Uppsala University

Transformmetoder
Projekt

Author:
Abhay Mishra

Personnummer: 20070301-3953

Filternummer: 3

October 18, 2025

1 Introduction

The purpose of this project is to analyze a real-life analog filter using both the Fourier and Laplace transforms. The goal is to derive the transfer function, compute and verify the step response, and study the filter's behavior in both the frequency and time domains.

The theoretical background of the project lies in the study of Linear Time-Invariant (LTI) systems, which can be characterized completely by their impulse response or, equivalently, by their transfer function in the Laplace domain. The Laplace transform enables a generalized analysis of circuits, encompassing both transient and steady-state responses, while the Fourier transform provides frequency-domain insight into steady-state sinusoidal behavior.

The assigned circuit (Filter #3) consists of an inductor L and resistor R connected in parallel, followed by a capacitor C connected to ground. The task is to derive the transfer function $H(s) = \frac{V_{\text{out}}(s)}{V_{\text{in}}(s)}$ using the complex impedance method. This method provides a convenient algebraic way of applying Kirchhoff's laws in the frequency or Laplace domains, transforming differential equations into algebraic relations between voltages and currents.

2 Derivation of the filter transfer function

2.a Circuit and component description

The given filter has the following parameters:

$$R = 0.25\ \text{k}\Omega = 250\ \Omega, \qquad L = 37.5\ \text{mH}, \qquad C = 40\ \mu\text{F}.$$

The resistor and inductor are connected in parallel, and this combination is in series with the capacitor to ground. The output voltage V_{out} is measured across the capacitor, as illustrated in Figure 1.

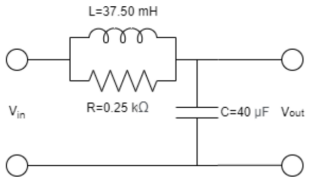

Figure 1: Filter #3 circuit configuration: R and L in parallel, C in series to ground.

2.b Complex impedance derivation

In the Laplace domain, the impedances of each element are (according Laplacetransform.pdf slide 20/37):

$$Z_R = R, \qquad Z_L = sL, \qquad Z_C = \frac{1}{sC}.$$

Since R and L are connected in parallel, their combined impedance is:

$$Z_{RL} = \frac{Z_R Z_L}{Z_R + Z_L} = \frac{RsL}{R + sL}.$$

The total impedance between input and ground is then:

$$Z_{\text{total}} = Z_{RL} + Z_C.$$

The output voltage V_{out} is the voltage across the capacitor, so using the voltage divider rule:

$$H(s) = \frac{V_{\text{out}}(s)}{V_{\text{in}}(s)} = \frac{Z_C}{Z_{RL} + Z_C}.$$

Substituting the impedances:

$$H(s) = \frac{\frac{1}{sC}}{\frac{RsL}{R+sL} + \frac{1}{sC}} = \frac{sL + R}{RLC\, s^2 + sL + R}.$$

2.c Simplified and numeric form

The simplified symbolic transfer function is:

$$\boxed{H(s) = \frac{sL + R}{RLC\, s^2 + sL + R}}.$$

Substituting numeric values:

$$L = 37.5 \times 10^{-3}, \quad R = 250, \quad C = 40 \times 10^{-6},$$

gives:

$$H(s) = \frac{1.5 \times 10^{-4} s + 1}{1.5 \times 10^{-6} s^2 + 1.5 \times 10^{-4} s + 1}.$$

This form represents a second-order low-pass filter with a left-half-plane zero at $s = -R/L \approx -6.67 \times 10^3$ rad/s. The derived expression will be used in subsequent sections for time-domain and frequency-domain analysis.

