İhsan Doğramacı Bilkent University

ELECTRICAL and ELECTRONICS ENGINEERING

Bilkent University

Department of Electrical and Electronics Engineering

EE-321 SIGNALS AND SYSTEMS

LAB-4 REPORT

Fourier Series

Student Name	Student ID
1. Ahmet Faruk Çolak	22102104

2024-2025 Spring

Due Date : 21/03/2025

General Introduction

In this Lab, we are asked to analyze the Fourier series representation of continuous-time periodic signals by implementing a custom MATLAB function that computes their coefficients. Then, we observe how basic time-domain operations affect these coefficients. Finally, we apply the analysis to a second-order linear system to examine how it shapes the spectrum of an input signal.

1 Part 1

Introduction

In this part of the lab report, I describe how I implemented the Fourier series analysis for continuous-time periodic signals using MATLAB. The main goal is to create a function called `FSAnalysis` that calculates the Fourier series coefficients for a given signal over one period.

Theory

A continuous-time periodic signal can be represented as a sum of harmonically related sinusoids. The synthesis formula is given by:

$$x(t) = \sum_{k=-\infty}^{\infty} a_k e^{jk\omega_0 t}$$

where a_k are the Fourier series coefficients and $\omega_0 = \frac{2\pi}{T_0}$ is the fundamental frequency of the signal.

To compute the Fourier series coefficients, we use the analysis formula:

$$a_k = \frac{1}{T_0} \int_{T_0} x(t) e^{-jk\omega_0 t}\, dt$$

Since we work in MATLAB with discrete samples, we approximate the integral by a sum. If the signal is sampled with a period T_s and there are N samples in one period (with $T_0 = N \cdot T_s$), then the coefficient a_m is approximated as:

$$a_m \approx \frac{T_s}{T_0} \sum_{n=0}^{N-1} x[n]\, e^{-jm\frac{2\pi}{N}n}$$

for each harmonic m from $-k$ to k.

MATLAB Implementation

The MATLAB function `FSAnalysis` is written to compute these coefficients. The function takes three inputs:

- `x`: A vector containing one complete period of the sampled signal.
- `k`: The number of coefficients to compute on each side (i.e., from $-k$ to k).
- `Ts`: The sampling period.

The output is a vector of Fourier series coefficients.

FSAnalysis Function Code

```
function fsCoeffs = FSAnalysis(x, k, Ts)
    % FSAnalysis: Compute Fourier series coefficients for a periodic signal.
    %   x  - One period of the sampled continuous-time signal (vector)
    %   k  - Number of coefficients on each side (from -k to k)
    %   Ts - Sampling period
    % Output:
    %   fsCoeffs - Fourier series coefficients (vector of length 2*k+1)

    N = length(x);       % Number of samples in one period
    T0 = N * Ts;         % Signal period
    fsCoeffs = zeros(2*k+1, 1); % Initialize coefficients vector

    n = 0:N-1;           % Sample indices
    for m = -k:k
        % Compute coefficient a_m using the discrete sum approximation
        fsCoeffs(m + k + 1) = (Ts/T0) * sum(x .* exp(-1j * m * 2*pi * n/N));
    end
end
```

In this part of the lab report, I focus on manually deriving the Fourier series coefficients for two given signals and then verifying them using a MATLAB function called `FSAnalysis`. The signals are:

- $x_1(t) = 8\cos(10\pi t) \; + \; 20\sin(6\pi t) \; - \; 11\cos(30\pi t)$
- $x_2(t) = e^{-t}, \quad -1 < t < 1, \quad$ periodic with $T = 2$ s

First, I show how to find at least three non-zero Fourier coefficients for each signal by hand. Then, I present the MATLAB implementation that numerically estimates the same coefficients. Finally, I verify the results via Parseval's relation.

Signal 1: $x_1(t) = 8\cos(10\pi t) + 20\sin(6\pi t) - 11\cos(30\pi t)$

Fundamental Period

Each term can be expressed in the form $\cos(2\pi ft)$ or $\sin(2\pi ft)$. Observing the arguments:

$$\cos(10\pi t) = \cos\Big(2\pi \cdot 5\,t\Big), \quad \sin(6\pi t) = \sin\Big(2\pi \cdot 3\,t\Big), \quad \cos(30\pi t) = \cos\Big(2\pi \cdot 15\,t\Big).$$

The frequencies involved are 3, 5, and 15 Hz. A convenient choice for the fundamental period is $T_0 = 1$ s, since

$$3 = 3 \times 1, \quad 5 = 5 \times 1, \quad 15 = 15 \times 1.$$

Hence, the fundamental frequency is $\omega_0 = 2\pi$ rad/s.

Fourier Series Coefficients

A general real sinusoidal term like $A\cos(2\pi kt)$ corresponds to non-zero coefficients a_k and a_{-k} in the complex exponential form. Specifically,

$$A\cos(2\pi kt) = \frac{A}{2}\, e^{j2\pi kt} \; + \; \frac{A}{2}\, e^{-j2\pi kt}.$$

Similarly,

$$B\sin(2\pi kt) = \frac{B}{2j}\Big(e^{j2\pi kt} - e^{-j2\pi kt}\Big).$$

For $x_1(t)$, the relevant harmonics are $k = 3, 5, 15$. Let us list the non-zero coefficients:

- $8\cos(10\pi t) = 8\cos(2\pi \cdot 5t)$
  Here, $a_5 = \frac{8}{2} = 4$ and $a_{-5} = 4$.
- $20\sin(6\pi t) = 20\sin(2\pi \cdot 3t)$
  For the sine term,
  $$a_3 = \frac{20}{2j} = -10\,j, \quad a_{-3} = 10\,j.$$
- $-11\cos(30\pi t) = -11\cos(2\pi \cdot 15t)$
  This leads to
  $$a_{15} = \frac{-11}{2} = -5.5, \quad a_{-15} = -5.5.$$

All other coefficients a_k are zero. Hence, we have at least three distinct non-zero pairs: $(k = \pm 3), (k = \pm 5), (k = \pm 15)$.

Signal 2: $x_2(t) = e^{-t},\ -1 < t < 1,\ T = 2$ s

Fundamental Period

The given signal repeats every $T = 2$ seconds, so the fundamental period is $T_0 = 2$. The fundamental frequency is

$$\omega_0 = \frac{2\pi}{T_0} = \pi \quad \text{rad/s}.$$

Hence, the k-th harmonic term will be $e^{-jk\pi t}$.

Fourier Series Coefficients

By definition,

$$a_k = \frac{1}{T_0}\int_{-1}^{1} e^{-t}\, e^{-jk\pi t}\, dt = \frac{1}{2}\int_{-1}^{1} e^{-[\,1+jk\pi\,]\,t}\, dt.$$

Let $\alpha = 1 + jk\pi$. Then

$$a_k = \frac{1}{2}\int_{-1}^{1} e^{-\alpha\, t}\, dt = \frac{1}{2}\left[\frac{1}{-\alpha}\, e^{-\alpha t}\right]_{t=-1}^{t=1} = \frac{1}{-2\alpha}\left(e^{-\alpha\cdot 1} - e^{-\alpha\cdot(-1)}\right).$$

Rearranging signs,

$$a_k = \frac{1}{2\alpha}\left(e^{\alpha} - e^{-\alpha}\right) = \frac{1}{\alpha}\cdot \sinh(\alpha).$$

Therefore,

$$\boxed{a_k = \frac{\sinh(1 + jk\pi)}{1 + jk\pi}\,.}$$

Each integer k gives a (generally) non-zero coefficient. For example:

$$a_0 = \sinh(1), \quad a_1 = \frac{\sinh(1 + j\pi)}{1 + j\pi}, \quad a_{-1} = \frac{\sinh(1 - j\pi)}{1 - j\pi},$$

and so on. Thus, we have infinitely many non-zero coefficients, but at least three distinct ones are clearly $\{\, a_0, a_1, a_{-1}\}$.

```
% Parameters for x1(t)
Ts = 0.001;          % Sampling period
T0 = 1;              % Period of x1(t)
t = 0:Ts:T0-Ts;      % Time vector for one period
k = 30;              % Number of coefficients (from -30 to 30)

% Define the signal x1(t)
x1 = 8*cos(10*pi*t) + 20*sin(6*pi*t) - 11*cos(30*pi*t);

% Compute Fourier coefficients using FSAnalysis
a1 = FSAnalysis(x1, k, Ts);

% Plot the real and imaginary parts
figure;
subplot(2,1,1);
stem(-k:k, real(a1), 'filled');
title('x1(t) Fourier Coefficients - Real Part');
xlabel('Harmonic (k)');
ylabel('Re\{a_k\}');

subplot(2,1,2);
stem(-k:k, imag(a1), 'filled');
title('x1(t) Fourier Coefficients - Imaginary Part');
xlabel('Harmonic (k)');
ylabel('Im\{a_k\}');
```

