Dynamic Analysis of a Series RLC Circuit

20221242 박수빈

목차

1. 설계 목적

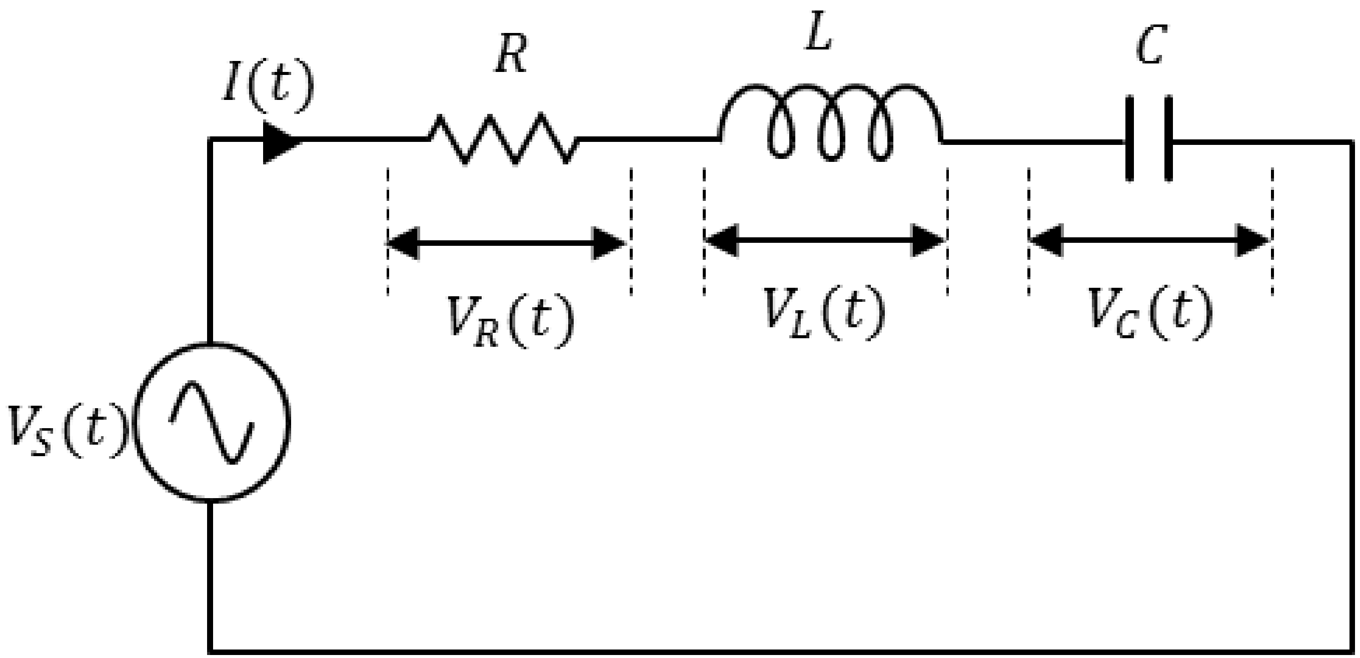
2. 기본 이론

   - series RLC circuit, damped oscillation, frquency response

3. 코드 설명

4. 실행 결과 및 결론

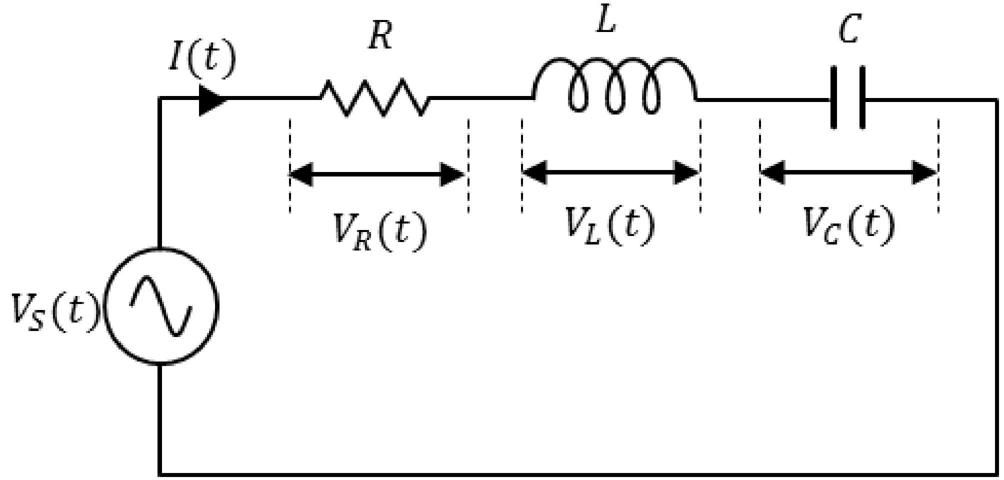
설계 목적



직렬 RLC 회로에서 V_C 분석 단순화 및 시각화

1. 감쇠비, Q 등 parameter 분석 후 damped oscillation 구분
2. 시간에 따른 V_C 그래프와 주파수 응답 plot

기본 이론 - 직렬 RLC 회로



직렬 RLC 회로에 Kirchhoff's law 적용

$$L\ddot{q} + R\dot{q} + \frac{1}{C}q = V_{\text{in}}(t), \quad V_C = \frac{q}{C}$$

$$\ddot{y} + 2\zeta\omega_0\,\dot{y} + \omega_0^2 y = (\text{입력}/\text{상수})$$

cf) damped oscillation model

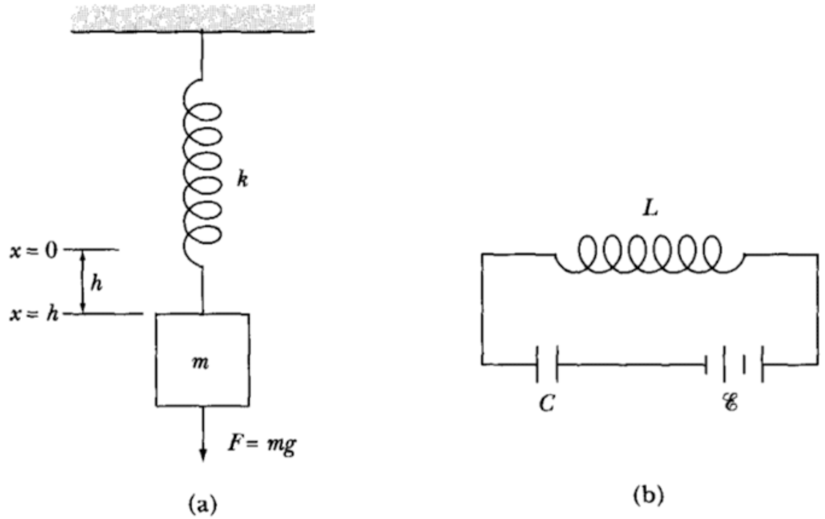


TABLE 3-1 Analogous Mechanical and Electrical Quantities

Mechanical		Electrical	
x	Displacement	q	Charge
$\dot{x}$	Velocity	$\dot{q} = I$	Current
m	Mass	L	Inductance
b	Damping resistance	R	Resistance
$1/k$	Mechanical compliance	C	Capacitance
F	Amplitude of impressed force	$\mathcal{E}$	Amplitude of impressed emf

기본 이론 - Damped Oscillations

$$L\ddot{q} + R\dot{q} + \frac{1}{C}q = V_{\mathrm{in}}(t), \quad V_C = \frac{q}{C}$$

1) 직류 전원일 때 (V_{in} 은 상수)

표준형: $\ddot{y} + 2\zeta\omega_0\,\dot{y} + \omega_0^2 y = (\text{입력}/\text{상수})$

$$\omega_0 = \frac{1}{\sqrt{LC}} \qquad \zeta = \frac{R}{2}\sqrt{\frac{L}{C}} \qquad Q = \frac{1}{2\zeta} = \frac{1}{R}\sqrt{\frac{L}{C}}$$