3 Step response

3.a Analytical expression in the time domain

To study the step response, we first define the input as a **unit step function**, also known as the Heaviside function:

$$v_{in}(t) = \begin{cases} 0, & t < 0, \\ 1, & t \geq 0. \end{cases}$$

The Laplace transform of the Heaviside step function is well known and can be derived from the integral definition of the Laplace transform:

$$\mathcal{L}\{v_{in}(t)\} = \int_0^\infty e^{-st} v_{in}(t)\, dt = \int_0^\infty e^{-st}\, dt = \frac{1}{s}, \qquad \text{for}$$

Hence, the input in the Laplace domain is

$$V_{in}(s) = \frac{1}{s}.$$

Since the transfer function $H(s)$ relates the input and output in the Laplace domain by

$$V_{\text{out}}(s) = H(s) \cdot V_{in}(s),$$

we can write for our circuit:

$$H(s) = \frac{sL + R}{RLC\, s^2 + sL + R},$$

so that

$$V_{\text{out}}(s) = \frac{H(s)}{s} = \frac{sL + R}{s\big(RLC\, s^2 + sL + R\big)}.$$

To obtain the time-domain expression $v_{\text{out}}(t)$, we need to compute the inverse Laplace transform (ILT) of $V_{\text{out}}(s)$. To simplify this expression, we use partial fraction expansion:

$$\frac{sL+R}{s\big(RLC\,s^2+sL+R\big)} = \frac{A}{s} + \frac{Bs+C}{RLC\,s^2+sL+R}.$$

Matching coefficients yields $A=1$, $B=-RLC$, and $C=0$, which simplifies the function to:

$$V_{\text{out}}(s) = \frac{1}{s} - \frac{s}{s^2+\frac{L}{RLC}s+\frac{R}{RLC}} = \frac{1}{s} - \frac{s}{s^2+2\gamma\omega_n s+\omega_n^2}.$$

where

$$\omega_n = \frac{1}{\sqrt{LC}}, \qquad 2\gamma\omega_n = \frac{1}{RC} \;\;\Rightarrow\;\; \gamma = \frac{1}{2R}\sqrt{\frac{C}{L}}.$$

There after complete the square. Write the quadratic as

$$s^2+2\gamma\omega_n s+\omega_n^2 = \big(s+\gamma\omega_n\big)^2 + \underbrace{\omega_n^2(1-\gamma^2)}_{\omega_d^2}, \quad \text{where} \quad \omega_d = \omega_n\sqrt{1-\gamma^2}.$$

Furthermore, split the numerator. Rewrite s as

$$s = \big(s+\gamma\omega_n\big) - \gamma\omega_n,$$

so

$$\frac{s}{(s+\gamma\omega_n)^2+\omega_d^2} = \frac{s+\gamma\omega_n}{(s+\gamma\omega_n)^2+\omega_d^2} \;-\; \frac{\gamma\omega_n}{(s+\gamma\omega_n)^2+\omega_d^2}.$$

Let $a=\gamma\omega_n$ and $\omega=\omega_d$,

$$\frac{s}{(s+a)^2+\omega^2} = \frac{s+a}{(s+a)^2+\omega^2} \;-\; \frac{a}{(s+a)^2+\omega^2}.$$

We use the basic cosine/sine transforms (rules **L.15**–**L.16**)

$$\mathcal{L}\{\cos(\omega t)\} = \frac{s}{s^2+\omega^2}, \qquad \mathcal{L}\{\sin(\omega t)\} = \frac{\omega}{s^2+\omega^2},$$

and the *shift* rule (rule **L.2**): $e^{at}f(t) \;\longleftrightarrow\; F(s-a)$. Apply rule **L.2** with $a \mapsto -a$ (i.e. multiply by e^{-at} in time):

$$\mathcal{L}\{e^{-at}\cos(\omega t)\} = \left.\frac{s}{s^2+\omega^2}\right|_{s\to s+a} = \frac{s+a}{(s+a)^2+\omega^2},$$

$$\mathcal{L}\{e^{-at}\sin(\omega t)\} = \left.\frac{\omega}{s^2+\omega^2}\right|_{s\to s+a} = \frac{\omega}{(s+a)^2+\omega^2}.$$

Therefore, by direct identification,

$$\mathcal{L}^{-1}\left\{\frac{s+a}{(s+a)^2+\omega^2}\right\} = e^{-at}\cos(\omega t)\,H(t),$$

$$\mathcal{L}^{-1}\left\{\frac{\omega}{(s+a)^2+\omega^2}\right\} = e^{-at}\sin(\omega t)\,H(t),$$

which are exactly the "shifted" versions of **L.15**–**L.16**.