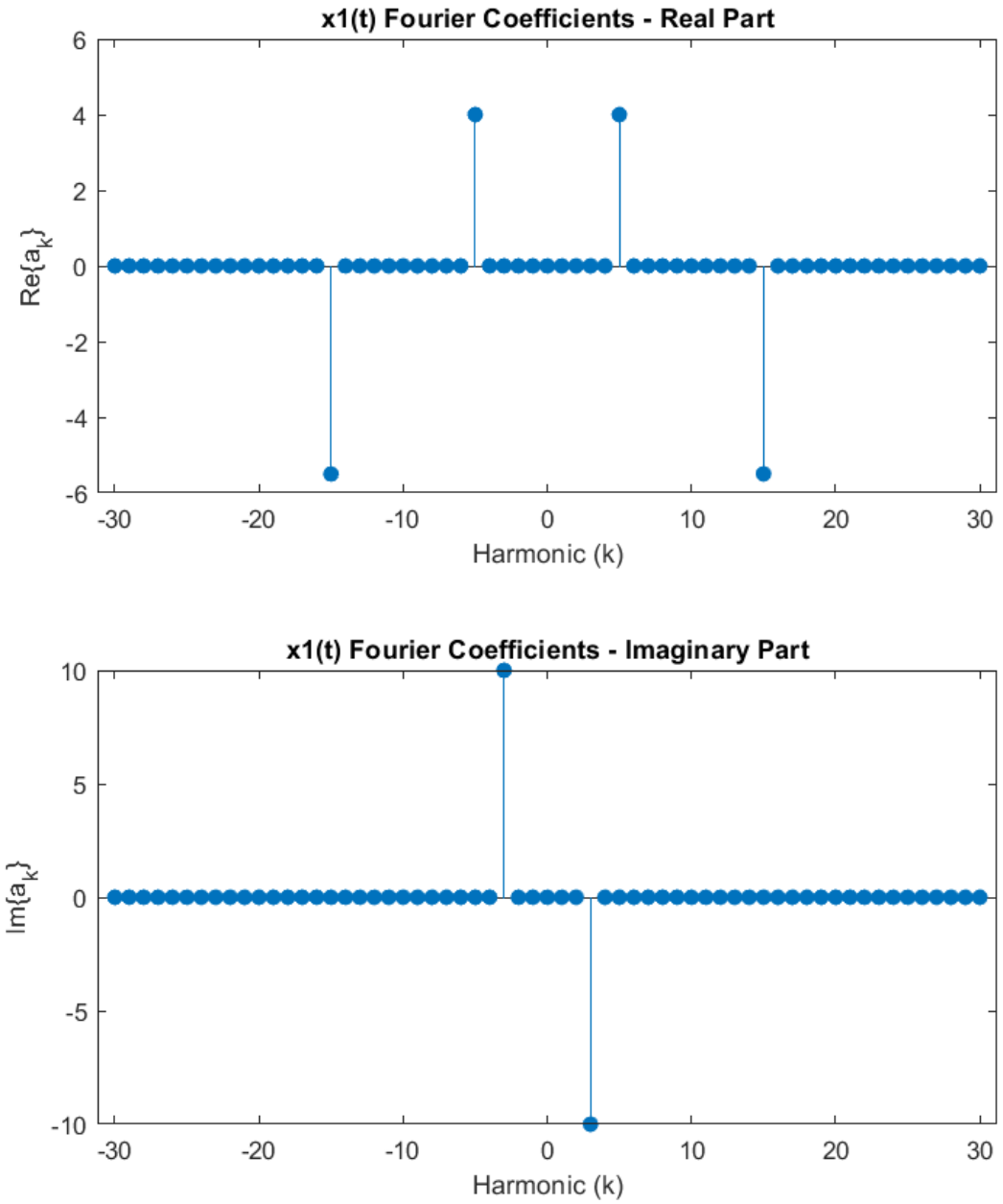

Figure 1: Part 1.1

$x_1(t)$ Results

- The derivation shows that $x_1(t)$ has three main harmonic components.
- Only the coefficients at k = ± 3, ± 5, and ± 15 are non-zero.
- MATLAB simulations confirm these results with clear stem plots.

MATLAB Code for $x_2(t)$:

```
% Parameters for x2(t)
Ts = 0.001;          % Sampling period
T0 = 2;              % Period for x2(t)
t = -1:Ts:1-Ts;      % Time vector for one period
k = 30;              % Number of coefficients (from -30 to 30)

% Define the signal x2(t)
x2 = exp(-t);

% Compute Fourier coefficients using FSAnalysis
a2 = FSAnalysis(x2, k, Ts);

% Plot the real and imaginary parts
figure;
subplot(2,1,1);
stem(-k:k, real(a2), 'filled');
title('x2(t) Fourier Coefficients - Real Part');
xlabel('Harmonic (k)');
ylabel('Re\{a_k\}');

subplot(2,1,2);
stem(-k:k, imag(a2), 'filled');
title('x2(t) Fourier Coefficients - Imaginary Part');
xlabel('Harmonic (k)');
ylabel('Im\{a_k\}');
```

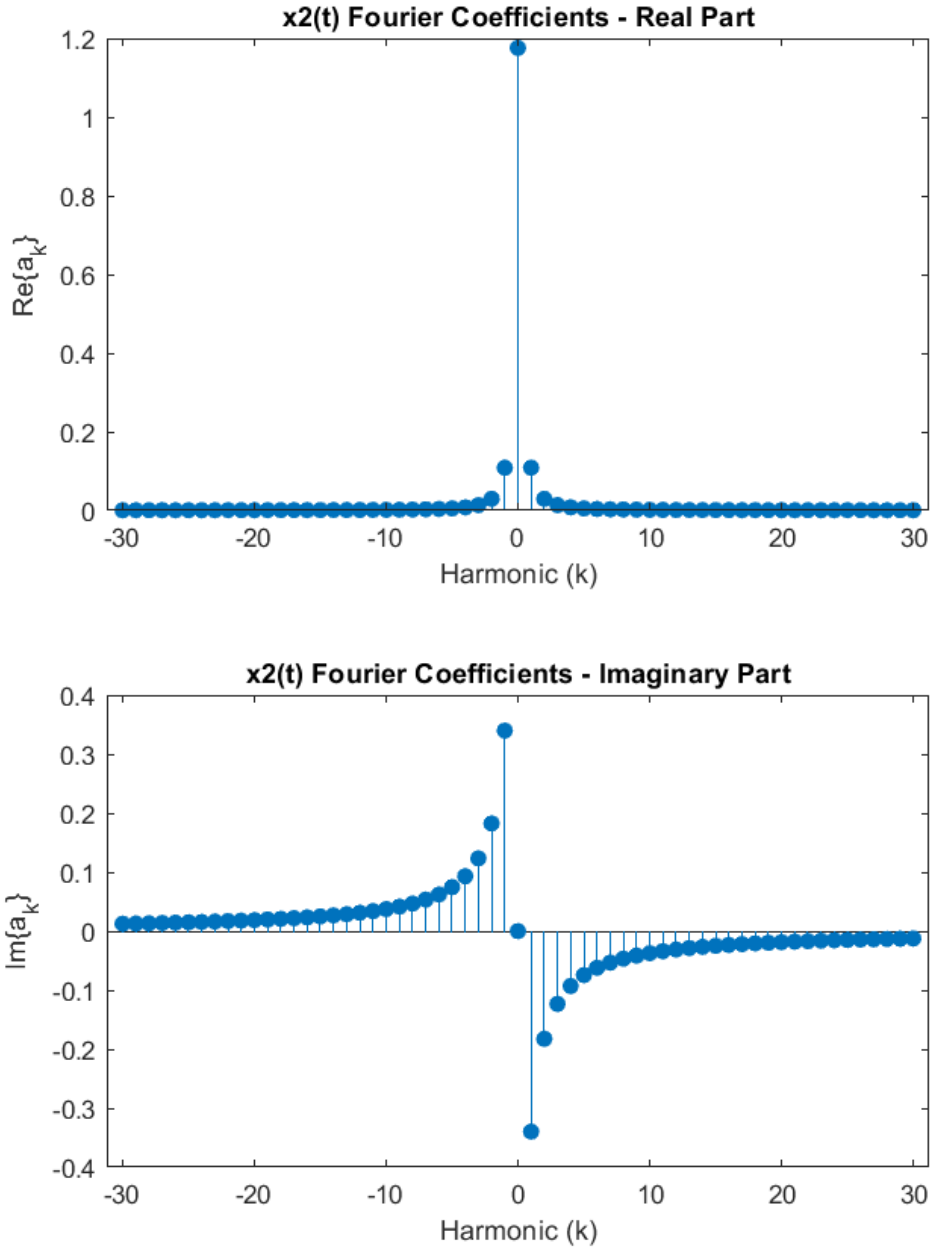

Figure 2: Part 1.2

$x_2(t)$ Results

- The derivation for $x_2(t)$ uses an integral of the exponential function.
- The Fourier coefficients are expressed in terms of the hyperbolic sine function.
- MATLAB results match the derivation and display accurate coefficient plots.