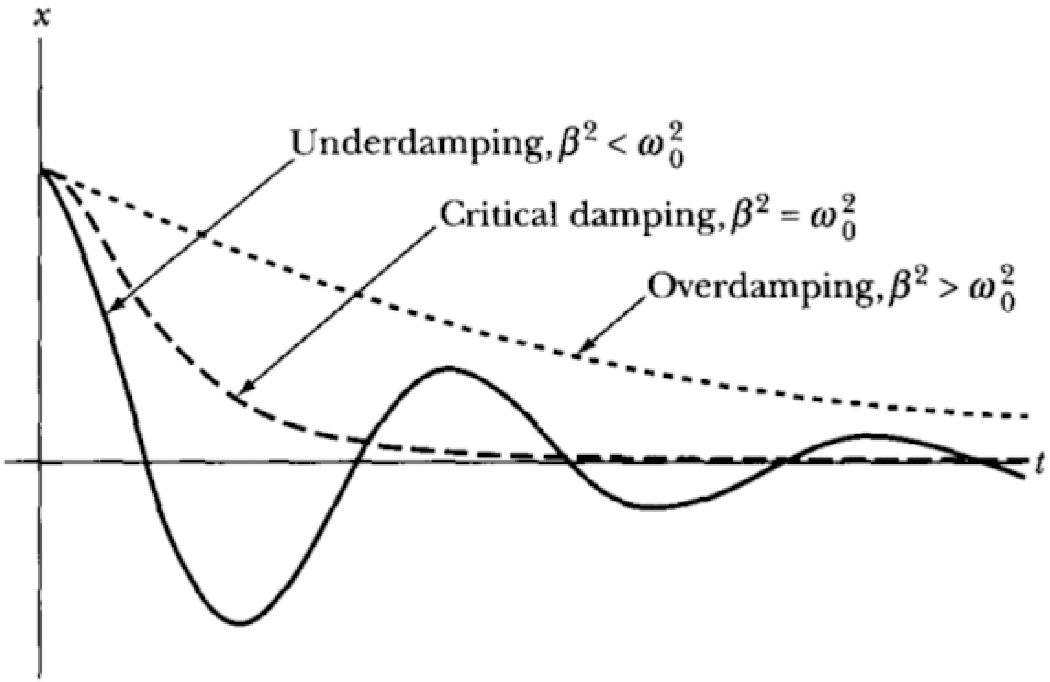
(에너지 저장 효율,
공진 특성, 대역폭과 반비례 관계)

미분 방정식: $s^2 + 2\zeta\omega_0 s + \omega_0^2 = 0$

$$s_{1,2} = \begin{cases} -\zeta\omega_0 \pm j\omega_d, & (\zeta < 1, Underdamping) \\ -\omega_0\ (\text{중근}), & (\zeta = 1, Critical\ damping) \\ -\omega_0(\zeta \pm \sqrt{\zeta^2 - 1}, & (\zeta > 1, Overdamping) \end{cases}$$

$$\omega_d = \omega_0\sqrt{1 - \zeta^2}$$

기본 이론 - Damped Oscillations



$V_C(0) = 0$ (방전)
$V_C(\infty) = V_{in}$ (충전)

- Underdamping

$$V_C(t) = V_{\text{in}}\left[1 - \frac{1}{\sqrt{1-\zeta^2}} e^{-\zeta\omega_0 t}\sin(\omega_d t + \arccos\zeta)\right]$$

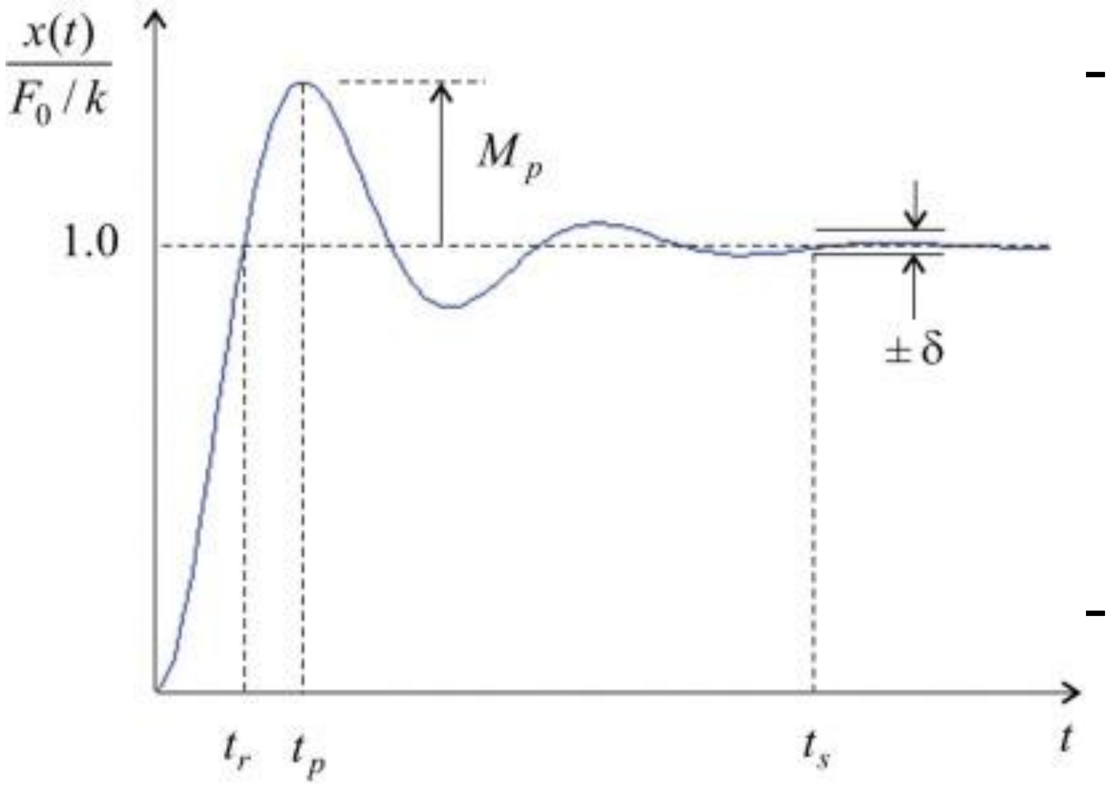
- Critical damping

$$V_C(t) = V_{\text{in}}[1 - \{1 + \omega_0 t\} e^{-\omega_0 t}]$$

- Overdamping

$$V_C(t) = V_{\text{in}}\left[1 - \frac{s_2 e^{s_1 t} - s_1 e^{s_2 t}}{s_2 - s_1}\right]$$

기본 이론 - Damped Oscillations



Underdamped Oscillation의 특성 지표

- T_s : 정착 시간

  응답 곡선이 수렴하는 값의 오차 범위에 들어가는 데 걸리는 시간

$$T_s = -\frac{1}{\zeta\omega_n}\ln\delta\sqrt{1-\zeta^2} = \frac{4}{\zeta\omega_n}, \quad \delta\text{=0.02}$$

- M_p: 최대 오버 슈트

  응답 곡선 최대 봉우리에서 1을 뺀 값

$$M_p = \exp\left(-\frac{\pi\zeta}{\sqrt{1-\zeta^2}}\right)$$

기본 이론 - Damped Oscillations

$$L\ddot{q} + R\dot{q} + \frac{1}{C}q = V_{\text{in}}(t), \quad V_C = \frac{q}{C}$$

2) 교류 전원일 때 (V_{in} 은 AC+DC)