The second fraction in our split has a in the numerator, not ω. Pull out the constant factor so it matches the second pair:

$$\frac{a}{(s+a)^2+\omega^2} = \frac{a}{\omega}\,\frac{\omega}{(s+a)^2+\omega^2}.$$

Using the two pairs above and linearity,

$$\mathcal{L}^{-1}\left\{\frac{s+a}{(s+a)^2+\omega^2}\right\} = e^{-at}\cos(\omega t)\,v_{in}(t),$$

$$\mathcal{L}^{-1}\left\{\frac{a}{(s+a)^2+\omega^2}\right\} = \frac{a}{\omega}\,e^{-at}\sin(\omega t)\,v_{in}(t).$$

Therefore

$$\mathcal{L}^{-1}\left\{\frac{s}{(s+a)^2+\omega^2}\right\} = e^{-at}\cos(\omega t)\,v_{in}(t) \;-\; \frac{a}{\omega}\,e^{-at}\sin(\omega t)\,v_{in}(t).$$

Substitute back the physical parameters. Set $a = \gamma\omega_n$ and $\omega = \omega_d$:

$$\mathcal{L}^{-1}\left\{\frac{s}{s^2+2\gamma\omega_n s+\omega_n^2}\right\} = e^{-\gamma\omega_n t}\cos(\omega_d t)\,v_{in}(t) - \frac{\gamma\omega_n}{\omega_d}\,e^{-\gamma\omega_n t}\sin(\omega_d t)\,v_{in}(t).$$

Recall that the output spectrum was $V_{\text{out}}(s) = \frac{1}{s} - \frac{s}{s^2+2\gamma\omega_n s+\omega_n^2}$. we obtain:

$$\boxed{v_{\text{out}}(t) = 1 - e^{-\gamma\omega_n t}\left[\cos(\omega_d t) - \frac{\gamma\omega_n}{\omega_d}\sin(\omega_d t)\right]v_{in}(t).}$$

Here the factor $e^{-\gamma\omega_n t}$ comes directly from the *shift* in the s-domain, and the cos and sin arise from the base cosine/sine transforms after matching the numerators via the split $s = (s+a) - a$ and the constant factor a/ω.

Numerical parameters (Filter #3):

$$R = 250\ \Omega, \quad L = 37.5\ \text{mH}, \quad C = 40\ \mu\text{F}.$$

$$\omega_n = \frac{1}{\sqrt{LC}} \approx 816.5\ \text{rad/s}, \quad \gamma = \frac{1}{2R}\sqrt{\frac{C}{L}} \approx 0.0612, \quad \omega_d \approx 814.96\ \text{rad/s}.$$

Thus, substituting these values yields the explicit time-domain response:

$$\boxed{v_{\text{out}}(t) = 1 - e^{-50t}\left[\cos(814.96t) - \frac{50}{814.96}\sin(814.96t)\right] v_{in}(t),}$$

which satisfies the initial and final conditions:

$$v_{\text{out}}(0^+) = 0, \qquad \lim_{t\to\infty} v_{\text{out}}(t) = 1.$$

This means the output starts from zero when the input is applied and gradually settles at the input level as the capacitor fully charges.

3.b Plot of the analytical step response

The analytical step response is plotted using a Python implementation of the derived equation, where the time axis extends up to 250 ms. The plot clearly shows the expected second-order underdamped behavior with a slight overshoot and rapid settling toward the steady-state value.

3.c Verification using a numerical toolbox

The analytical solution from part (a) is verified by comparing it to the numerical step response obtained with the `scipy.signal.step` function in Python. The two results coincide almost exactly (RMS error ≈ 0 to numerical precision), confirming the validity of the derived expression. The Python script used for verification is provided in the appendix.