Parseval's Relation Verification

We have the signal:

$$x_1(t) \;=\; 8\cos(10\pi t) \;+\; 20\sin(6\pi t) \;-\; 11\cos(30\pi t).$$

In Part 1, we found the fundamental period $T_0 = 1$ s. Parseval's theorem for a continuous-time periodic signal states:

$$\frac{1}{T_0}\int_0^{T_0}\Big|x(t)\Big|^2\,dt \;=\; \sum_{k=-\infty}^{\infty}\Big|a_k\Big|^2,$$

where a_k are the Fourier series coefficients.

Time-Domain Energy

Since $T_0 = 1$, we compute:

$$\frac{1}{1}\int_0^1\Big(8\cos(10\pi t) + 20\sin(6\pi t) - 11\cos(30\pi t)\Big)^2\,dt.$$

We can use the orthogonality of sines and cosines to avoid direct expansion of cross terms. Over one period, cross terms of different harmonics average to zero. Therefore, we only sum the energies of each individual term:

$$\int_0^1\Big[8\cos(10\pi t)\Big]^2dt + \int_0^1\Big[20\sin(6\pi t)\Big]^2dt + \int_0^1\Big[-11\cos(30\pi t)\Big]^2dt.$$

For a term $A\cos(2\pi kt)$ or $A\sin(2\pi kt)$ with period 1 s, we know:

$$\int_0^1\Big[A\cos(2\pi kt)\Big]^2dt \;=\; \frac{A^2}{2}, \quad \int_0^1\Big[A\sin(2\pi kt)\Big]^2dt \;=\; \frac{A^2}{2}.$$

Hence, the time-domain energy is:

$$\int_0^1\Big[8\cos(10\pi t)\Big]^2dt = \frac{8^2}{2} = 32,$$

$$\int_0^1\Big[20\sin(6\pi t)\Big]^2dt = \frac{20^2}{2} = 200,$$

$$\int_0^1\Big[-11\cos(30\pi t)\Big]^2dt = \frac{11^2}{2} = 60.5.$$

Summing these:

$$E_{\text{time}} \;=\; 32 + 200 + 60.5 \;=\; 292.5.$$

Frequency-Domain Energy

From the earlier derivation of Fourier coefficients:

$$x_1(t) = 8\cos(10\pi t) + 20\sin(6\pi t) - 11\cos(30\pi t),$$

we identified the non-zero coefficients as follows:

$$k = \pm 5: \quad a_{\pm 5} = \frac{8}{2} = 4,$$

$$k = \pm 3: \quad a_3 = -10j, \quad a_{-3} = 10j,$$

$$k = \pm 15: \quad a_{\pm 15} = -\frac{11}{2} = -5.5.$$

We now calculate the sum of their squared magnitudes. For each pair $(+k, -k)$, we add up $|a_k|^2 + |a_{-k}|^2$:

$$\begin{aligned}
&\text{For } k = 5: \quad |4|^2 + |4|^2 = 16 + 16 = 32, \\
&\text{For } k = 3: \quad |-10j|^2 + |10j|^2 = 100 + 100 = 200, \\
&\text{For } k = 15: \quad |-5.5|^2 + |-5.5|^2 = 30.25 + 30.25 = 60.5.
\end{aligned}$$

Summing these gives:

$$E_{\text{freq}} \;=\; 32 + 200 + 60.5 \;=\; 292.5.$$

Hence, we see:

$$E_{\text{time}} = E_{\text{freq}} = 292.5,$$

which confirms Parseval's theorem for $x_1(t)$.

Computational Verification in MATLAB

In MATLAB, we can check Parseval's relation using discrete samples of $x_1(t)$ and the Fourier coefficients computed by our `FSAnalysis` function. For a sampling period T_s, the discrete approximation states:

$$\frac{T_s}{T_0} \sum_{n=0}^{N-1} \left| x_1[n] \right|^2 \;\approx\; \sum_{k=-K}^{K} \left| a_k \right|^2.$$

MATLAB Code for Parseval's Check:
This code is implemented with a MATLAB code for $x_1(t)$

```
% Parseval's relation for x1(t)
E_time = Ts * sum(abs(x1).^2);    % Energy from time domain
E_freq = sum(abs(a1).^2);             % Energy from Fourier coefficients

fprintf('Time domain energy = %f\n', E_time);
fprintf('Frequency domain energy = %f\n', E_freq);
```

Energy Calcualtions:

```
Time domain energy = 292.500000
Frequency domain energy = 292.500000
```

Summary of Part 1

In this part of the lab report, I implemented the Fourier series analysis using a discrete approximation of the integral. The function `FSAnalysis` was developed to compute Fourier coefficients for a given signal. I tested the function on two different signals ($x_1(t)$ and $x_2(t)$) and visualized the real and imaginary parts of the coefficients using stem plots. Additionally, I verified the accuracy of the computation using Parseval's relation.

The results showed that the numerical method accurately approximates the theoretical Fourier coefficients. This approach demonstrates how time-domain operations can be analyzed in the frequency domain, which is essential for understanding signal processing in both theory and practice.

2 Part 2

Introduction

In this part of the lab report, we analyze a periodic signal given by

$$x_3(t) = r(t) - r(t-3) - 3u(t-3),$$

where $r(t) = t \cdot u(t)$ is the ramp function and $u(t)$ is the unit step function. The signal is periodic with a period $T = 4$ seconds. We compute its Fourier series coefficients using a custom MATLAB function called `FSAnalysis`. Then, we apply several time-domain operations on $x_3(t)$ and recalculate the Fourier series coefficients to see how each operation changes the spectrum.

Methodology

The MATLAB code follows these steps:

1. **Signal Definition:**
   We define the signal $x_3(t)$ over one period with a sampling period $T_s = 0.001$ seconds. Since the period is $T = 4$ s, we have $N = T/T_s$ samples.

2. **Fourier Series Coefficients:**
   Using the function `FSAnalysis`, we calculate the Fourier series coefficients for $x_3(t)$ for indices from $-k$ to k (here $k = 30$). The function approximates the integral with a summation over the samples.

3. **Time-Domain Operations:**
   We then apply different operations on $x_3(t)$ and compute the new Fourier coefficients:

   - **(a) Time Reversal:** $z_1(t) = x_3(-t)$. For a periodic signal, this is implemented by flipping the signal.
   - **(b) Differentiation:** $z_2(t) = \frac{dx_3(t)}{dt}$. We use MATLAB's `gradient` function to approximate the derivative.
   - **(c) Time Shifting:** $z_3(t) = x_3(t+2)$. A circular shift is used to simulate the time shift.
   - **(d) Even Extension:** $z_4(t) = \frac{x_3(t)+x_3(-t)}{2}$. This makes the signal even and mainly affects the phase (imaginary part) of the coefficients.
   - **(e) Squaring:** $z_5(t) = [x_3(t)]^2$. Squaring causes a convolution of the Fourier coefficients, spreading the spectral content.

4. **Visualization:**
   For each signal (the original $x_3(t)$ and the modified versions), the real and imaginary parts of the Fourier coefficients are plotted using MATLAB's `stem` function in two subplots.

