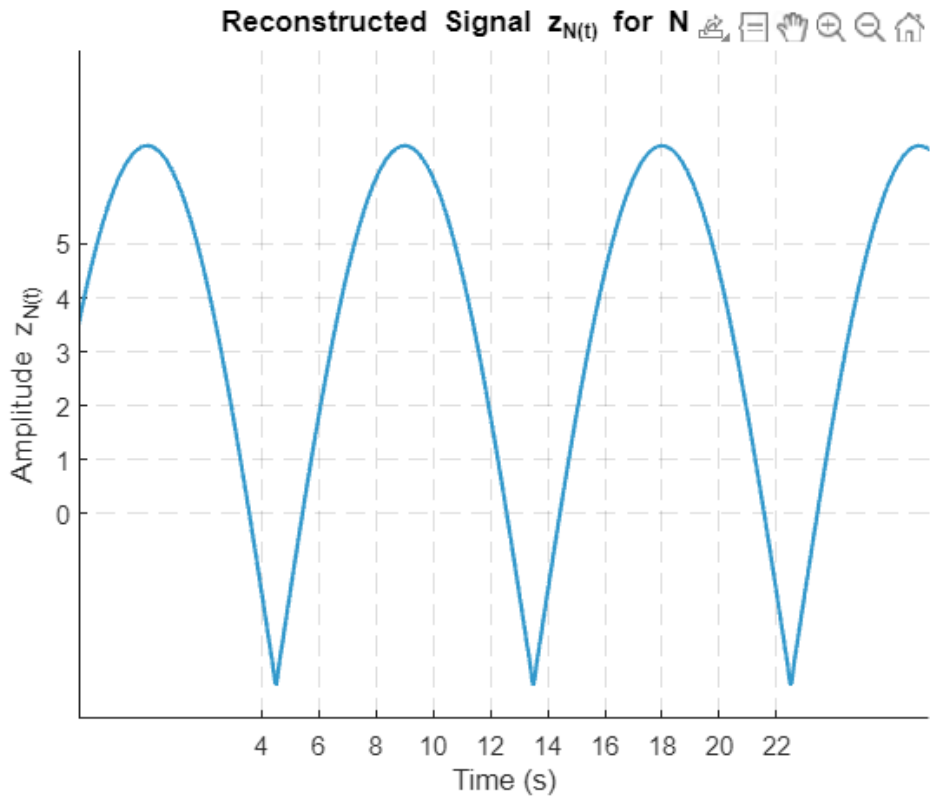

Figure 21: Plot of $z_N[n]$ for N = 150

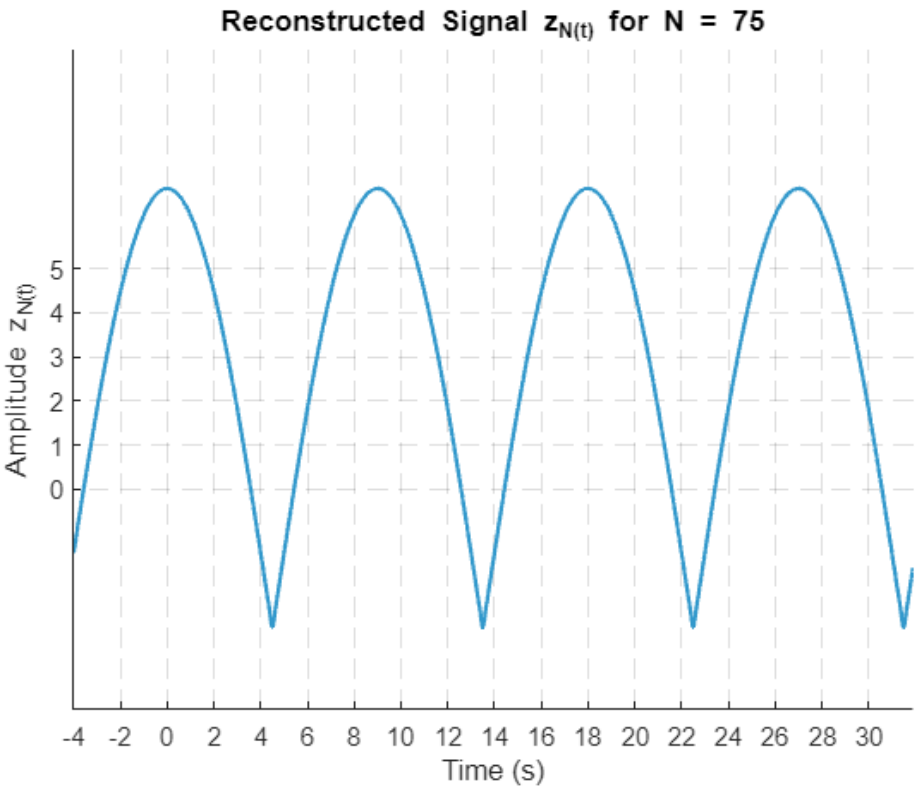

Figure 22: Plot of $z_N[n]$ for N = 75

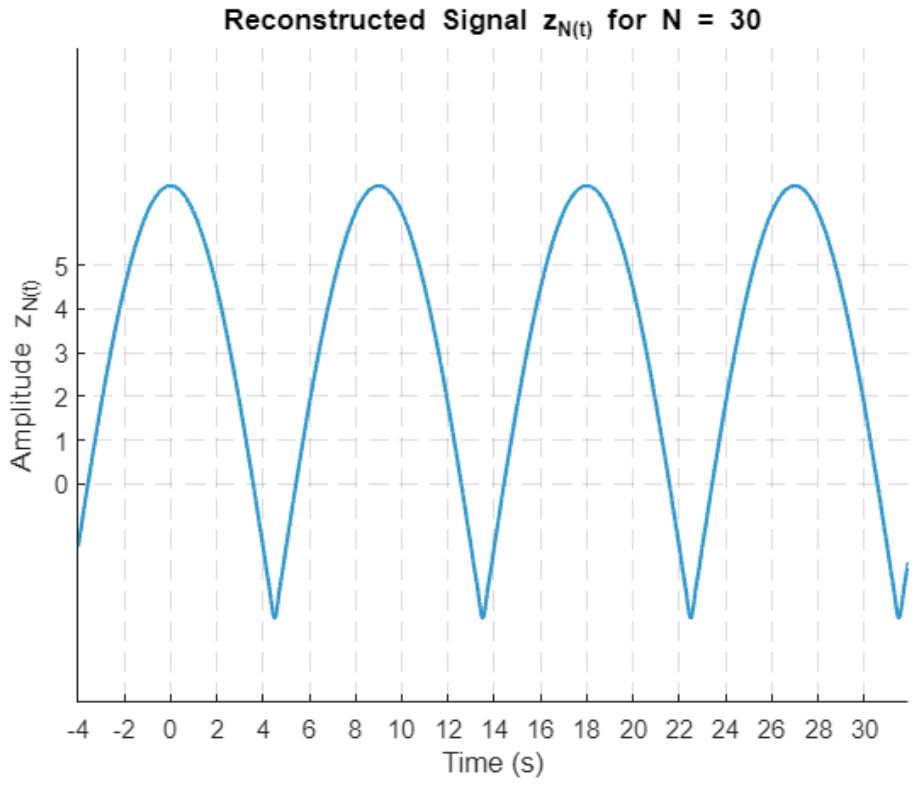