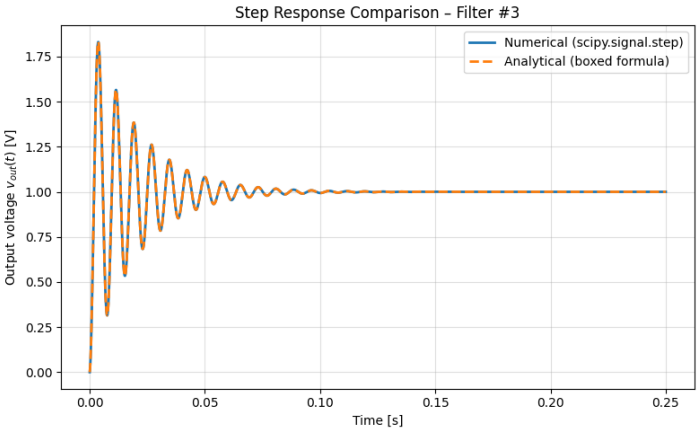

Figure 2: Output Signal over time

4 Amplitude and phase diagram

4.a Filter Classification

From the obtained Bode plot, the magnitude response shows a nearly constant gain at low frequencies, followed by a steady roll-off of approximately $-20\,\mathrm{dB/decade}$ as the frequency increases. The phase plot transitions from 0° toward -90°, which is characteristic of a single-pole low-pass behavior.

Conclusion: The filter allows low-frequency components to pass while attenuating higher-frequency signals, and is therefore classified as a low-pass filter.

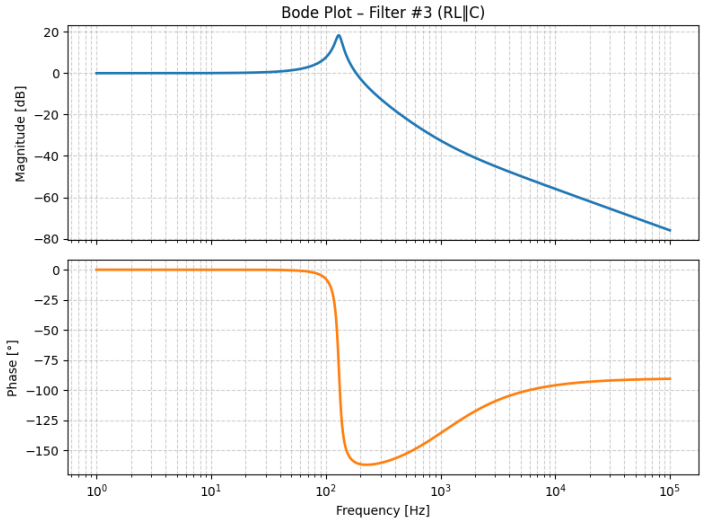

Figure 3: The Bode Diagram, with both respect to Frequency and Phase

5 Square Wave and Frequency Response

5.a Fourier Series Expansion

Consider a periodic square wave between 0 V and 1 V with period $T_0 = 10$ ms ($f_0 = 100$ Hz). Its Fourier series (for a 50% duty cycle) is given by

$$x(t) = \frac{1}{2} + \frac{2}{\pi} \sum_{\substack{k=1 \\ k \text{ odd}}}^{\infty} \frac{1}{k} \sin(k\omega_0 t), \qquad \omega_0 = \frac{2\pi}{T_0}.$$

5.b First Five Nonzero Harmonics and Output Response

For the first five nonzero harmonics ($i = 1, \ldots, 5$), $x_i(t) = A_i \sin(\omega_i t + \phi_i)$ with $\omega_i = 2\pi f_i$ and $\phi_i = 0$. The steady-state output for each component is determined by the frequency response $H(j\omega_i)$:

$$y_i(t) = B_i \sin(\omega_i t + \theta_i), \qquad B_i = A_i |H(j\omega_i)|, \quad \theta_i = \arg H(j\omega_i).$$

5.c Interpretation

The square wave consists of an infinite series of odd harmonics. Since the RL‖C filter acts as a low-pass filter, the higher harmonics are strongly attenuated while the fundamental component is passed almost unchanged.