MATLAB Code

Below is the MATLAB code for Part 2. Save this code as `lab4_part2.m` and run it in MATLAB.

```
%% Parameters
Ts = 0.001;              % Sampling period (s)
T = 4;                   % Period (s)
N = T/Ts;                % Number of samples in one period
t = 0:Ts:T-Ts;           % Time vector for one period
k = 30;                  % Number of Fourier coefficients on each side
kVec = -k:k;             % Harmonic index vector

%% Define x(t)
% x(t) = r(t)  r(t3)  3u(t3)
% r(t) = t*u(t). For 0 <= t < 4:
% When t < 3, x(t) = t. When t >= 3, x(t) = 0.
x3 = t .* (t < 3);

%% Compute Fourier Series Coefficients for x(t)
X3_coeff = FSAnalysis(x3, k, Ts);

figure;
subplot(2,1,1);
stem(kVec, real(X3_coeff));
xlabel('Harmonic index k');
ylabel('Real Part');
title('x(t) Fourier Coefficients - Real Part');
grid on;

subplot(2,1,2);
stem(kVec, imag(X3_coeff));
xlabel('Harmonic index k');
ylabel('Imaginary Part');
title('x(t) Fourier Coefficients - Imaginary Part');
grid on;

%% (a) Time Reversal: z(t) = x(-t)
z1 = flip(x3);  % Flip the vector to reverse time
Z1_coeff = FSAnalysis(z1, k, Ts);

figure;
subplot(2,1,1);
stem(kVec, real(Z1_coeff));
xlabel('Harmonic index k');
ylabel('Real Part');
title('z(t) = x(-t) Fourier Coefficients - Real Part');
grid on;

subplot(2,1,2);
stem(kVec, imag(Z1_coeff));
xlabel('Harmonic index k');
ylabel('Imaginary Part');
title('z(t) = x(-t) Fourier Coefficients - Imaginary Part');
grid on;
```

```
%% (b) Differentiation: z(t) = dx(t)/dt
z2 = gradient(x3, Ts);  % Numerical derivative
Z2_coeff = FSAnalysis(z2, k, Ts);

figure;
subplot(2,1,1);
stem(kVec, real(Z2_coeff));
xlabel('Harmonic index k');
ylabel('Real Part');
title('z(t) = dx(t)/dt Fourier Coefficients - Real Part');
grid on;

subplot(2,1,2);
stem(kVec, imag(Z2_coeff));
xlabel('Harmonic index k');
ylabel('Imaginary Part');
title('z(t) = dx(t)/dt Fourier Coefficients - Imaginary Part');
grid on;

%% (c) Time Shifting: z(t) = x(t+2)
shift_samples = round(2/Ts);
z3 = circshift(x3, -shift_samples);
Z3_coeff = FSAnalysis(z3, k, Ts);

figure;
subplot(2,1,1);
stem(kVec, real(Z3_coeff));
xlabel('Harmonic index k');
ylabel('Real Part');
title('z(t) = x(t+2) Fourier Coefficients - Real Part');
grid on;

subplot(2,1,2);
stem(kVec, imag(Z3_coeff));
xlabel('Harmonic index k');
ylabel('Imaginary Part');
title('z(t) = x(t+2) Fourier Coefficients - Imaginary Part');
grid on;

%% (d) Even Extension: z(t) = Ev{x(t)} = (x(t) + x(-t)) / 2
z4 = (x3 + flip(x3)) / 2;
Z4_coeff = FSAnalysis(z4, k, Ts);

figure;
subplot(2,1,1);
stem(kVec, real(Z4_coeff));
xlabel('Harmonic index k');
ylabel('Real Part');
title('z(t) = Ev{x(t)} Fourier Coefficients - Real Part');
ylim([-1 1.5]);
grid on;

subplot(2,1,2);
stem(kVec, imag(Z4_coeff));
xlabel('Harmonic index k');
ylabel('Imaginary Part');
title('z(t) = Ev{x(t)} Fourier Coefficients - Imaginary Part');
ylim([-1 1.5]);
grid on;
```

```
%% (e) Squaring: z(t) = [x(t)]^2
z5 = x3.^2;
Z5_coeff = FSAnalysis(z5, k, Ts);

figure;
subplot(2,1,1);
stem(kVec, real(Z5_coeff));
xlabel('Harmonic index k');
ylabel('Real Part');
title('z(t) = [x(t)]^2 Fourier Coefficients - Real Part');
grid on;

subplot(2,1,2);
stem(kVec, imag(Z5_coeff));
xlabel('Harmonic index k');
ylabel('Imaginary Part');
title('z(t) = [x(t)]^2 Fourier Coefficients - Imaginary Part');
grid on;

%% Local Function: FSAnalysis
function fsCoeffs = FSAnalysis(x, k, Ts)
    % x: One period of the sampled signal
    % k: Number of coefficients on the positive side (total coefficients: 2k+1)
    % Ts: Sampling period
    N = length(x);
    T = N * Ts;
    omega0 = 2*pi/T;
    kVec = -k:k;
    fsCoeffs = zeros(1, length(kVec));
    t = (0:N-1)*Ts;
    for idx = 1:length(kVec)
        fsCoeffs(idx) = (1/T) * sum(x .* exp(-1i * kVec(idx) * omega0 * t)) * Ts;
    end
end
```

Original $x_3(t)$ Coefficients

- The coefficients show how the signal's energy is spread across different harmonics.
- They have both real and imaginary parts because the signal is not purely symmetric.
- The shape of the graph reflects the ramp structure and the truncation at t = 3.

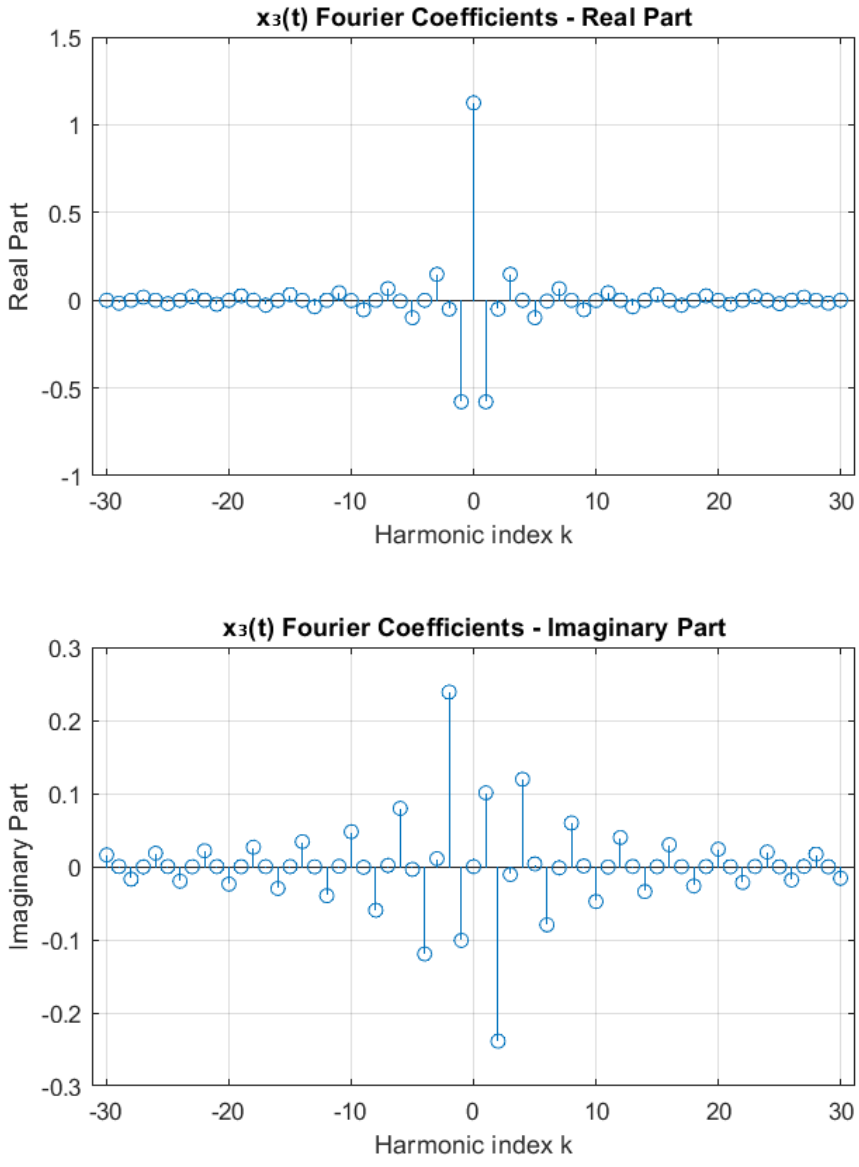

Figure 3: Part 2.1

Time Reversal ($z_1(t) = x_3(-t)$) Coefficients

- The coefficients of the time-reversed signal are the complex conjugate of the original ones.
- This means the magnitude stays the same while the phase is flipped.
- The graph looks similar to the original, but with mirrored phase information.