Figure 23: Plot of $z_N[n]$ for N = 30

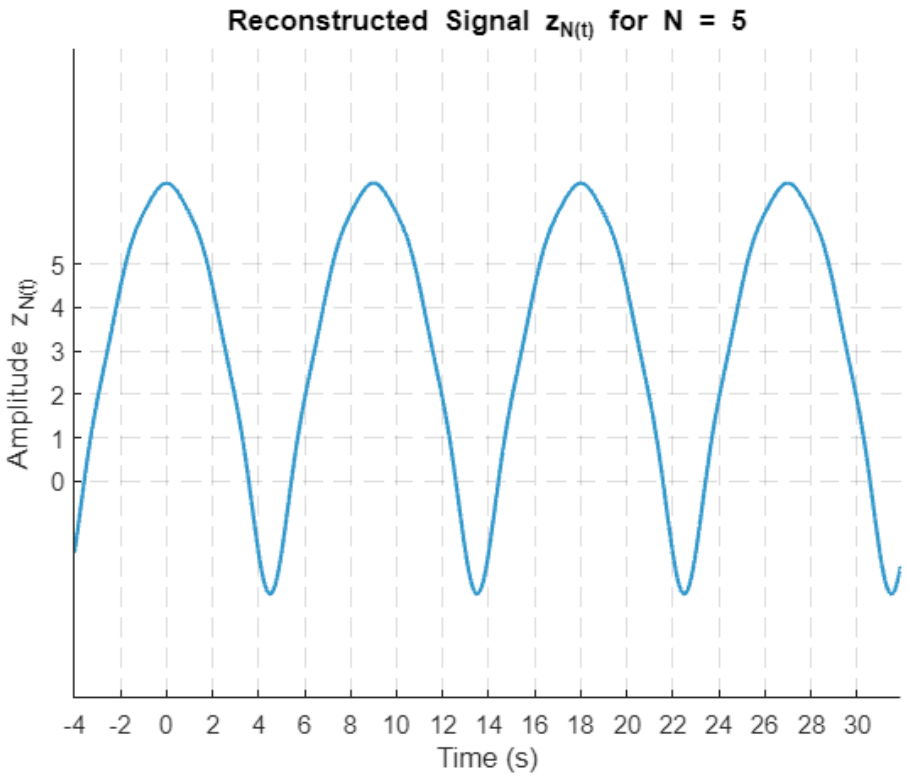

Figure 24: Plot of $z_N[n]$ for N = 5

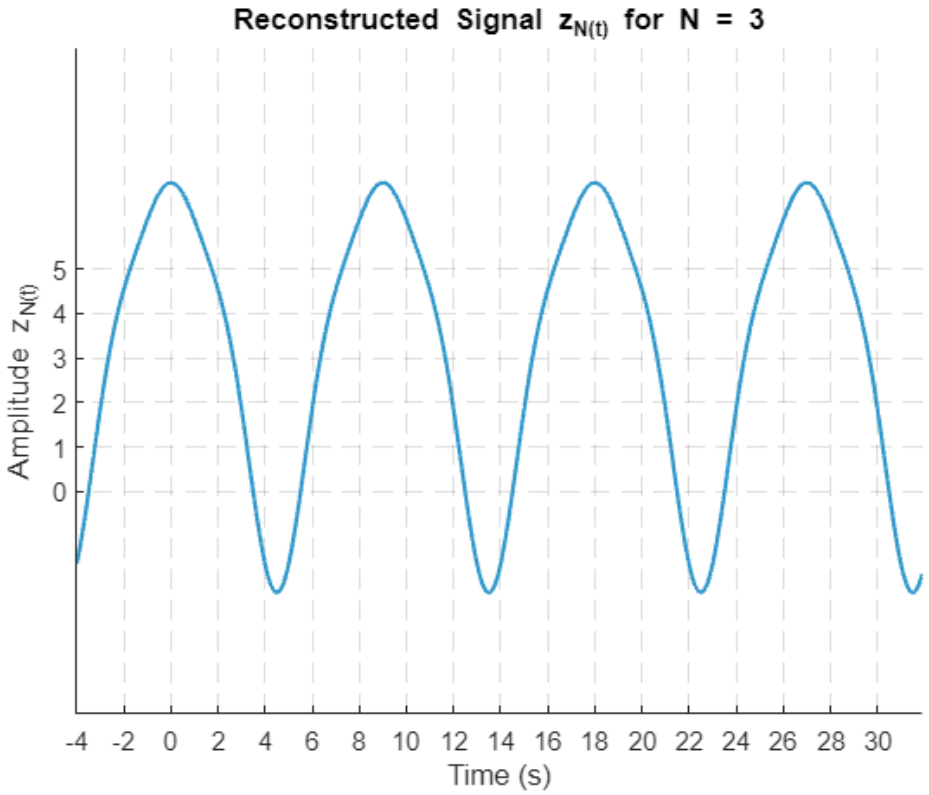

Figure 25: Plot of $z_N[n]$ for N = 3