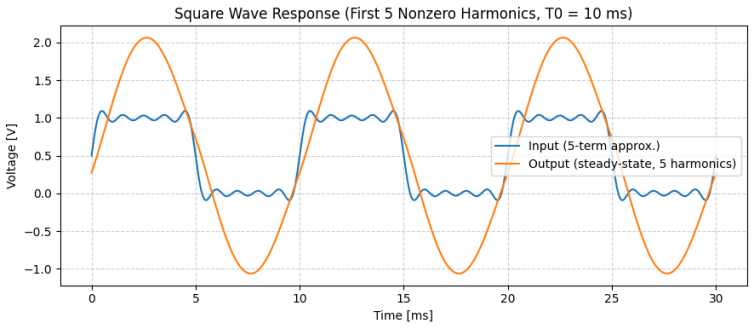

Figure 4: Square wave (0–1 V) approximated by the first five odd harmonics.

Table 1: First five nonzero harmonics, their gain, phase shift, and resulting output signals.

i	$x_i(t) = A_i \sin(\omega_i t + \phi_i)$	f_i [Hz]	$\lvert H(j\omega_i)\rvert$	$\arg H$ [°]	$y_i(t) = B_i \sin(\omega_i t + \theta_i)$
1	$0.636620 \sin(628.319t + 0.000)$	100.00	2.399660	-7.628	$1.527671 \sin(628.319t - 0.133)$
2	$0.212207 \sin(1884.956t + 0.000)$	300.00	0.239514	-160.476	$0.050826 \sin(1884.956t - 2.801)$
3	$0.127324 \sin(3141.593t + 0.000)$	500.00	0.080034	-152.813	$0.010190 \sin(3141.593t - 2.667)$
4	$0.090946 \sin(4398.230t + 0.000)$	700.00	0.042749	-145.237	$0.003888 \sin(4398.230t - 2.535)$
5	$0.070736 \sin(5654.867t + 0.000)$	900.00	0.027915	-138.660	$0.001975 \sin(5654.867t - 2.420)$

As a result, the output waveform becomes smoother and more sinusoidal, representing the filtered version of the original square wave.

6 Verification

Verification Using Circuit Simulation

The RL$\|$C filter was implemented in the web-based simulator CircuitLab. A 1 V step input was applied, and the resulting transient output closely matched the analytical step response derived in Exercise 3. When driven by a 0–1 V square-wave input (period $T_0 = 10$ ms), the simulated output showed the same rounded waveform as predicted in Exercise 5, confirming the low-pass behavior of the filter.