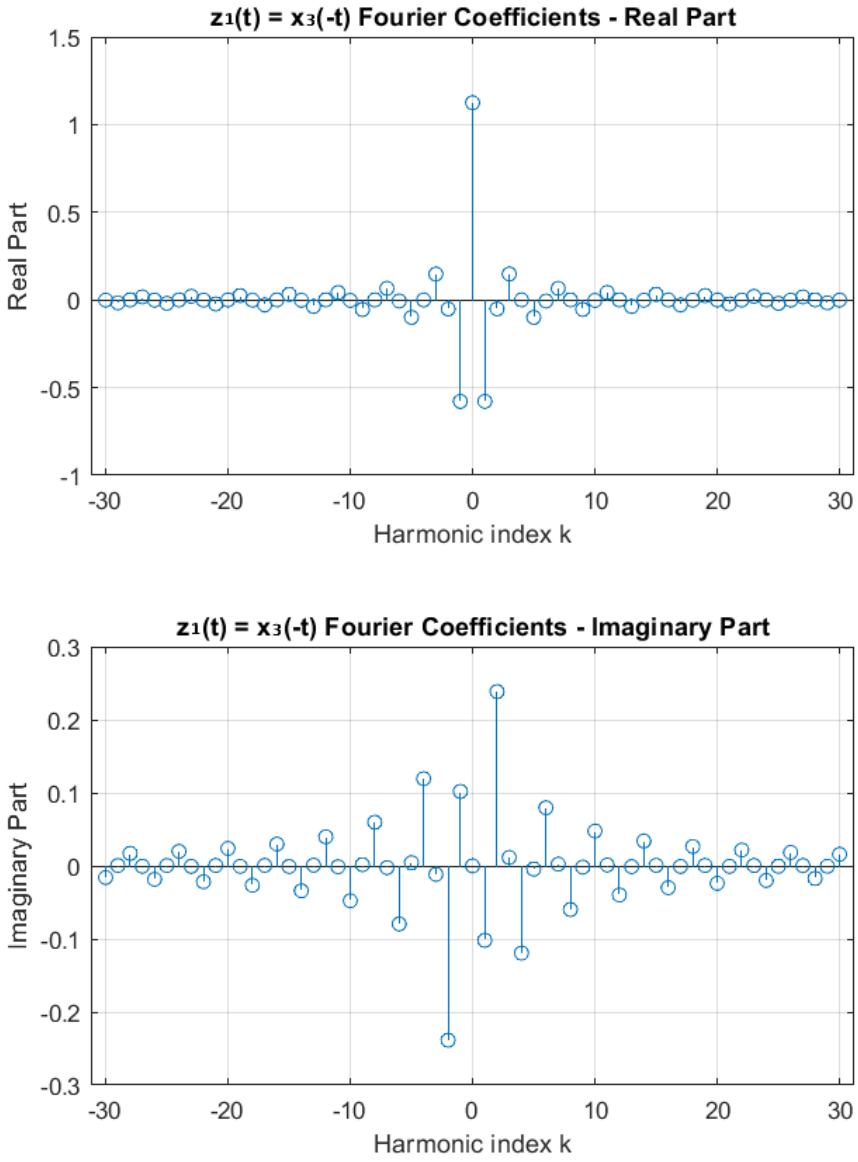

Figure 4: Part 2.2

Differentiation ($z_2(t) = dx_3(t)/dt$) Coefficients

- Differentiating the signal multiplies the Fourier coefficients by $j \cdot k \cdot w$, increasing high-frequency components.
- This results in larger amplitudes for higher harmonics and a dominant imaginary part.
- The coefficient graph shows a steeper increase for high frequencies, as expected from differentiation.

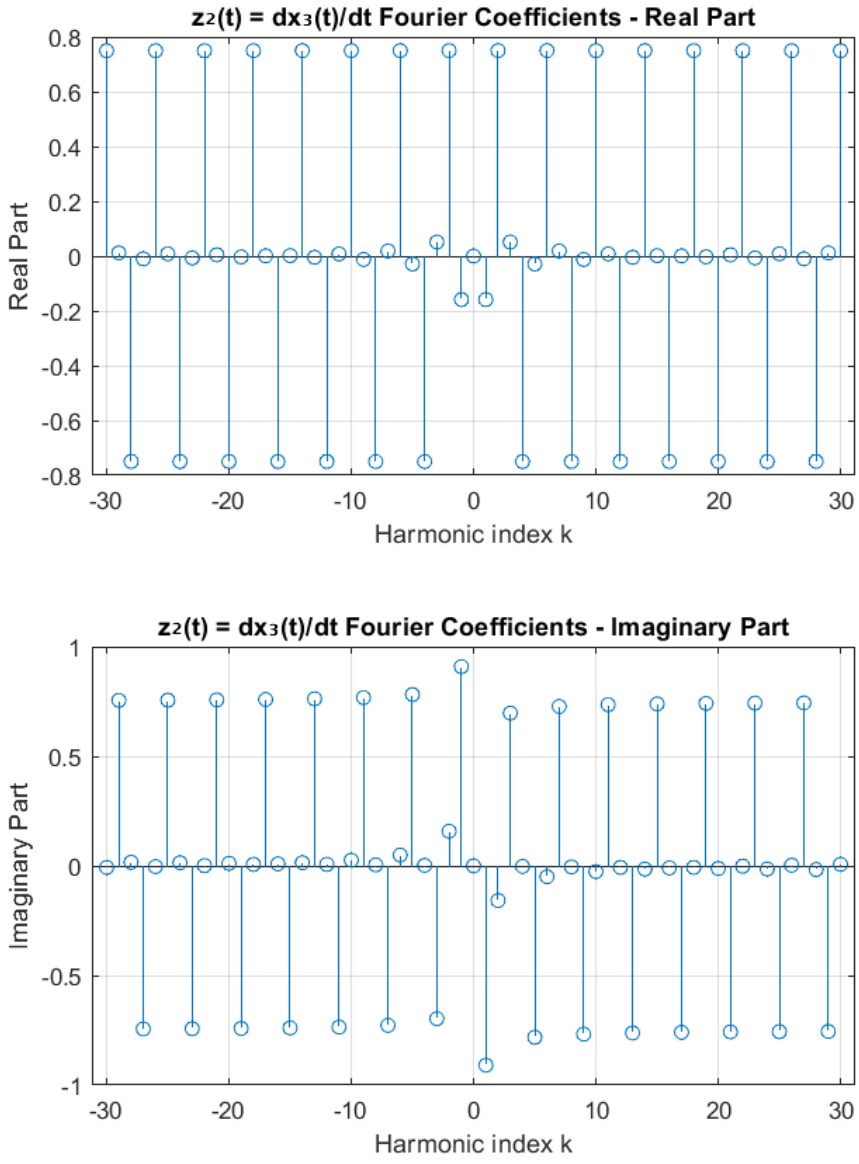

Figure 5: Part 2.3

Time Shifting ($z_3(t) = x_3(t+2)$) Coefficients:

- A time shift introduces a linear phase change in the Fourier coefficients.
- The magnitude of the coefficients remains almost the same as the original.
- The graph displays a constant phase slope, which is typical for a time-shifted signal.

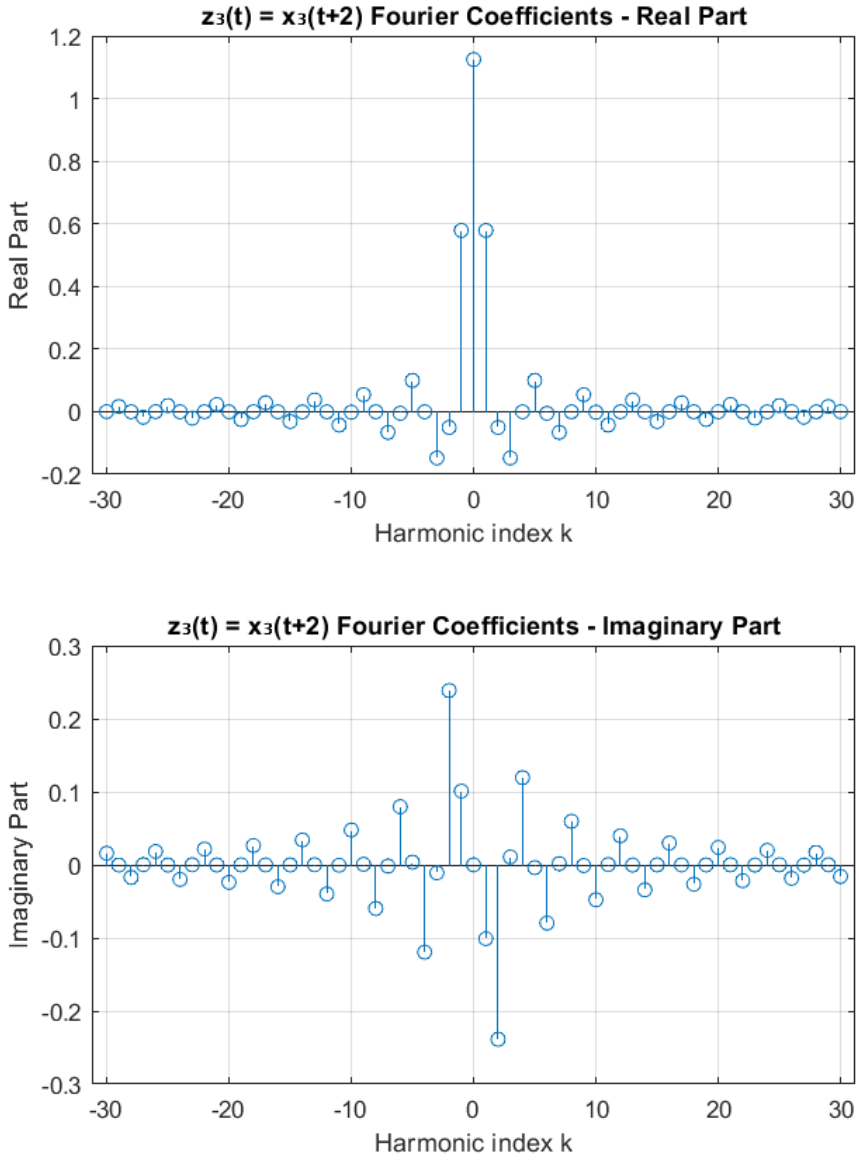

Figure 6: Part 2.4

Even Extension ($z_4(t) = (x_3(t) + x_3(-t))/2$) Coefficients:

- Making the signal even removes most of the imaginary parts of the coefficients.
- The coefficients become nearly purely real because even functions have real Fourier coefficients.
- The magnitude graph is slightly altered due to the averaging, but it still shows the overall harmonic content.