표준형: $\ddot{y} + 2\zeta\omega_0\dot{y} + \omega_0^2 y = V_{in}\sin(\omega t)$, $y(t) = V_C(t)$

$$\omega_0 = \frac{1}{\sqrt{LC}} \qquad \zeta = \frac{R}{2}\sqrt{\frac{L}{C}} \qquad Q = \frac{1}{2\zeta} = \frac{1}{R}\sqrt{\frac{L}{C}}$$

$$V_C(t) = V_C^{(h)}(t) + V_C^{(p)}(t)$$

$V_C^{(h)}(t)$는 자유응답, complementary function과 유사 (앞서 구함)

$V_C^{(p)}(t)$는 강제응답, particular solution과 유사

기본 이론 - Damped Oscillations

$$V_C(t) = V_C^{(h)}(t) + V_C^{(p)}(t)$$

$V_C^{(p)}(t)$는 전달 함수를 이용해 쉽게 구할 수 있음
-> 전달함수는 steady-state 상황에서 페이저 변환을 이용해 계산

임피던스: $Z_R = R,\ Z_L = j\omega L,\ Z_C = \dfrac{1}{j\omega C}$

전달 함수: $H(j\omega) = \dfrac{V_C}{V_{in}} = \dfrac{Z_C}{Z_{tot}} = \dfrac{1}{1 - j\omega RC - \omega^2 LC}$

$$|H(j\omega)| = \frac{1}{\sqrt{(1-\omega LC)^2-(\omega RC)^2}} \qquad \angle H(j\omega) = \phi = \arctan\left(-\frac{\omega RC}{1-\omega^2 LC}\right)$$

-> $V_C^{(p)}(t) = |H(j\omega)| V_{in} \sin(\omega t + \phi)$

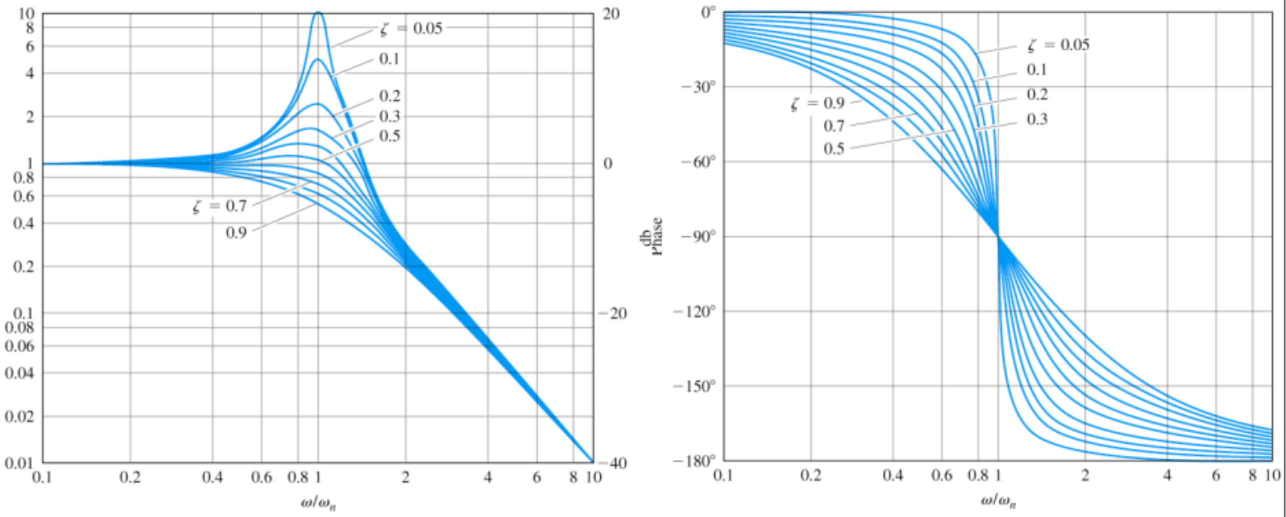
기본 이론 - 주파수 응답

주파수 응답: 신호 주파수의 변화에 따른 회로 동작의 변화, 교류 전원 인가시 필요

전달 함수: $H(j\omega) = \frac{V_C}{V_{in}} = \frac{Z_C}{Z_{tot}} = \frac{1}{1 - j\omega RC - \omega^2 LC}$

$|H(j\omega)| = \frac{1}{\sqrt{(1-\omega LC)^2 - (\omega RC)^2}}$ $\quad \angle H(j\omega) = \phi = \arctan\left(-\frac{\omega RC}{1-\omega^2 LC}\right)$

- Frequency response of $G(s) = \frac{1}{(s/\omega_n)^2 + 2\zeta(s/\omega_n) + 1}$



Magnitude $M(\omega)$ Phase $\phi(\omega)$

Q가 커질수록 공진주파수 근처에서
진폭 응답 뾰족해짐
위상 응답 기울기 가팔라짐

이미지 출처
https://velog.io/@baeksh1997/%EC%9E%90%EB%8F%99%EC%A0%9C%EC%96%B4-The-Frequency-Response-Design-Method

코드 설명

1. DC 입력
  - 핵심 parameter 정의
  - ODE solver와 analytic solution 통해 Vc 계산
  - 출력할 그래프 세부 설정
  - 각각의 oscillation에 대한 출력값 설정
  - 메인 함수를 통해 앞서 정의한 함수 불러오고, plotting
2. AC 입력
  - 파라미터, 전달함수, 정의
  - ODE solver 통해 Vc 계산
  - 메인 함수 통해 plotting

코드 설명 - DC 입력

```python
import numpy as np
import matplotlib.pyplot as plt
import ipywidgets as widgets
from ipywidgets import link
from IPython.display import display, clear_output, Math
from math import sqrt, pi
from scipy.integrate import solve_ivp
# --- core formulas ---
def omega0(L, C): return 1.0/np.sqrt(L*C)
def zeta(R, L, C): return (R/2.0)*np.sqrt(C/L)
def compute_poles(R, L, C):
    w0, z = omega0(L, C), zeta(R, L, C)
    if z < 1.0:
        wd = w0*np.sqrt(1.0 - z*z)
        return complex(-z*w0, +wd), complex(-z*w0, -wd)
    # z >= 1: real poles
    root = np.sqrt(max(0.0, z*z - 1.0))
    s1 = -w0*(z - root)
    s2 = -w0*(z + root)
    return complex(s1, 0.0), complex(s2, 0.0)
```

실근, 허근일 때 나눠 근을 구함

```python
def H_jw(R, L, C, w):
    Zc = 1.0/(1j*w*C)
    return Zc / (R + 1j*w*L + Zc)    # Vc/Vin
```

전달함수 계산

ODE solver 통해 Vc 파형 얻음

```python
# --- simulations ---
# Simulate RLC step response using ODE solver
def simulate_step_Vc_only(R, L, C, V_step, t_end, n=3000):
    if t_end <= 0: return np.array([]), np.array([])
    def f(t, x):
        q, i = x
        return [i, (V_step - R*i - q/C)/L]
    t = np.linspace(0, t_end, n)
    sol = solve_ivp(f, [0, t_end], [0.0, 0.0], t_eval=t, max_step=t_end/max(n,1))
    if not sol.success: return t, np.full_like(t, np.nan)
    return t, sol.y[0] / C  # Vc = q/C
```