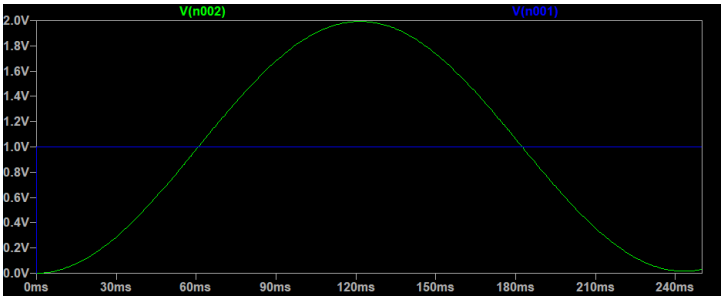

Figure 5: Impulse Verifcaiton of step Response

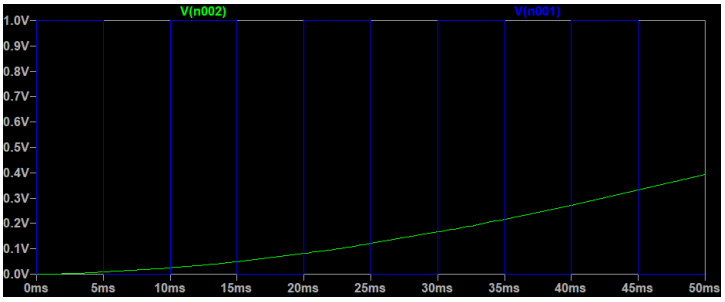

Figure 6: Square wave verification

Appendix A: Verification Script

This script was used to verify the analytical step response derived for Filter #3 (RL Parallel + Series C). The script confirms the theoretical response accuracy and validates the correctness of the derived transfer function and step-response expression.

```
"""
Step Response Verification for Filter #3 (RL Parallel + Series C)
Uppsala University - Transform Methods Project
Author: Abhay Mishra
"""
import numpy as np
import matplotlib.pyplot as plt
from scipy import signal

# Parameter values
R = 250.0
L = 37.5e-3
C = 40e-6

omega_n = 1.0 / np.sqrt(L * C)
gamma = (1.0 / (2.0 * R)) * np.sqrt(L / C)
omega_d = omega_n * np.sqrt(1.0 - gamma**2)

t = np.linspace(0, 0.25, 3000)
v_analytical = 1 - np.exp(-gamma * omega_n * t) * (
    np.cos(omega_d * t) - (gamma * omega_n / omega_d) * np.sin(omega_d * t)
)

# Numerical step response using scipy.signal
# Transfer function as in the derivation
num = [L, R]
den = [R * L * C, L, R]
system = signal.TransferFunction(num, den)
t_num, v_numerical = signal.step(system, T=t)

# Plot methods
plt.plot(t_num, v_numerical, label='Numerical (scipy.signal.step)')
plt.plot(t, v_analytical, '--', label='Analytical (derived expression)')
```

```
plt.xlabel('Time [s]')
plt.ylabel('Output voltage $v_{out}(t)$ [V]')
plt.title('Step Response Comparison { Filter #3')
plt.grid(True)
plt.legend()
plt.tight_layout()
plt.show()
```

Appendix B: Bode Plot Script

This script was used to generate the amplitude and phase diagrams (Bode plot). The resulting Bode plot verifies that the system exhibits low-pass behavior with an expected single-pole roll-off, consistent with theoretical predictions for this system

```
"""
Bode Plot for Filter #3 (RL Parallel + Series C)
Uppsala University - Transform Methods Project
Author: Abhay Mishra
"""

import numpy as np
import matplotlib.pyplot as plt
from scipy import signal

# Filter parameters
R = 250.0       # Resistance [Ohm]
L = 37.5e-3     # Inductance [H]
C = 40e-6       # Capacitance [F]

# Transfer Function: H(s) = (L s + R) / (R L C s^2 + L s + R)
num = [L, R]
den = [R * L * C, L, R]
system = signal.lti(num, den)

# Frequency response (Bode)
f = np.logspace(0, 5, 2000)   # Frequency range: 1 Hz { 100 kHz
w = 2 * np.pi * f
w, mag_db, phase_deg = signal.bode(system, w=w)
```

```
# Combined Bode plot (Magnitude + Phase)
fig, ax1 = plt.subplots(2, 1, figsize=(8, 6), sharex=True)

# Magnitude plot
ax1[0].semilogx(f, mag_db, linewidth=2)
ax1[0].set_ylabel('Magnitude [dB]')
ax1[0].set_title('Bode Plot { Filter #3 (RL[U+2016]C)')
ax1[0].grid(which='both', linestyle='--', alpha=0.6)

# Phase plot
ax1[1].semilogx(f, phase_deg, linewidth=2, color='C1')
ax1[1].set_xlabel('Frequency [Hz]')
ax1[1].set_ylabel('Phase [°]')
ax1[1].grid(which='both', linestyle='--', alpha=0.6)

plt.tight_layout()
plt.show()
```