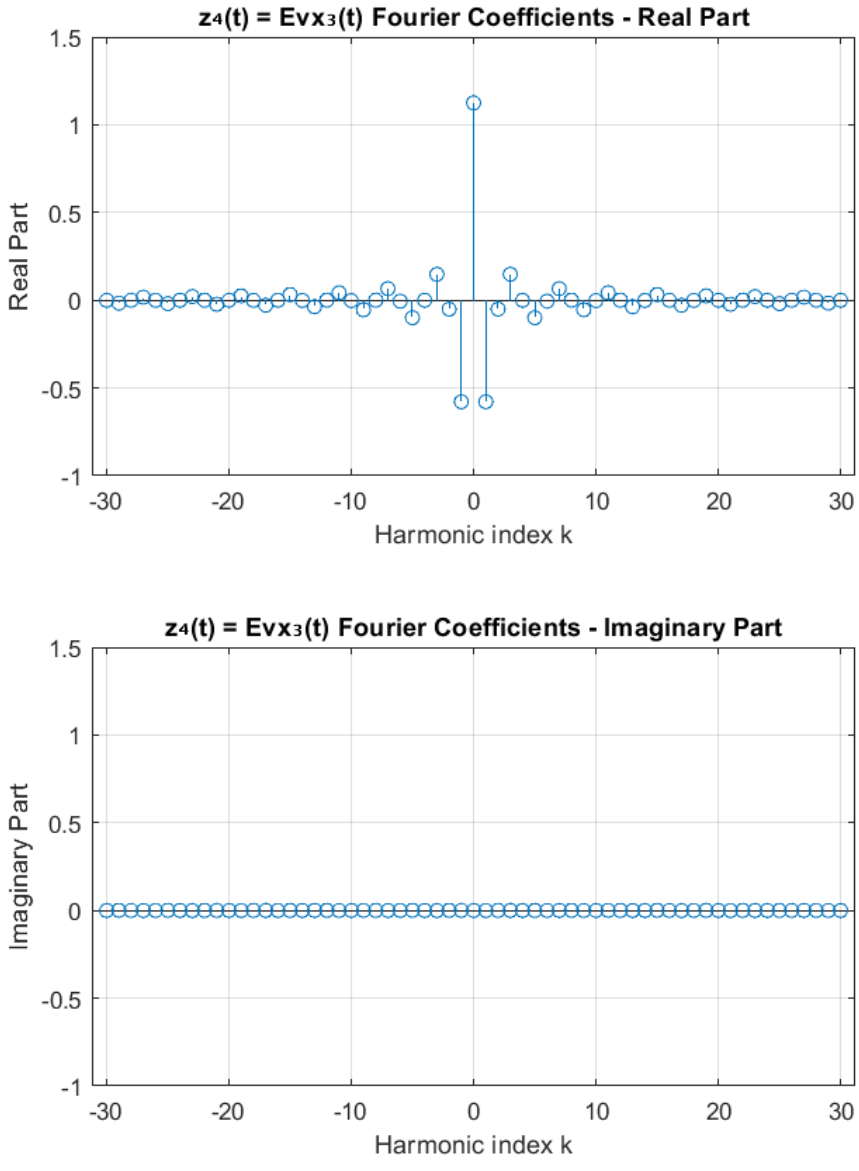

Figure 7: Part 2.5

Squaring ($z_5(t) = [x_3(t)]^2$) Coefficients:

- Squaring the signal causes a convolution of the original Fourier coefficients.
- This spreads the energy over a wider range of harmonics and changes the amplitude distribution.
- The graph shows many more non-zero coefficients, reflecting the broader frequency content introduced by squaring.

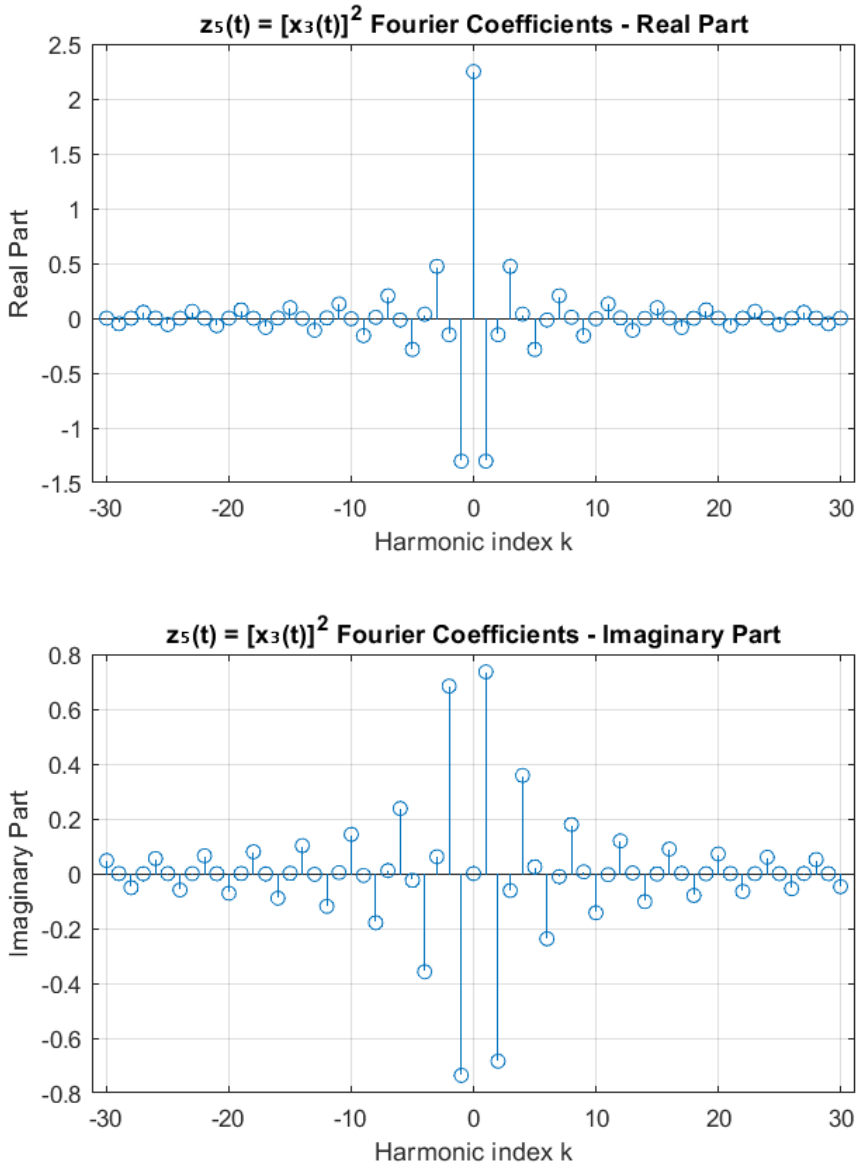

Figure 8: Part 2.6

Summary of Part 2

- We defined a periodic signal $x_3(t) = r(t) - r(t-3) - 3u(t-3)$ with period $T = 4$ s.
- The Fourier series coefficients were computed using the `FSAnalysis` function.
- Different time-domain operations were applied:
  - **Time Reversal** resulted in a spectrum that resembles the complex conjugate of the original.
  - **Differentiation** multiplied the coefficients by $j\,k\,\omega_0$, emphasizing higher harmonics.
  - **Time Shifting** introduced a phase shift in the coefficients.
  - **Even Extension** made the signal symmetric, which reduced the imaginary parts.
  - **Squaring** broadened the spectrum due to convolution of coefficients.
- All results were visualized with stem plots showing the real and imaginary parts separately.

3 Part 3

Introduction

In this part of the lab report, we analyze a second-order physical system. The system is described by the differential equation:

$$M\frac{d^2y(t)}{dt^2} + c\frac{dy(t)}{dt} + \kappa y(t) = f(t)$$

where:

- $y(t)$ is the displacement (output),
- $f(t)$ is the applied force (input),
- M is the mass,
- c is the damping coefficient, and
- κ is the stiffness of the spring.

The goal is to express the system using Fourier series coefficients and then implement a simulation in MATLAB using a backward difference method.

Theoretical Analysis

3.1 Fourier Series Representation

We can represent the input and output signals in their Fourier series forms:

$$f(t) = \sum_{k=-\infty}^{\infty} a_k\, e^{jk\omega_0 t} \quad \text{and} \quad y(t) = \sum_{k=-\infty}^{\infty} b_k\, e^{jk\omega_0 t},$$

where a_k and b_k are the Fourier coefficients for $f(t)$ and $y(t)$ respectively, and ω_0 is the fundamental frequency.

When we take derivatives, the Fourier coefficients change as follows:

- First derivative: $\frac{dy(t)}{dt} \to jk\omega_0\, b_k$.
- Second derivative: $\frac{d^2y(t)}{dt^2} \to -(k\omega_0)^2\, b_k$.