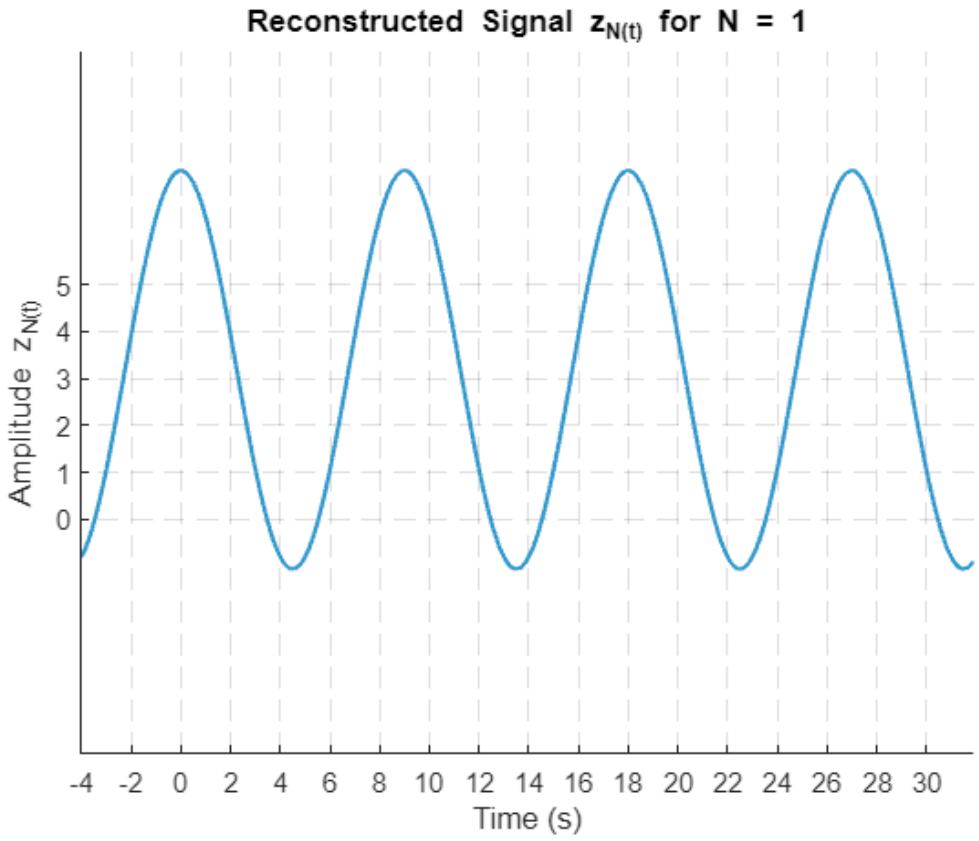

Figure 26: Plot of $z_N[n]$ for N = 1

e) The harmonics of the half wave rectifier are shown in Figure 27 and its code is listed in Code Listing A.8.