주어진 꼴을 이용해 Vc 계산

```python
# Analytic solution for RLC step response
def analytic_Vc(Vin, R, L, C, t):
    w0, z = omega0(L, C), zeta(R, L, C)
    if z < 1.0:
        wd  = w0*np.sqrt(1.0 - z*z)
        A   = 1.0/np.sqrt(1.0 - z*z)
        phi = np.arccos(z)
        return Vin*(1 - A*np.exp(-z*w0*t)*np.sin(wd*t + phi))
    elif abs(z-1.0) <= 1e-12:
        return Vin*(1 - (1 + w0*t)*np.exp(-w0*t))
    else:
        s1 = -w0*(z - np.sqrt(z*z - 1.0))
        s2 = -w0*(z + np.sqrt(z*z - 1.0))
        return Vin*(1 - ((s2*np.exp(s1*t) - s1*np.exp(s2*t))/(s2 - s1)))
# --- plotting ranges  ---
# choose time samples based on t_end
def choose_time_samples(t_end):
    return 1500 if t_end < 5e-3 else 3000 if t_end < 5e-2 else 5000 if t_end < 0.2 else 8000

# choose frequency sweep based on w0 and z
def choose_freq_sweep(w0, z):
    span = 100.0 if (z < 0.5) else 40.0 if (z < 1.0) else 20.0
    wmin = max(1e-3, w0/span); wmax = w0*span
    Nf   = 1200 if span >= 40 else 800
    return np.logspace(np.log10(wmin), np.log10(wmax), Nf)

# --- formatting float number ---
def _fmt(x, precision=6):
    s = np.format_float_positional(float(x), precision=precision, trim='-')
    if s.startswith('-0') and round(float(x), precision) == 0.0:
        s = '0'
    return s
```

코드 설명 - DC 입력

```python
# Display numeric formulas
def show_numeric_formulas(R, L, C, Vin, s1, s2):
    w0 = omega0(L, C)
    z  = zeta(R, L, C)
    def tex(s):
        display(Math(s))

    # UNDERDAMPED: ζ < 1 \
    if z < 1.0:
        wd  = abs(s1.imag)                    # damped natural freq
        A   = 1.0/np.sqrt(1.0 - z*z)          # amplitude scale in step response
        phi = np.arccos(z)                    # phase shift
        Q   = 1.0/(2.0*z)                     # quality factor
        Mp  = np.exp(-np.pi*z/np.sqrt(1.0 - z*z))  # overshoot fraction
        Ts  = 4.0/(z*w0)                      # ~ settling time

        # 상태 + 핵심 파라미터
        tex(
            r"\textbf{UNDERDAMPED }(\zeta<1)"
            + r"\quad \zeta=" + _fmt(z)
            + r",\ \omega_0=" + _fmt(w0) + r"\ \text{rad/s}"
            + r",\ Q=" + _fmt(Q)
        )
        # 시간 응답 Vc(t)
        tex(
            r"V_C(t)=" + _fmt(Vin)
            + r"\Big[1-"
            + _fmt(A)
            + r" e^{-("+ _fmt(z*w0) + r")t}"
            + r"\sin(" + _fmt(wd) + r"t+" + _fmt(phi) + r")\Big]"
        )
        # 핵심 성능 지표
        tex(
            r"M_p \approx "
            + f"{Mp*100:.2f}"
            + r"\% \quad,\quad "
            + r"T_s \approx "
            + _fmt(Ts)
            + r"\ \text{s}"
        )
```

Zeta 값 기준으로 damping 구분

Underdamping 상황에서 성능 지표 계산 후 출력

```python
    # CRITICALLY DAMPED: ζ = 1 → 중복 실근, 진동 없이 가장 빠르게 감쇠
    elif abs(z - 1.0) < 0.02: # zeta가 1에 매우 근접한 경우
        tau = 1.0/w0  # dominant time constant
        tex(
            r"\textbf{CRITICALLY DAMPED }(\zeta=1)"
            + r"\quad \zeta=" + _fmt(z)
            + r",\ \omega_0=" + _fmt(w0) + r"\ \text{rad/s}"
            + r",\ \tau=" + _fmt(tau) + r"\ \text{s}"
        )
        tex(
            r"V_C(t)=" + _fmt(Vin)
            + r"\Big[1-(1+" + _fmt(w0)
            + r"t)e^{-(" + _fmt(w0) + r")t}\Big]"
        )
        tex(r"\text{no overshoot, fastest return}")
```

Critical damping 상황에서 Vc 출력

```python
    # OVERDAMPED: ζ > 1 → 서로 다른 두 실근, 진동 없이 느리게 감쇠
    else:
        s1r, s2r = s1.real, s2.real
        tau_slow = 1.0/abs(min(s1r, s2r, key=lambda x: abs(x)))
        tex(
            r"\textbf{OVERDAMPED }(\zeta>1)"
            + r"\quad \zeta=" + _fmt(z)
            + r",\ \omega_0=" + _fmt(w0) + r"\ \text{rad/s}"
            + r",\ \tau_{\text{slow}}\approx " + _fmt(tau_slow) + r"\ \text{s}"
        )
        tex(
            r"V_C(t)=" + _fmt(Vin)
            + r"\left[1-\frac{(" + _fmt(s2r) + r")e^{("
            + _fmt(s1r) + r")t}-(" + _fmt(s1r)
            + r")e^{(" + _fmt(s2r)
            + r")t}}{" + _fmt(s2r - s1r) + r"}\right]"
        )
        tex(r"\text{no oscillation, slow tail}")
```

Overdamping 상황에서 Vc 출력

코드 설명 - DC 입력

```python
# --- 입력 UI ---
# R (Ohm)
R_slider = widgets.FloatSlider(
    description='R (Ω)',
    min=0.1, max=100.0, step=0.1,
    value=10.0,
    readout=True,
    continuous_update=True,  # 드래그 중에도 업데이트
    layout=widgets.Layout(width='250px')
)
# L (H) - 슬라이더
L_slider = widgets.FloatLogSlider(
    description='L (H)',
    base=10,
    min=-6,  # 1e-6 H
    max=-1,  # 1e-1 H
    step=0.1,
    value=1e-3,
    readout=True,
    layout=widgets.Layout(width='250px')
)
# C (F) - 슬라이더
C_slider = widgets.FloatLogSlider(
    description='C (F)',
    base=10,
    min=-9,  # 1e-9 F
    max=-4,  # 1e-4 F
    step=0.1,
    value=1e-5,
    readout=True,
    layout=widgets.Layout(width='250px')
)

VIN_in  = widgets.FloatText(value=1.0, description='Vin_step [V]', layout=widgets.Layout(width='170px'))
TEND_in = widgets.FloatText(value=5e-3, description='t_end [s]', layout=widgets.Layout(width='150px'))
out     = widgets.Output()
```

R,L,C는 슬라이더로 입력받아 그래프에 실시간 반영

VIN과 t_end는 값으로 입력 받음

```python
def main(_=None):
    with out:
        clear_output()
        R = R_slider.value
        L = L_slider.value
        C = C_slider.value
        Vin, t_end = VIN_in.value, TEND_in.value
        if L<=0 or C<=0 or R<0 or t_end<=0:
            print("Invalid parameters: require L>0, C>0, R>=0, t_end>0."); return

        # (1) numeric formulas (with substituted values)
        s1, s2 = compute_poles(R, L, C)
        show_numeric_formulas(R, L, C, Vin, s1, s2)

        # (2) time response: numeric vs analytic
        n_time = choose_time_samples(t_end)
        t, Vc_num = simulate_step_Vc_only(R, L, C, Vin, t_end, n_time)
        Vc_ana = analytic_Vc(Vin, R, L, C, t)
        fig1, ax1 = plt.subplots(figsize=(6.6,3.2))
        ax1.plot(t, Vc_num, label='Numeric (solve_ivp)')
        ax1.plot(t, Vc_ana, '--', label='Analytic')
        ax1.set_title("Step response: Vc(t)")
        ax1.set_xlabel("t [s]"); ax1.set_ylabel("Vc [V]")
        ax1.grid(True, ls=':'); ax1.legend(); plt.tight_layout(); plt.show()
for w in [R_slider, L_slider, C_slider,
          VIN_in, TEND_in]:
    w.observe(main, names='value')

display(widgets.VBox([
    widgets.HTML("<h4>DC Step - R,L,C,Vin_step,t_end (compact)</h4>"),
    widgets.HBox([R_slider, L_slider,C_slider,
                  VIN_in, TEND_in]),
    out
]))
```