Substitute these into the differential equation:

$$M\left[-(k\omega_0)^2\, b_k\right] + c\left[jk\omega_0\, b_k\right] + \kappa\, b_k = a_k.$$

This simplifies to:

$$b_k\left[\kappa - M(k\omega_0)^2 + jck\omega_0\right] = a_k.$$

Thus, the relationship between the Fourier coefficients is:

$$b_k = \frac{a_k}{\kappa - M(k\omega_0)^2 + jck\omega_0}.$$

In other words, the system's frequency response $H(jk\omega_0)$ is given by:

$$H(jk\omega_0) = \frac{b_k}{a_k} = \frac{1}{\kappa - M(k\omega_0)^2 + jck\omega_0}.$$

3.2 Discretization Using Backward Difference

For the simulation, we use the following parameters:

$$M = 100, \quad c = 0.1, \quad \kappa = 0.1.$$

We assume the force $f(t)$ is given by the signal $x_3(t)$ defined as:

$$x_3(t) = r(t) - r(t-3) - 3u(t-3),$$

where $r(t) = t$ is a ramp function and $u(t)$ is the unit step function.

To simulate the system in discrete time, we approximate the derivatives with the backward difference method:

$$y'(t) \approx \frac{y[n] - y[n-1]}{T_s},$$
$$y''(t) \approx \frac{y[n] - 2y[n-1] + y[n-2]}{T_s^2}.$$

Substituting these into the differential equation, we have:

$$\frac{M(y[n] - 2y[n-1] + y[n-2])}{T_s^2} + \frac{c(y[n] - y[n-1])}{T_s} + \kappa\, y[n] = x_3[n].$$

Solving for $y[n]$:

$$y[n] = \frac{T_s^2 \cdot x_3[n] + (2M + c\,T_s)\, y[n-1] - M\, y[n-2]}{M + c\,T_s + \kappa\, T_s^2}.$$

MATLAB Implementation

Below is the MATLAB code that implements the above derivation. The code creates the $x_3(t)$ signal, simulates the system using the derived backward difference equation, and computes the Fourier series coefficients for both $x_3(t)$ and $y(t)$ using the `FSAnalysis` function.

```
Ts = 0.001;
T = 4;
t = 0:Ts:T-Ts;


x3 = zeros(size(t));
idx1 = t < 3;
x3(idx1) = t(idx1);
idx2 = t >= 3;
x3(idx2) = t(idx2) - (t(idx2)-3) - 3;


M = 100;
c = 0.1;
kappa = 0.1;
denom = M + c + kappa;

% y(n) = [ x3(n) + (2*M + c)*y(n-1) - M*y(n-2) ] / (M + c + kappa)
y = zeros(size(t));
y(1) = 0; y(2) = 0;
for n = 3:length(t)
    y(n) = ( x3(n) + (2*M + c)*y(n-1) - M*y(n-2) ) / denom;
end


figure;
subplot(2,1,1);
plot(t, x3, 'LineWidth',1.5);
title('x_3[n] (Giriş Sinyali)');
xlabel('t (s)'); ylabel('x_3[n]');
grid on;

subplot(2,1,2);
plot(t, y, 'LineWidth',1.5);
title('y[n] (Sistem Çıkışı)');
xlabel('t (s)'); ylabel('y[n]');
grid on;


k = 30;
a3 = FSAnalysis(x3, k, Ts);
b  = FSAnalysis(y, k, Ts);

figure;
subplot(2,2,1);
stem(-k:k, real(a3), 'filled');
title('x_3[n] Fourier Coefficients - Real');
xlabel('Harmonic (k)'); ylabel('Re\{a_k\}');
grid on;
```

```
subplot(2,2,2);
stem(-k:k, imag(a3), 'filled');
title('x_3[n] Fourier Coefficients - Imag');
xlabel('Harmonic (k)'); ylabel('Im\{a_k\}');
grid on;

subplot(2,2,3);
stem(-k:k, real(b), 'filled');
title('y[n] Fourier Coefficients - Real');
xlabel('Harmonic (k)'); ylabel('Re\{b_k\}');
grid on;

subplot(2,2,4);
stem(-k:k, imag(b), 'filled');
title('y[n] Fourier Coefficients - Imag');
xlabel('Harmonic (k)'); ylabel('Im\{b_k\}');
grid on;

function fsCoeffs = FSAnalysis(x, k, Ts)
% FSAnalysis: Compute Fourier series coefficients for a periodic signal.
% Inputs:
% x - One period of the sampled continuous-time signal (vector)
% k - Number of coefficients on each side (from -k to k)
% Ts - Sampling period
%
% Output:
% fsCoeffs - Fourier series coefficients (vector of length 2*k+1)
N = length(x); % Number of samples in one period
T0 = N * Ts; % Signal period
fsCoeffs = zeros(2*k+1, 1); % Initialize coefficients vector
n = 0:N-1; % Sample indices
    for m = -k:k
    % Compute coefficient a_m using the discrete sum approximation
    fsCoeffs(m + k + 1) = (Ts/T0) * sum(x .* exp(-1j * m * 2*pi * n/N));
    end
end
```