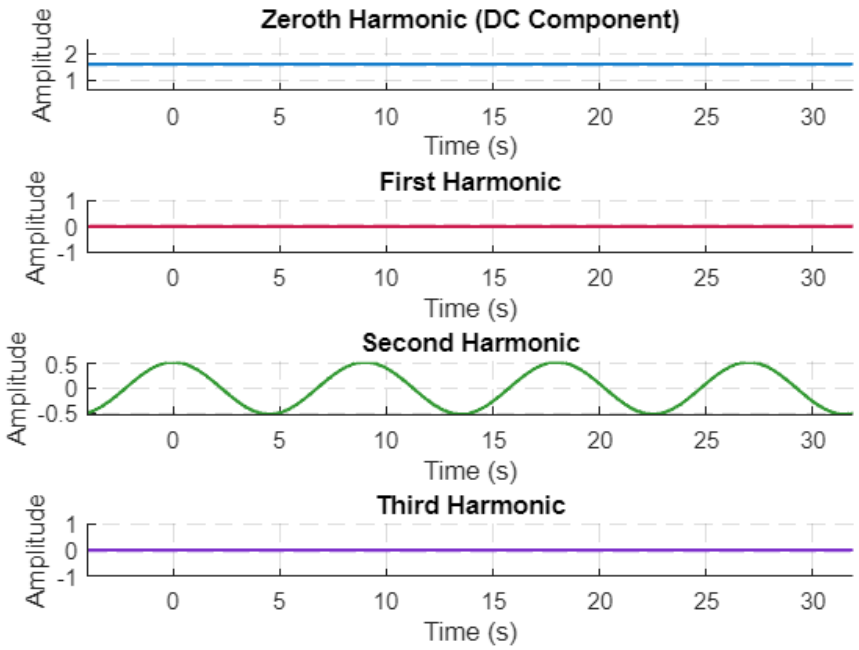

Figure 27: Plots of the Harmonics of the Half Wave Recitifer

Appendices

Appendix A

Code Listing A.1: Full-Wave Rectifier Plot

```
% Parameters
Ts = 0.1;                       % Sampling period
T = 9;                          % Period of the full-wave rectified signal
n = -40:319;                    % Range of n values
t = n * Ts;                     % Corresponding time values


% Define the discretized version
y = abs(5 * cos(pi * t / 9));  % Full-wave rectified cosine


% Define a finer time vector for smooth plotting
t_fine = linspace(min(t), max(t), 1000);


% Interpolate the full-wave rectified cosine signal for smooth plotting
y_fine = interp1(t, y, t_fine, 'spline');


% Plot the interpolated signal y(t)
figure;
plot(t_fine, y_fine, 'LineWidth', 1.5, 'Color', [0.2, 0.6, 0.8]);
xlabel('Time (s)', 'FontSize', 12);
ylabel('Amplitude y(t)', 'FontSize', 12);
title('Full-Wave Rectified Cosine Signal |5 cos(\pi/9 * t)|', 'FontSize', 14);
xlim([min(t_fine), max(t_fine)]);
ylim([0, 5.5]);
xticks(-4:2:32);
yticks(0:1:5);
grid on;
set(gca, 'GridLineStyle', '--', 'FontSize', 10);
ax = gca;
ax.Box = 'off';


hold off;
```