Main 함수에서 앞서 정의한 함수 호출하고, 그래프 plot

실행 결과 - DC 입력

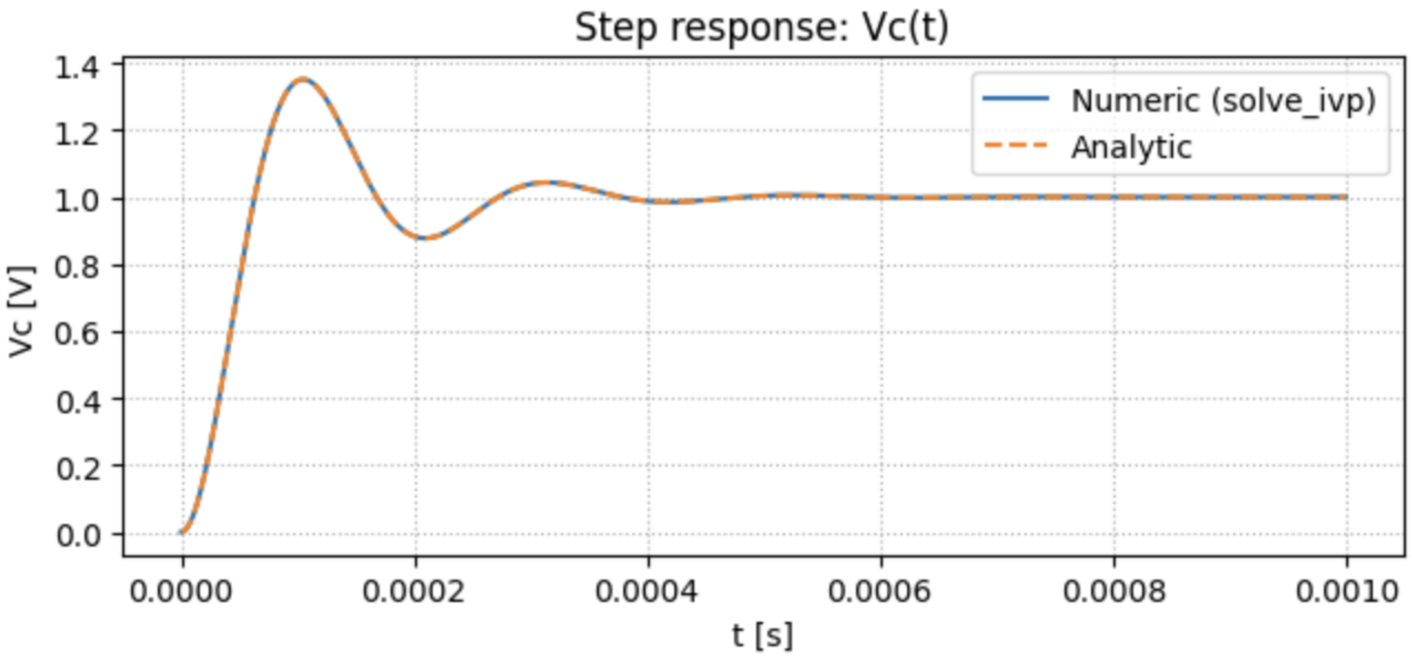
DC Step — R,L,C,Vin_step,t_end (compact)

R (Ω) 20.00 L (H) 0.00100 C (F) 0.00000100 Vin_step [V] 1 t_end [s] 0.001

$\textbf{UNDERDAMPED}\ (\zeta < 1) \quad \zeta = 0.316228,\ \omega_0 = 31622.776602\ \mathrm{rad/s},\ Q = 1.581139$

$V_C(t) = 1\left[1 - 1.054093e^{-(10000)t}\sin(30000t + 1.249046)\right]$

$M_p \approx 35.09\% \quad , \quad T_s \approx 0.0004\ \mathrm{s}$



실행 결과 - DC 입력

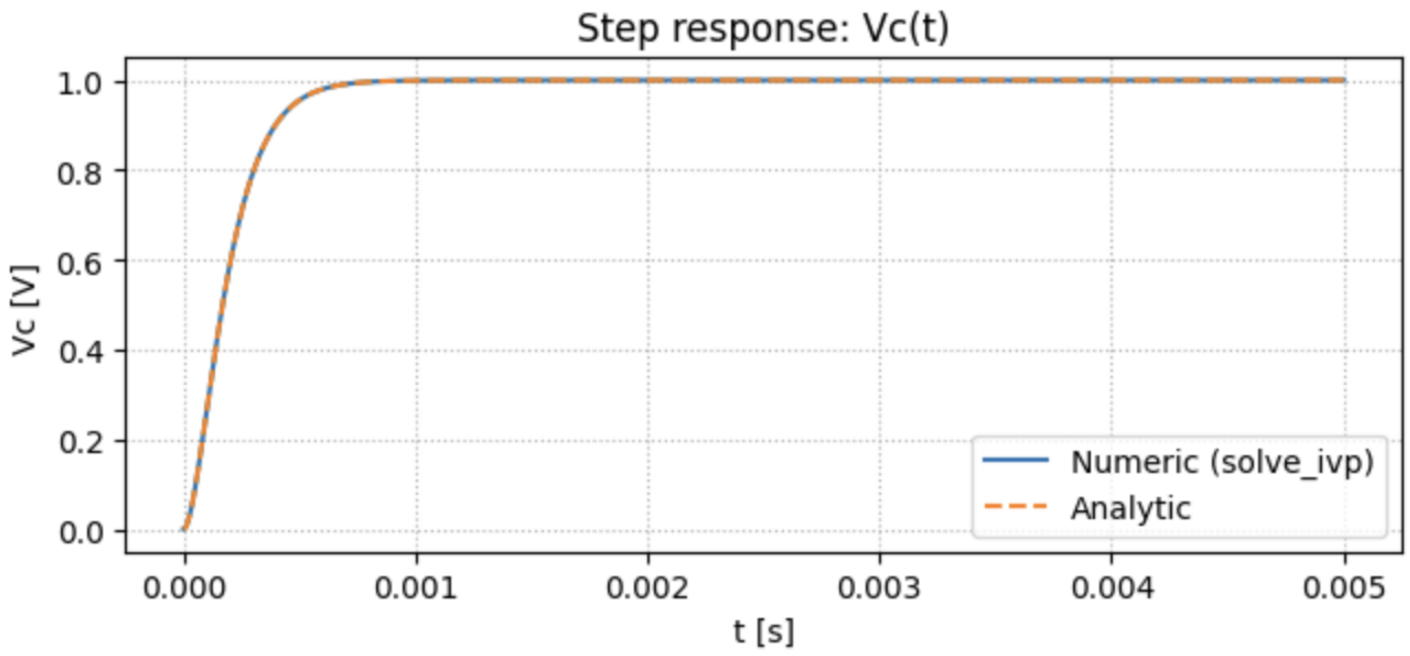
DC Step — R,L,C,Vin_step,t_end (compact)

R (Ω) 20.30 L (H) 0.00100 C (F) 0.0000100 Vin_step [V] 1 t_end [s] 0.005

$\mathbf{CRITICALLY\ DAMPED}\ (\zeta = 1) \quad \zeta = 1.015,\ \omega_0 = 10000\ \mathrm{rad/s},\ \tau = 0.0001\ \mathrm{s}$