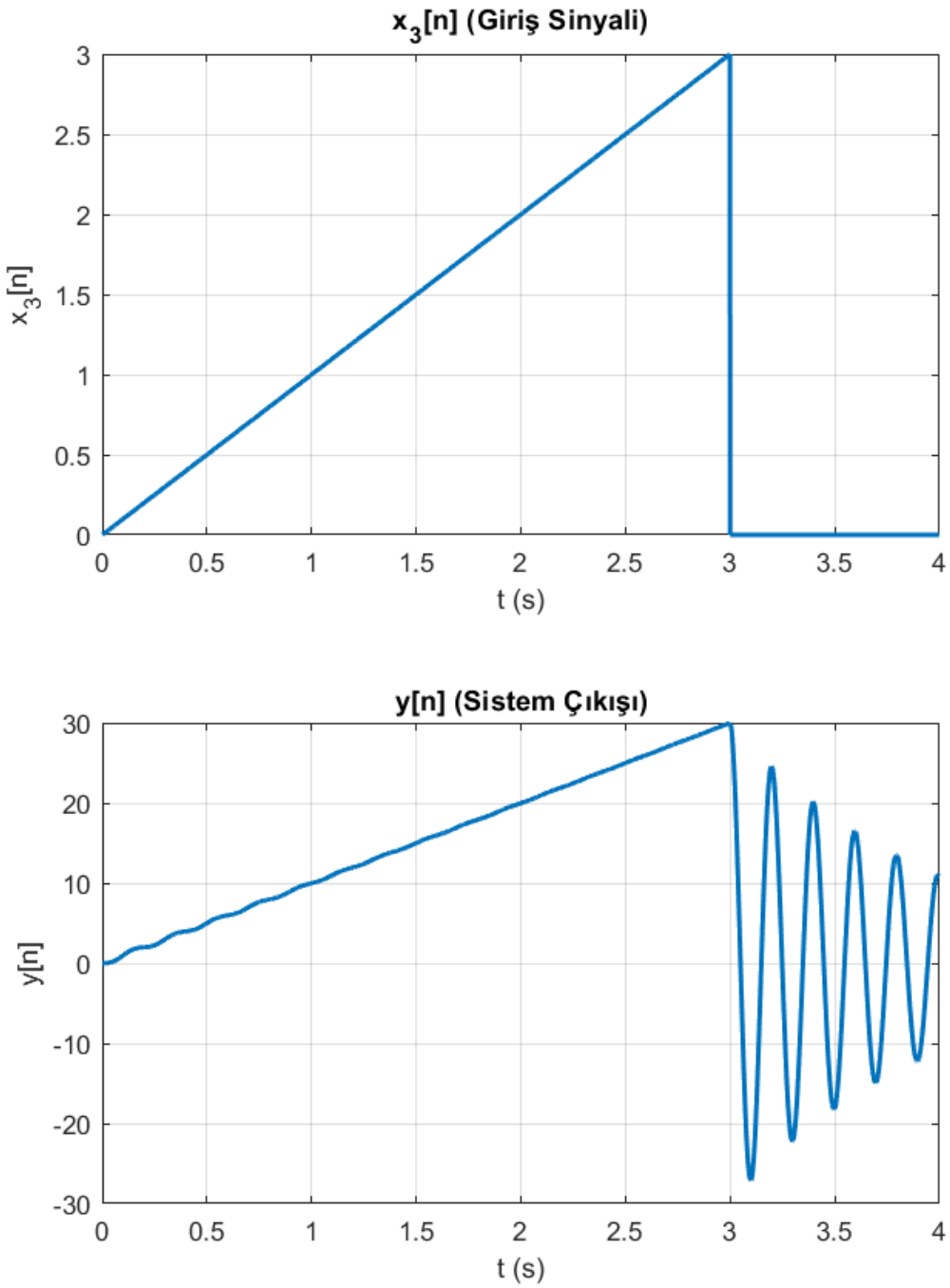

Figure 9: Part 3.1

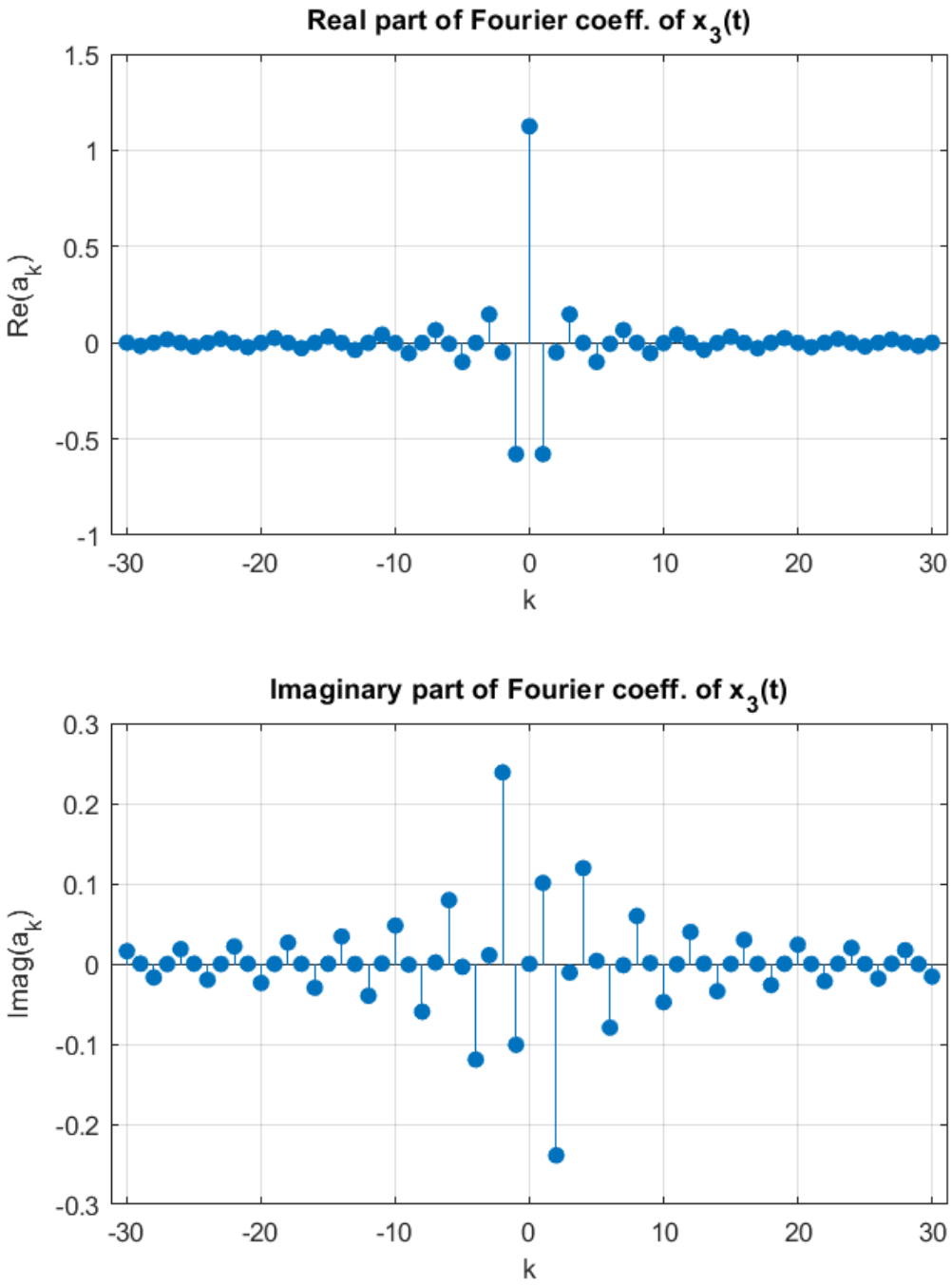

Figure 10: Part 3.2

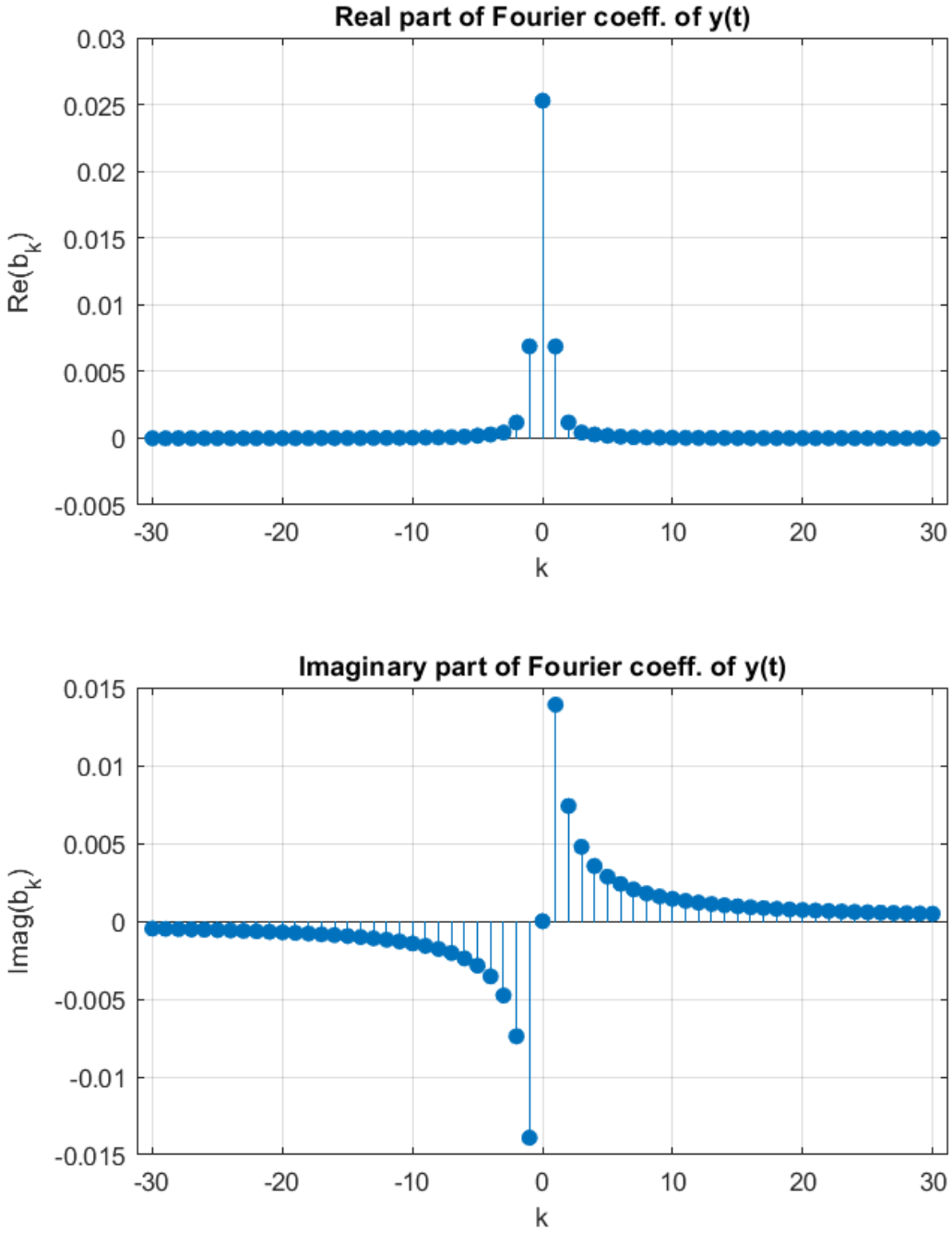

Figure 11: Part 3.3

Summary of Part 3

In this part, we analyzed a second-order system by:

- Expressing the differential equation in the Fourier series domain to find the relationship:
$$b_k = \frac{a_k}{\kappa - M(k\omega_0)^2 + jck\omega_0},$$
which defines the system's frequency response.

- Discretizing the system using the backward difference method. This allowed us to obtain the recursive equation:
$$y[n] = \frac{T_s^2 \cdot x_3[n] + (2M + c\,T_s)\,y[n-1] - M\,y[n-2]}{M + c\,T_s + \kappa\,T_s^2}.$$

- Implementing the simulation in MATLAB, where we generated the input signal $x_3(t)$, computed the output $y(t)$, and compared the Fourier series coefficients of both signals.

This detailed explanation and MATLAB code help us understand how a second-order physical system affects the Fourier series coefficients of an input signal.

General Conclusion

In this lab, I explored both the theoretical and practical aspects of Fourier series. I implemented the `FSAnalysis` function in MATLAB to compute Fourier coefficients of various periodic signals. Then, I analyzed how different time-domain operations—such as time reversal, shifting, differentiation, and squaring—affect the frequency-domain representation. Finally, I examined how a second-order physical system alters the spectral content of an input signal using both analytical derivation and numerical simulation. Overall, this lab helped me understand how time-domain and frequency-domain analyses are interconnected in signal processing.