Code Listing A.2: Spectrum of the Full Wave Rectifier

```
% Parameters
T = 9;                          % Period of the signal
omega0 = 2 * pi / T;            % Fundamental frequency
max_k = 20;                     % Maximum harmonic to calculate
```

```
% Initialize arrays for frequencies, magnitudes, and phases
k_values = -max_k:max_k;      % Range of k values
frequencies = k_values * omega0;   % Corresponding frequencies
a_k = zeros(size(k_values));       % Array to store Fourier coefficients


% Compute Fourier coefficients using the general formula
for i = 1:length(k_values)
    k = k_values(i);
    if k == 0
        % DC component
        a_k(i) = 10 / pi;
    else
        % General formula for non-zero k
        a_k(i) = (10 / pi) * ( ...
            sin(pi * (1 - 2 * k) / 2) / (1 - 2 * k) + ...
            sin(pi * (1 + 2 * k) / 2) / (1 + 2 * k));
    end
end


% Magnitude and phase for the spectrum plot
a_k_magnitude = abs(a_k);
a_k_phase = angle(a_k);


% Define finer frequency vector for smooth plotting
freq_fine = linspace(min(frequencies), max(frequencies), 1000);


% Interpolate magnitude and phase for smooth plotting
magnitude_continuous = interp1(frequencies, a_k_magnitude, freq_fine,
'spline');
phase_continuous = interp1(frequencies, rad2deg(a_k_phase), freq_fine,
'spline');


% The Magnitude Spectrum Plot
figure;
subplot(2,1,1);
plot(freq_fine, magnitude_continuous, 'LineWidth', 1.5, 'Color', [0.8, 0.1,
0.3]);
xlabel('Frequency (rad/s)', 'FontSize', 12);
ylabel('|a_k|', 'FontSize', 12);
title('Magnitude Spectrum of y_a(t)', 'FontSize', 14);
grid on;
set(gca, 'GridLineStyle', '--', 'FontSize', 10);
xlim([-max(frequencies), max(frequencies)]);


% The Phase Spectrum Plot
subplot(2,1,2);
```

```
plot(freq_fine, phase_continuous, 'LineWidth', 1.5, 'Color', [0.2, 0.6, 0.8]);
xlabel('Frequency (rad/s)', 'FontSize', 12);
ylabel('Phase \angle a_k (degrees)', 'FontSize', 12);
title('Phase Spectrum of y_a(t)', 'FontSize', 14);
grid on;
set(gca, 'GridLineStyle', '--', 'FontSize', 10);
xlim([-max(frequencies), max(frequencies)]);
```

Code Listing A.3: Plot of $z_N[n]$ with FSE Coefficients of the Full Wave Rectifier

```
% Parameters
Ts = 0.1;                       % Sampling period
T = 9;                          % Period of the signal
omega0 = 2 * pi / T;            % Fundamental frequency
n = -40:319;                    % Range of n values
t = n * Ts;                     % Corresponding time values


% Range of N values for reconstruction
N_values = [150, 75, 30, 5, 3, 1];


% Maximum harmonic for calculation
max_k = max(N_values);


% Calculate Fourier coefficients
a_k = zeros(2 * max_k + 1, 1);  % Array to hold coefficients from -max_k to
+max_k
k_indices = -max_k:max_k;       % Range of k values


% Populating the Fourier coefficients using the general formula
for i = 1:length(k_indices)
    k = k_indices(i);
    if k == 0
        a_k(i) = 10 / pi;  % DC component
    else
        % General formula for non-zero k
        a_k(i) = (10 / pi) * ( ...
            sin(pi * (1 - 2 * k) / 2) / (1 - 2 * k) + ...
            sin(pi * (1 + 2 * k) / 2) / (1 + 2 * k));
    end
end


% Loop over each N value and compute the truncated Fourier series
reconstruction
for idx = 1:length(N_values)
    N = N_values(idx);          % Current value of N
    z_N = zeros(size(n));
```

```
    % Sum the Fourier series components for the range k = -N to k = N
    for i = 1:length(k_indices)
        k = k_indices(i);
        if abs(k) <= N  % Only include terms up to the current N value
            % Find the Fourier coefficient for this k
            a_k_current = a_k(k + max_k + 1);  % Shift index for 1-based
indexing
            % Add the current Fourier component to z_N
            z_N = z_N + a_k_current * exp(1j * omega0 * k * t);
        end
    end

    % Define finer time vector for smooth plotting
    t_fine = linspace(min(t), max(t), 1000);
    z_N_fine = interp1(t, real(z_N), t_fine, 'spline');


    % Plot the real part of z_N for the current N
    figure;
    plot(t_fine, z_N_fine, 'LineWidth', 1.5, 'Color', [0.2, 0.6, 0.8]);
    xlabel('Time (s)', 'FontSize', 12);
    ylabel('Amplitude z_N(t)', 'FontSize', 12);
    title(['Reconstructed Signal z_N(t) for N = ', num2str(N)], 'FontSize',
14);
    xlim([min(t_fine), max(t_fine)]);
    ylim([-5, 10]);
    xticks(-4:2:32);
    yticks(0:1:5);
    grid on;
    set(gca, 'GridLineStyle', '--', 'FontSize', 10);
    ax = gca;
    ax.Box = 'off';
    hold off;
end
```