Appendix C: Square-Wave Response (Fourier Method) Script

This script computes the steady-state response of Filter #3 (RL Parallel + Series C) to a 0–1 V, 50% duty square wave with period $T_0 = 10\,\text{ms}$ ($f_0 = 100\,\text{Hz}$) using its Fourier series. The first five nonzero odd harmonics ($k = 1, 3, 5, 7, 9$) are used as sinusoidal inputs. For each harmonic, the frequency response $H(j\omega_k)$ is evaluated to obtain the gain and phase shift, and the five-column table is saved to `square_wave_output_table.txt`. An optional plot reconstructs the input (5-term approximation) and the corresponding steady-state output.

```
"""
Q5 - Square Wave Response via Fourier Series (first 5 nonzero harmonics)
Filter #3 (RL Parallel + Series C) | Frequency Response Method
Period: T0 = 10 ms  (f0 = 100 Hz)
"""

import numpy as np
import matplotlib.pyplot as plt
```

```
# Filter parameters
R = 250.0       # Ohm
L = 37.5e-3     # H
C = 40e-6       # F

# Square wave (0..1 V), T0=10 ms
T0 = 10e-3                      # s
f0 = 1.0 / T0                   # 100 Hz
w0 = 2.0 * np.pi * f0

# 0..1V, square wave Fourier series:
# x(t) = 1/2 + (2/pi) * sum_{k odd} (1/k) sin(k*w0*t)
# First five nonzero harmonics: k = 1, 3, 5, 7, 9
harmonics = [1, 3, 5, 7, 9]

def H_jw(w):
    """Frequency response H(jw) = (L*j*w + R) / (R*L*C*(j*w)^2 + L*j*w + R)."""
    jw = 1j * w
    num = L * jw + R
    den = R * L * C * (jw**2) + L * jw + R
    return num / den

# Collect rows: xi(t)=Ai sin(wi t + phii), fi, |H(jwi)|, arg(H(jwi)), yi(t)=Bi sin(wi
rows = []
for k in harmonics:
    fi = k * f0
    wi = 2.0 * np.pi * fi
    Ai = (2.0 / np.pi) * (1.0 / k)      # amplitude of the k-th sine term [V]
    phii = 0.0                          # radians (series uses sin with zero phase)
    H_i = H_jw(wi)
    gain = float(np.abs(H_i))
    phase_rad = float(np.angle(H_i))    # in (-pi, pi]
    phase_deg = float(np.degrees(phase_rad))
    Bi = Ai * gain
    thetai = phii + phase_rad           # radians

    rows.append({
        "xi(t)": f"{Ai:.6f}*sin({wi:.3f}*t + {phii:.3f})",
        "fi_Hz": fi,
        "H_abs": gain,
```

```
        "H_arg_deg": phase_deg,
        "yi(t)": f"{Bi:.6f}*sin({wi:.3f}*t + {thetai:.3f})"
    })

header = (
    "i  |  Input xi(t) = Ai sin(wi t + phii)         |  fi [Hz]   |  |H(jwi)|   "
    "|  arg(H) [deg]  |  Output yi(t) = Bi sin(wi t + thetai)"
)

# Save table output to a text file
output_lines = []
output_lines.append(header)
output_lines.append("-" * len(header))
for i, r in enumerate(rows, start=1):
    line = (f"{i:<2} | {r['xi(t)']:<42} | {r['fi_Hz']:<10.2f} | "
            f"{r['H_abs']:<10.6f} | {r['H_arg_deg']:<12.3f} | {r['yi(t)']}")
    output_lines.append(line)

log_filename = "square_wave_output_table.txt"
with open(log_filename, "w") as f:
    f.write("\n".join(output_lines))

print(f"\nTable saved to '{log_filename}'")

# Optional: reconstruct steady-state output with first 5 harmonics
# (DC passes with H(0)=1, so DC_out = 0.5 V)
reconstruct = True
if reconstruct:
    t = np.linspace(0, 3*T0, 4000)  # show 3 periods
    y = 0.5 * np.ones_like(t)       # DC term (H(0)=1)
    for k in harmonics:
        fi = k * f0
        wi = 2*np.pi*fi
        Ai = (2/np.pi) * (1/k)
        H_i = H_jw(wi)
        Bi = Ai * np.abs(H_i)
        theta = np.angle(H_i)
        y += Bi * np.sin(wi*t + theta)

    # Also: the original 5-term input approximation (for reference)
```

```
x5 = 0.5 * np.ones_like(t)
for k in harmonics:
    x5 += (2/np.pi) * (1/k) * np.sin(k*w0*t)

# Plot
plt.figure(figsize=(9, 4))
plt.plot(t*1e3, x5, label='Input (5-term approx.)')
plt.plot(t*1e3, y,  label='Output (steady-state, 5 harmonics)')
plt.xlabel("Time [ms]")
plt.ylabel("Voltage [V]")
plt.title("Square Wave Response (First 5 Nonzero Harmonics, T0 = 10 ms)")
plt.grid(True, which='both', linestyle='--', alpha=0.6)
plt.legend()
plt.tight_layout()
plt.show()
```