$V_C(t) = 1\left[1 - (1 + 10000t)e^{-(10000)t}\right]$

no overshoot, fastest return



실행 결과 - DC 입력

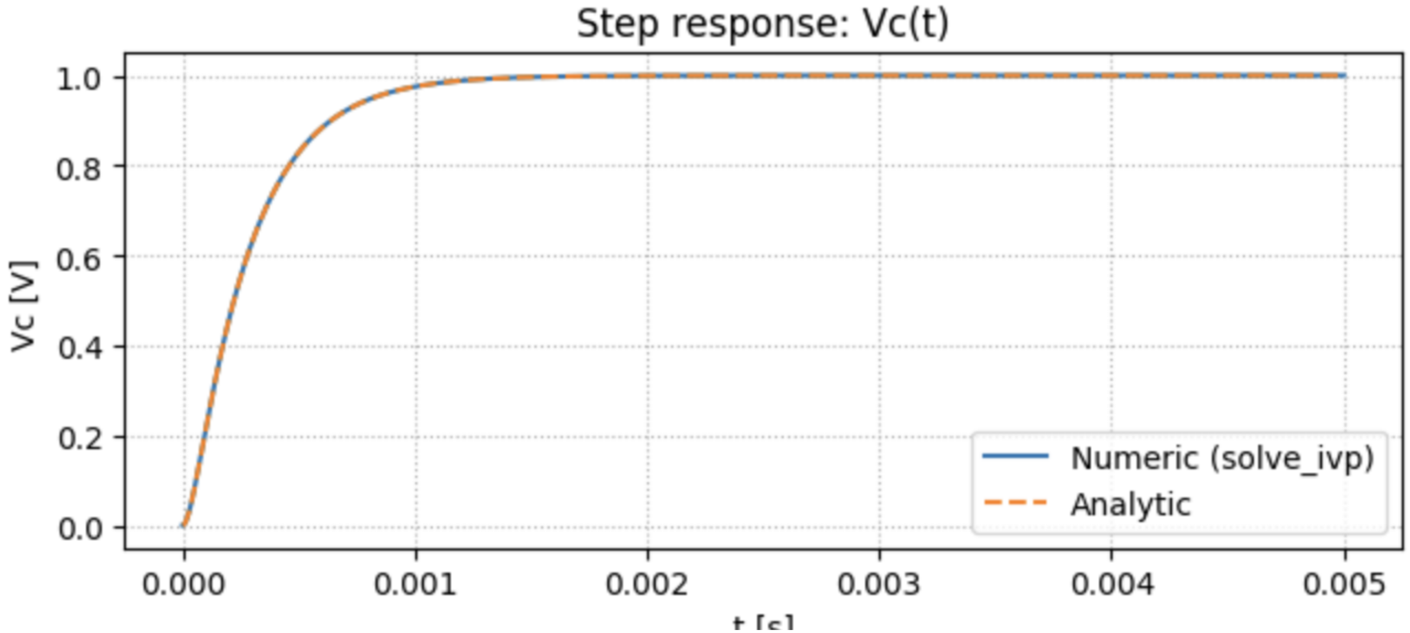
DC Step — R,L,C,Vin_step,t_end (compact)

R (Ω) 29.80 L (H) 0.00100 C (F) 0.0000100 Vin_step [V] 1 t_end [s] 0.005

$\textbf{OVERDAMPED}\ (\zeta > 1) \quad \zeta = 1.49,\ \omega_0 = 10000\ \mathrm{rad/s},\ \tau_{\mathrm{slow}} \approx 0.000259\ \mathrm{s}$

$$V_C(t) = 1\left[1 - \frac{(-25945.813687)e^{(-3854.186313)t} - (-3854.186313)e^{(-25945.813687)t}}{-22091.627373}\right]$$

no oscillation, slow tail



코드 설명 - AC 입력

```python
def show_ac_summary(R, L, C, Vac, w_drive):

    # 시스템 파라미터
    w0 = omega0(L, C)
    z  = zeta(R, L, C)
    # 전달함수 H(jω) = Vc/Vin
    H_drive = H_jw(R, L, C, w_drive)
    mag     = np.abs(H_drive)          # |H|
    phi     = np.angle(H_drive)        # [rad]

    def tex(s):
        display(Math(s))
    # 요약 텍스트
    tex(
        r"V_C^{\text{(steady)}}(t) \approx "
        + _fmt(mag * Vac)
        + r"\,\sin\!\big("
        + _fmt(w_drive)
        + r"t + "
        + _fmt(phi)
        + r"\big)"
    )
    tex(
        r"|H| = " + _fmt(mag)
        + r",\quad"
        r"\angle H = " + _fmt(phi)
        + r"\ \mathrm{rad}"
        + r"\ (" + _fmt(phi * 180/np.pi) + r"^\circ)"
    )
```

AC 입력시 파라미터,
전달함수 정의 후 출력

```python
def simulate_ac_Vc(R, L, C, Vac, w_drive, t_end, n=3000):
    """
    dq/dt = i
    di/dt = (Vin(t) - R i - q/C)/L
    Vin(t) = Vac * sin(w_drive * t)
    """
    def f(t, x):
        q, i = x
        Vin_t = Vac * np.sin(w_drive * t)
        dqdt = i
        didt = (Vin_t - R*i - q/C)/L
        return [dqdt, didt]
    # 시간축
    t = np.linspace(0, t_end, n)

    q0 = 0.0    # => Vc(0)=0
    i0 = 0.0
    sol = solve_ivp(
        f,
        [0, t_end],
        [q0, i0],
        t_eval=t,
        max_step=t_end/max(n, 1)
    )
    if not sol.success:
        return t, np.full_like(t, np.nan), np.full_like(t, np.nan)
    q = sol.y[0]
    Vc = q / C
    Vin_t = Vac * np.sin(w_drive * t)
    return t, Vc, Vin_t
```

ODE solver 이용해
Vc 파형 얻음

코드 설명 - AC 입력

(위젯 관련 코드 생략)

```
# --- (3) AC 전용 main 함수 ---
def main_ac(_=None):
    with out_ac:
        clear_output()
        # --- RLC / AC 파라미터 읽기 ---
        R   = R_slider.value
        L   = L_slider.value
        C   = C_slider.value
        Vac = VAC_in.value          # [V]   사인파 진폭
        f_drive = FREQ_slider.value  # [Hz]  구동 주파수
        w_drive = 2.0 * np.pi * f_drive   # [rad/s]
        t_end_ac = TAC_in.value      # [s]   시뮬레이션 길이
        # --- 유효성 검사 ---
        if L <= 0 or C <= 0 or R < 0:
            print("Invalid RLC: need L>0, C>0, R>=0.")
            return
        if t_end_ac <= 0 or f_drive < 0:
            print("Invalid AC params: need t_end>0 and f>=0.")
            return
        # --- 적당한 샘플 수 정하기
        n_time_ac = choose_time_samples(t_end_ac)
        # ==================================================
        # --- 정상상태(해석적) 표현을 먼저 보여주기
        show_ac_summary(R, L, C, Vac, w_drive)
        # --- 실제(수치) 시뮬레이션: 과도 + 정상상태 포함
        t_ac, Vc_ac, Vin_ac = simulate_ac_Vc(
            R, L, C,
            Vac,
            w_drive,
            t_end_ac,
            n_time_ac
        )
```

Main 함수에서 정의한 함수 불러오고 graph plot

```
        # (1) 전체 시간 파형: Vin vs Vc
        fig_full, ax_full = plt.subplots(figsize=(6.6, 3.2))
        ax_full.plot(t_ac, Vin_ac, label='Vin(t) = Vac·sin(ωt)')
        ax_full.plot(t_ac, Vc_ac, '--', label='Vc(t)')
        ax_full.set_title("AC driven response (pure sine) - full window")
        ax_full.set_xlabel("t [s]")
        ax_full.set_ylabel("Voltage [V]")
        ax_full.grid(True, ls=':')
        ax_full.legend()
        plt.tight_layout()
        plt.show()
```