Code Listing A.4: Harmonics of the Full-Wave Rectifier

```
% Parameters
Ts = 0.1;                       % Sampling period
T = 9;                          % Period of the signal
omega0 = 2 * pi / T;            % Fundamental frequency
n = -40:319;                    % Range of n values
t = n * Ts;                     % Corresponding time values


% Fourier coefficients
a_0 = 10 / pi;
a_1 = 20 / (3 * pi);
a_2 = -4 / (3 * pi);
```

```
a_3 = 4 / (7 * pi);


% Calculate individual harmonics
harmonic_0 = a_0 * ones(size(n));                    % Zeroth harmonic (DC
component)
harmonic_1 = 2 * real(a_1 * exp(1j * omega0 * t));   % First harmonic
harmonic_2 = 2 * real(a_2 * exp(1j * 2 * omega0 * t)); % Second harmonic
harmonic_3 = 2 * real(a_3 * exp(1j * 3 * omega0 * t)); % Third harmonic


% Define finer time vector for smooth plots
t_fine = linspace(min(t), max(t), 1000);


% Interpolate each harmonic for smooth plotting
harmonic_0_fine = interp1(t, harmonic_0, t_fine, 'spline');
harmonic_1_fine = interp1(t, harmonic_1, t_fine, 'spline');
harmonic_2_fine = interp1(t, harmonic_2, t_fine, 'spline');
harmonic_3_fine = interp1(t, harmonic_3, t_fine, 'spline');


% Plot each harmonic
figure;


% Plot zeroth harmonic (DC component)
subplot(4,1,1);
plot(t_fine, harmonic_0_fine, 'LineWidth', 1.5, 'Color', [0.1, 0.5, 0.8]);
xlabel('Time (s)', 'FontSize', 10);
ylabel('Amplitude', 'FontSize', 10);
title('Zeroth Harmonic (DC Component)', 'FontSize', 12);
grid on;
set(gca, 'GridLineStyle', '--', 'FontSize', 10);
xlim([min(t_fine), max(t_fine)]);


% Plot first harmonic
subplot(4,1,2);
plot(t_fine, harmonic_1_fine, 'LineWidth', 1.5, 'Color', [0.8, 0.1, 0.3]);
xlabel('Time (s)', 'FontSize', 10);
ylabel('Amplitude', 'FontSize', 10);
title('First Harmonic', 'FontSize', 12);
grid on;
set(gca, 'GridLineStyle', '--', 'FontSize', 10);
xlim([min(t_fine), max(t_fine)]);


% Plot second harmonic
subplot(4,1,3);
plot(t_fine, harmonic_2_fine, 'LineWidth', 1.5, 'Color', [0.2, 0.6, 0.2]);
xlabel('Time (s)', 'FontSize', 10);
```

```
ylabel('Amplitude', 'FontSize', 10);
title('Second Harmonic', 'FontSize', 12);
grid on;
set(gca, 'GridLineStyle', '--', 'FontSize', 10);
xlim([min(t_fine), max(t_fine)]);


% Plot third harmonic
subplot(4,1,4);
plot(t_fine, harmonic_3_fine, 'LineWidth', 1.5, 'Color', [0.5, 0.2, 0.8]);
xlabel('Time (s)', 'FontSize', 10);
ylabel('Amplitude', 'FontSize', 10);
title('Third Harmonic', 'FontSize', 12);
grid on;
set(gca, 'GridLineStyle', '--', 'FontSize', 10);
xlim([min(t_fine), max(t_fine)]);


for i = 1:4
    ax = subplot(4,1,i);
    ax.Box = 'off';  % Turn off the box/frame around each plot
end
```

Code Listing A.5: Plot of the Half-Wave Rectifier

```
% Parameters
Ts = 0.1;                       % Sampling period
T = 18;                         % Period of the signal
active_duration = 4.5;          % Duration for the active half-wave rectified
cosine part
n = -40:319;                    % Range of n values
t = n * Ts;                     % Corresponding time values


% Define the discretized version of y_a(t)
y = zeros(size(t));  % Initialize the signal


% Calculate y[n] based on the periodic definition of y_a(t)
for i = 1:length(t)
    % Map t(i) into the range [0, T) to handle periodicity
    t_mod = mod(t(i), T);
    if t_mod < active_duration  % Within the active range [0, 4.5)
        y(i) = abs(5 * cos(pi * t_mod / 9));
    else
        y(i) = 0;  % Set to zero for the inactive part [4.5, 13.5)
    end
end
```

```
% Define a finer time vector for smooth plotting
t_fine = linspace(min(t), max(t), 1000);


% Interpolate the signal for smooth plotting
y_fine = interp1(t, y, t_fine, 'spline');


% Plot the interpolated signal y(t)
figure;
plot(t_fine, y_fine, 'LineWidth', 1.5, 'Color', [0.2, 0.6, 0.8]);
xlabel('Time (s)', 'FontSize', 12);
ylabel('Amplitude y(t)', 'FontSize', 12);
title('Half-Wave Rectified Cosine Signal y(t)', 'FontSize', 14);
grid on;
xlim([min(t_fine), max(t_fine)]);
ylim([0, 5.5]);
set(gca, 'GridLineStyle', '--', 'FontSize', 10);
ax = gca;
ax.Box = 'off';
```

Code Listing A.6: Half-Wave Rectifier Spectrum

```
% Parameters
T = 18;                         % Period of the signal
omega0 = 2 * pi / T;            % Fundamental frequency
max_k = 20;                     % Maximum harmonic order to calculate


% Range of k values for both positive and negative harmonics
k_values = -max_k:max_k;        % From -max_k to max_k
frequencies = k_values * omega0;  % Corresponding frequencies in rad/s
a_k = zeros(size(k_values));      % Array to store Fourier coefficients


% Calculate DC component (k = 0)
a_k(k_values == 0) = 5 / pi;  % This is a_0


% Calculate Fourier coefficients for k >= 1 and k <= -1
for k = 1:max_k
    % Using the derived formula for a_k for positive k
    a_k(k_values == k) = (5 / (2 * pi)) * ( ...
        sin(pi * (1 - k) / 2) / (1 - k) + ...
        sin(pi * (1 + k) / 2) / (1 + k));

    % Since the signal is real and even, a_{-k} = a_k for negative k
    a_k(k_values == -k) = a_k(k_values == k);
end


% Calculate magnitude and phase for plotting
a_k_magnitude = abs(a_k);            % Magnitude of a_k
```

```
a_k_phase = angle(a_k) * (180 / pi); % Phase of a_k in degrees


% Define finer frequency vector for smooth plotting
freq_fine = linspace(min(frequencies), max(frequencies), 1000);


% Interpolate magnitude and phase for smooth plotting
magnitude_continuous = interp1(frequencies, a_k_magnitude, freq_fine,
'spline');
phase_continuous = interp1(frequencies, a_k_phase, freq_fine, 'spline');


% Plotting the Magnitude Spectrum
figure;
subplot(2,1,1);
plot(freq_fine, magnitude_continuous, 'LineWidth', 1.5, 'Color', [0.8, 0.1,
0.3]);
xlabel('Frequency (rad/s)', 'FontSize', 12);
ylabel('|a_k|', 'FontSize', 12);
title('Magnitude Spectrum of y_a(t)', 'FontSize', 14);
grid on;
set(gca, 'GridLineStyle', '--', 'FontSize', 10);
xlim([-max(frequencies) - 2, max(frequencies) + 2]);


% Plotting the Phase Spectrum
subplot(2,1,2);
plot(freq_fine, phase_continuous, 'LineWidth', 1.5, 'Color', [0.2, 0.6, 0.8]);
xlabel('Frequency (rad/s)', 'FontSize', 12);
ylabel('Phase \angle a_k (degrees)', 'FontSize', 12);
title('Phase Spectrum of y_a(t)', 'FontSize', 14);
grid on;
set(gca, 'GridLineStyle', '--', 'FontSize', 10);
xlim([-max(frequencies) - 2, max(frequencies) + 2]);
```

Code Listing 7: Plots of $z_N[n]$

```
% Parameters
Ts = 0.1;                       % Sampling period
T = 18;                         % Period of the signal
omega0 = 2 * pi / T;            % Fundamental frequency
n = -40:319;                    % Range of n values
t = n * Ts;                     % Corresponding time values


% Range of N values for reconstruction
N_values = [150, 75, 30, 5, 3, 1];


% Maximum harmonic for calculation
```

```
max_k = max(N_values);


% Calculate Fourier coefficients for all k up to max_k
a_k = zeros(2 * max_k + 1, 1);  % Array to hold coefficients from -max_k to
+max_k
k_indices = -max_k:max_k;       % Range of k values


% Populate the Fourier coefficients using the general formula
for i = 1:length(k_indices)
    k = k_indices(i);
    if k == 0
        a_k(i) = 5 / pi;  % DC component
    else
        % General formula for non-zero k, with protection against division by
zero
        denom1 = 1 - k;
        denom2 = 1 + k;
        term1 = 0;
        term2 = 0;

        % Only calculate terms if denominators are non-zero
        if denom1 ~= 0
            term1 = sin(pi * denom1 / 2) / denom1;
        end
        if denom2 ~= 0
            term2 = sin(pi * denom2 / 2) / denom2;
        end

        a_k(i) = (5 / (2 * pi)) * (term1 + term2);
    end
end


% Loop over each N value and compute the truncated Fourier series
reconstruction
for idx = 1:length(N_values)
    N = N_values(idx);           % Current value of N
    z_N = zeros(size(n));         % Initialize the reconstructed signal for
this N

    % Sum the Fourier series components for the range k = -N to k = N
    for i = 1:length(k_indices)
        k = k_indices(i);
        if abs(k) <= N  % Only include terms up to the current N value
            % Find the Fourier coefficient for this k (adjust index to match
a_k)
            a_k_current = a_k(k + max_k + 1);  % Shift index for 1-based
indexing
            % Add the current Fourier component to z_N, handling any NaN
values
```

```
            z_N = z_N + a_k_current * cos(k * omega0 * t);
        end
    end

    % Define finer time vector for smooth plotting
    t_fine = linspace(min(t), max(t), 1000);
    z_N_fine = interp1(t, real(z_N), t_fine, 'spline');


    % Plot the reconstructed signal z_N(t)
    figure;
    plot(t_fine, z_N_fine, 'LineWidth', 1.5, 'Color', [0.2, 0.6, 0.8]);
    xlabel('Time (s)', 'FontSize', 12);
    ylabel('Amplitude z_N(t)', 'FontSize', 12);
    title(['Reconstructed Signal z_N(t) for N = ', num2str(N)], 'FontSize',
14);
    xlim([min(t_fine), max(t_fine)]);
    ylim([0, 5.5]);
    xticks(-4:2:32);
    yticks(0:1:5);
    grid on;
    set(gca, 'GridLineStyle', '--', 'FontSize', 10);
    ax = gca;
    ax.Box = 'off';
    hold off;
end
```