파형 graph plot

```
        # (2) 주파수 응답 크기 (Bode magnitude)
        w0 = omega0(L, C)
        z  = zeta(R, L, C)
        ws = choose_freq_sweep(w0, z)      # log-spaced ω range
        Hs = H_jw(R, L, C, ws)
        mag_db = 20*np.log10(np.abs(Hs) + 1e-30)
        fig_mag, ax_mag = plt.subplots(figsize=(6.4, 3.2))
        ax_mag.semilogx(ws, mag_db)
        ax_mag.axvline(w0, alpha=0.35, ls='--', label='ω0')
        ax_mag.set_title("Frequency response: |Vc/Vin| [dB]")
        ax_mag.set_xlabel("ω [rad/s]")
        ax_mag.set_ylabel("20 log10 |H(jω)| [dB]")
        ax_mag.grid(True, which='both', ls=':')
        ax_mag.legend()
        plt.tight_layout()
        plt.show()
```

주파수 응답 (크기, 위상) plot

```
        # (3) 주파수 응답 위상 (Phase)
        phi = np.unwrap(np.angle(Hs))          # [rad]
        fig_phase, ax_phase = plt.subplots(figsize=(6.4, 3.2))
        ax_phase.semilogx(ws, np.degrees(phi))
        ax_phase.axvline(w0, alpha=0.35, ls='--', label='ω0 (~ -90°)')
        ax_phase.set_title("Phase response: ∠H(jω)")
        ax_phase.set_xlabel("ω [rad/s]")
        ax_phase.set_ylabel("phase [deg]")
        ax_phase.grid(True, which='both', ls=':')
        ax_phase.legend()
        plt.tight_layout()
        plt.show()
```

실행 결과 - AC 입력

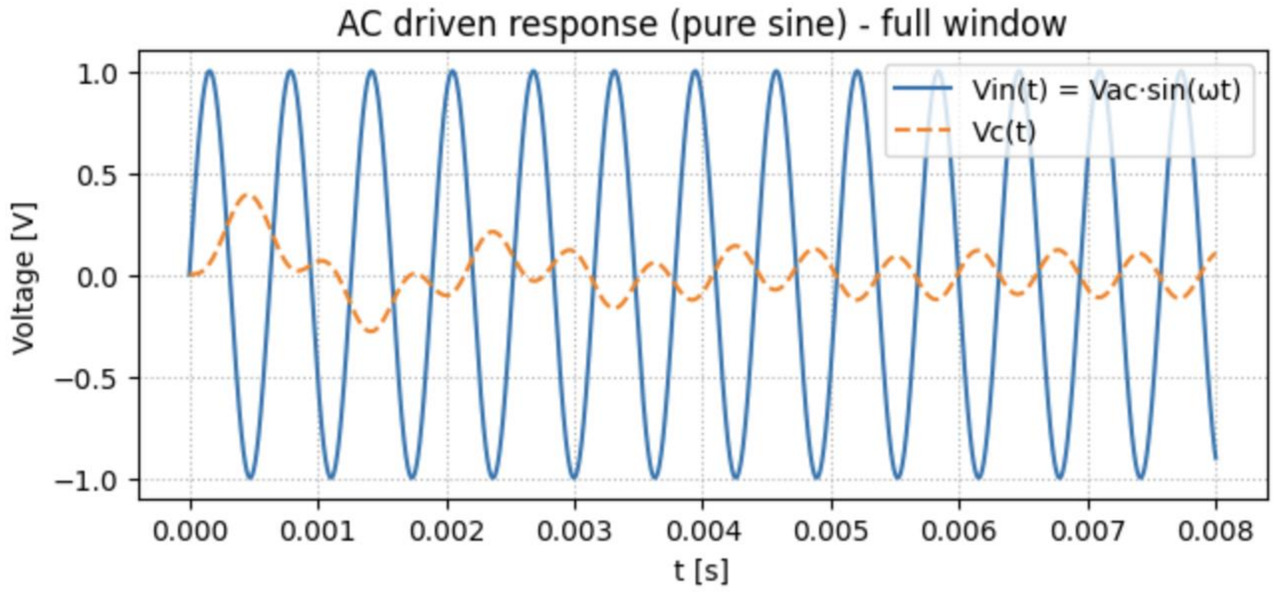
R=10Ω, L=10mH, C=10uF, ω_0 =3162.27766 rad/s, Q=3.162278

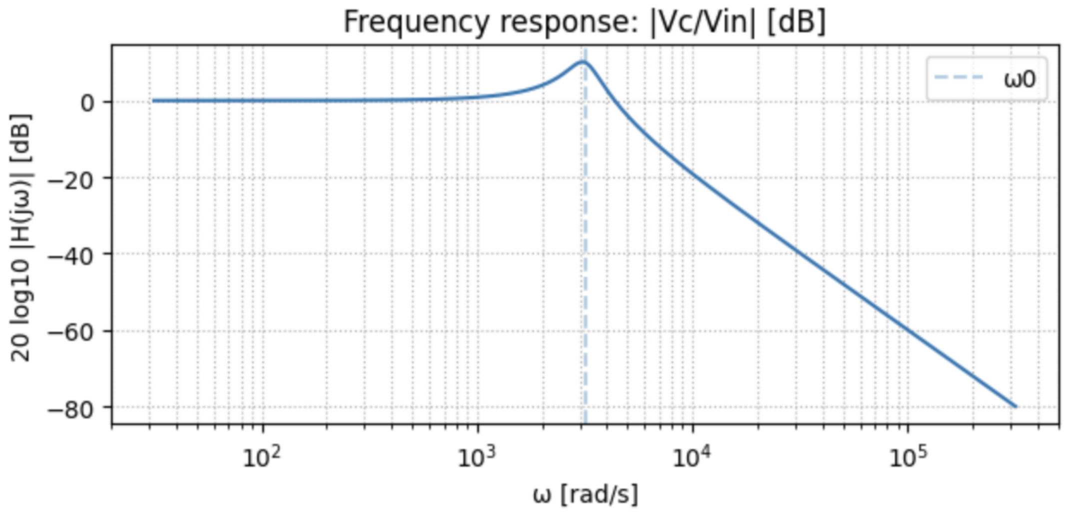
AC drive settings

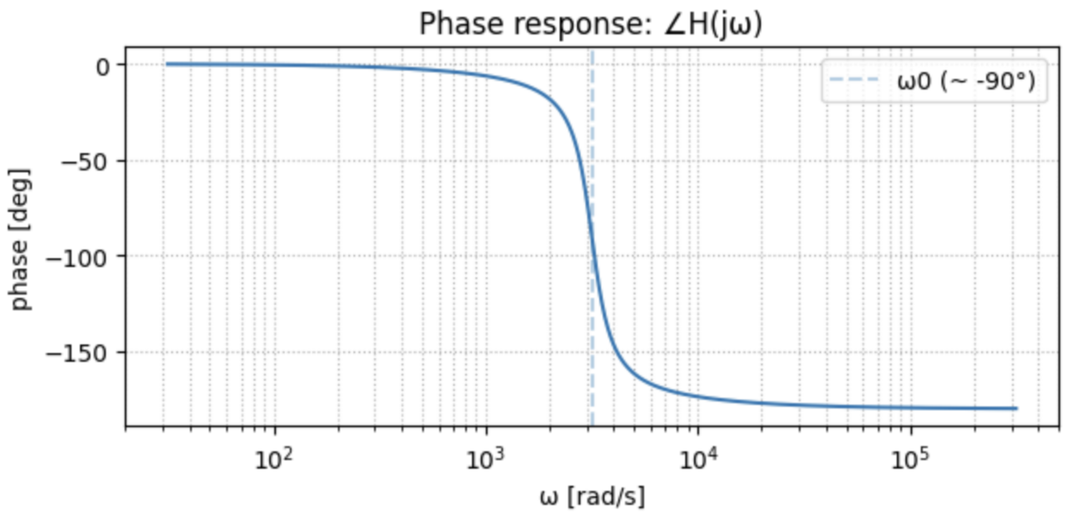
Vac [V] 1 f (Hz) 1.58e+3 t_end_AC [s] 0.008

$V_C^{(\mathrm{steady})}(t) \approx 0.111458 \ \sin\big(9958.17762t + -3.030371\big)$

$|H| = 0.111458, \quad \angle H = -3.030371 \ \mathrm{rad} \ (-173.627488^\circ)$







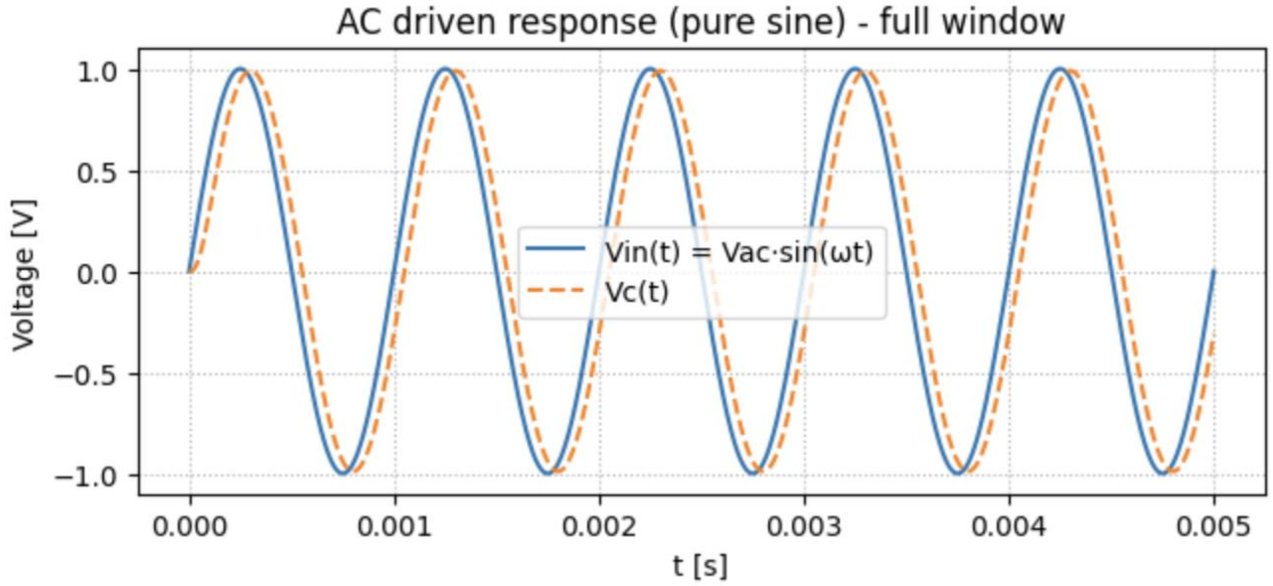
실행 결과 - AC 입력

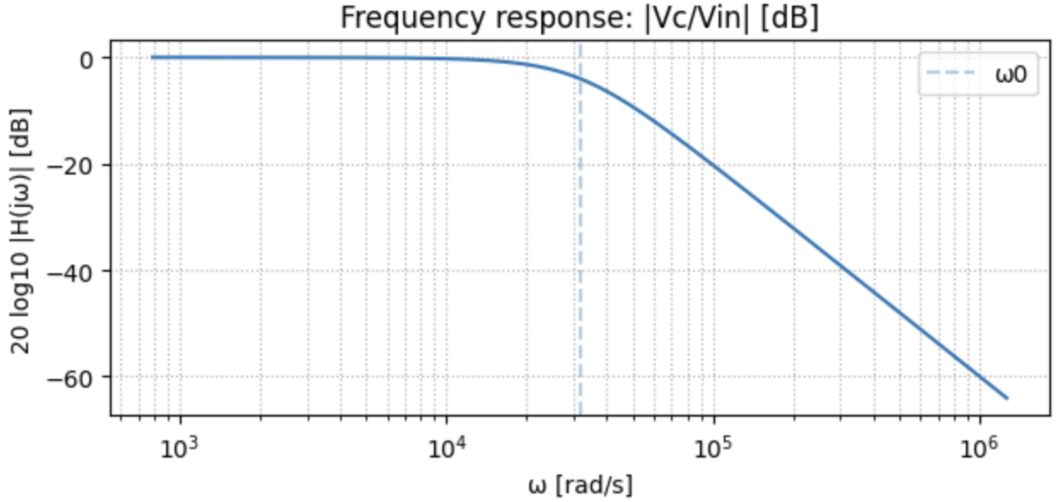
R=50Ω, L=1mH, C=0.1uF, ω_0 =31622.776602 rad/s, Q=0.632456

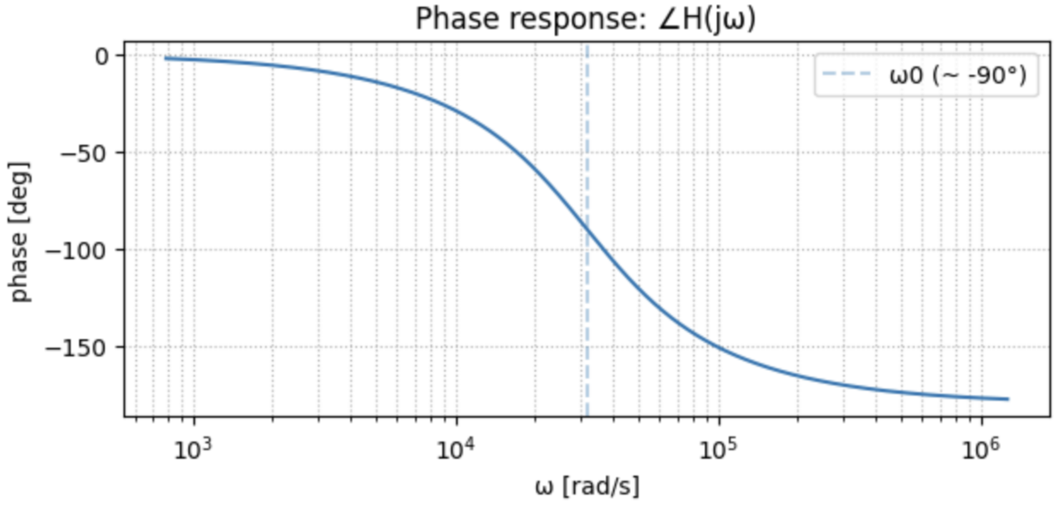
AC drive settings

Vac [V] 1 f (Hz) 1.00e+3 t_end_AC [s] 0.005

$$V_C^{(\text{steady})}(t) \approx 0.989518\ \sin\big(6283.185307t + -0.316104\big)$$
$$|H| = 0.989518, \quad \angle H = -0.316104\ \text{rad}\ (-18.111447^\circ)$$







결론

- 직렬 RLC 회로의 소자 값과 입력 전압을 입력했을 때 간단하게 Vc 값을 파악하고 Oscillation 종류를 구분할 수 있음

- 파형 그래프와 주파수 응답 그래프를 확인하고, Q 값에 따른 주파수 응답 변화를 쉽게 관찰할 수 있음

- 활용: oscillation 종류와 성능 지표를 입력하면 그에 맞는 회로를 구성하는 코드를 설계해 공학적인 도구로서 발전

참고문헌

- Thornton & Marion, *Classical Dynamics of Particles and Systems*, 5th ed., Cengage Learning, 2003, pp. 108-126.
- Irwin & Nelms, *Basic Engineering Circuit Analysis*, 12th ed., Wiley, 2021, pp. 532-558.
- Underdamped System, https://www.sciencedirect.com/topics/engineering/underdamped-system