Code Listing 8: Plot of the Harmonics of the Half Wave Rectifier

```
% Parameters
Ts = 0.1;                       % Sampling period
T = 18;                         % Period of the signal
omega0 = 2 * pi / T;            % Fundamental frequency
n = -40:319;                    % Range of n values
t = n * Ts;                     % Corresponding time values


% Fourier coefficients for the first four harmonics
a_0 = 5 / pi;  % DC component
a_1 = (5 / (2 * pi)) * (sin(pi * (1 - 1) / 2) / (1 - 1) + sin(pi * (1 + 1) /
2) / (1 + 1));
a_2 = (5 / (2 * pi)) * (sin(pi * (1 - 2) / 2) / (1 - 2) + sin(pi * (1 + 2) /
2) / (1 + 2));
a_3 = (5 / (2 * pi)) * (sin(pi * (1 - 3) / 2) / (1 - 3) + sin(pi * (1 + 3) /
2) / (1 + 3));


% Handle cases where denominators would be zero
if isnan(a_1), a_1 = 0; end
if isnan(a_2), a_2 = 0; end
if isnan(a_3), a_3 = 0; end
```

```
% Individual harmonics calculation
harmonic_0 = a_0 * ones(size(t));               % Zeroth harmonic (DC
component)
harmonic_1 = a_1 * cos(1 * omega0 * t);         % First harmonic
harmonic_2 = a_2 * cos(2 * omega0 * t);         % Second harmonic
harmonic_3 = a_3 * cos(3 * omega0 * t);         % Third harmonic


% Define finer time vector for smooth plotting
t_fine = linspace(min(t), max(t), 1000);


% Interpolate each harmonic for smooth plotting
harmonic_0_fine = interp1(t, harmonic_0, t_fine, 'spline');
harmonic_1_fine = interp1(t, harmonic_1, t_fine, 'spline');
harmonic_2_fine = interp1(t, harmonic_2, t_fine, 'spline');
harmonic_3_fine = interp1(t, harmonic_3, t_fine, 'spline');


% Plot each harmonic
figure;


% Plot zeroth harmonic (DC component)
subplot(4,1,1);
plot(t_fine, harmonic_0_fine, 'LineWidth', 1.5, 'Color', [0.1, 0.5, 0.8]);
xlabel('Time (s)', 'FontSize', 10);
ylabel('Amplitude', 'FontSize', 10);
title('Zeroth Harmonic (DC Component)', 'FontSize', 12);
grid on;
set(gca, 'GridLineStyle', '--', 'FontSize', 10);
xlim([min(t_fine), max(t_fine)]);


% Plot first harmonic
subplot(4,1,2);
plot(t_fine, harmonic_1_fine, 'LineWidth', 1.5, 'Color', [0.8, 0.1, 0.3]);
xlabel('Time (s)', 'FontSize', 10);
ylabel('Amplitude', 'FontSize', 10);
title('First Harmonic', 'FontSize', 12);
grid on;
set(gca, 'GridLineStyle', '--', 'FontSize', 10);
xlim([min(t_fine), max(t_fine)]);


% Plot second harmonic
subplot(4,1,3);
plot(t_fine, harmonic_2_fine, 'LineWidth', 1.5, 'Color', [0.2, 0.6, 0.2]);
xlabel('Time (s)', 'FontSize', 10);
ylabel('Amplitude', 'FontSize', 10);
```

```
title('Second Harmonic', 'FontSize', 12);
grid on;
set(gca, 'GridLineStyle', '--', 'FontSize', 10);
xlim([min(t_fine), max(t_fine)]);


% Plot third harmonic
subplot(4,1,4);
plot(t_fine, harmonic_3_fine, 'LineWidth', 1.5, 'Color', [0.5, 0.2, 0.8]);
xlabel('Time (s)', 'FontSize', 10);
ylabel('Amplitude', 'FontSize', 10);
title('Third Harmonic', 'FontSize', 12);
grid on;
set(gca, 'GridLineStyle', '--', 'FontSize', 10);
xlim([min(t_fine), max(t_fine)]);


for i = 1:4
    ax = subplot(4,1,i);
    ax.Box = 'off';
end
```