System Dynamics Homework 5

November 17, 2023

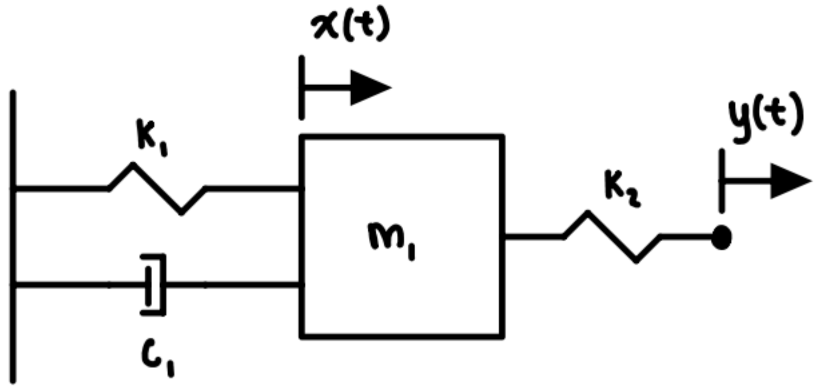
Gabe Morris

[1]:
```python
import control as ct
import sympy as sp
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import fsolve

plt.style.use('../maroon_ipynb.mplstyle')
```

Contents

1 Problem 1

1.1 Given



The mass above is being controlled by the input position $y(t)$. Take $k_1 = 10\,\frac{lbf}{in}$, $k_2 = 100\,\frac{lbf}{in}$, $c_1 = 1\,\frac{lbf\,s}{in}$, and $m_1 = 0.0518\,goobs$ where $1\,goob = \frac{lbs\,s^2}{in}$. The number comes from the mass weighing $20\,lbf$.

$$W = mg$$
$$20\,lbf = m \cdot 32.2\,\frac{ft}{s^2}$$
$$m = \frac{20}{32.2}\,slugs = 0.621\,\frac{lbf\,s^2}{ft} \cdot \frac{ft}{12\,in} = 0.0518\frac{lbf\,s^2}{in} = 0.0518\,goobs$$

1.2 Find

For $y(t) = 1.5\sin(\omega_r t)$,

a. Find the equation of motion.
b. Find the transfer function $\frac{X(s)}{Y(s)}$.
c. Plot the Magnitude ($M(\omega)$ - not in decibels) and Phase Response for $1 \leq \omega \leq 1000\,rad/s$. Use the `bode()` function for checking. Note, the `bode()` function will produce a log-log plot on the y and x-axis, so the usual plot of $M(\omega)$ will look different, since we use a linear scale for the y-axis.
d. Find the resonant frequency ω_r.
e. Find the steady state function $x_{ss}(t)$ at the resonant frequency and plot the forced response on top of the steady state response up to 1 second.

1.3 Solution

1.3.1 Part A - Finding the Equation of Motion

[2]:
```
t, s, k1, k2, c1, m1 = sp.symbols('t s k1 k2 c1 m1')
x, y = sp.Function('x')(t), sp.Function('y')(t)

k1_, k2_ = 10, 100  # lbf/in
c1_ = 1  # lbs*s/in
m1_ = 0.0518  # goobs

eq = sp.Eq(m1*x.diff(t, 2), -k1*x - c1*x.diff() + k2*(y - x))
eq
```

[2]:

$$m_1 \frac{d^2}{dt^2} x(t) = -c_1 \frac{d}{dt} x(t) - k_1 x(t) + k_2 \left(-x(t) + y(t)\right)$$

1.3.2 Part B - Finding the Transfer Function

[3]:
```
lp = lambda expr: sp.laplace_transform(expr, t, s)[0]
eq_s = sp.Eq(lp(eq.lhs), lp(eq.rhs.expand()))

sub_ics = [
    (x.subs(t, 0), 0),
    (x.diff().subs(t, 0), 0)
]

eq_s = eq_s.subs(sub_ics)
eq_s
```

[3]:

$$m_1 s^2 \mathcal{L}_t\left[x(t)\right](s) = -c_1 s \mathcal{L}_t\left[x(t)\right](s) - k_1 \mathcal{L}_t\left[x(t)\right](s) - k_2 \mathcal{L}_t\left[x(t)\right](s) + k_2 \mathcal{L}_t\left[y(t)\right](s)$$

[4]:
```
T_s = sp.solve(eq_s, lp(x))[0]/lp(y)
T_s
```

[4]:

$$\frac{k_2}{c_1 s + k_1 + k_2 + m_1 s^2}$$

[5]:
```
T = ct.tf(k2_, [m1_, c1_, k1_ + k2_])
T
```

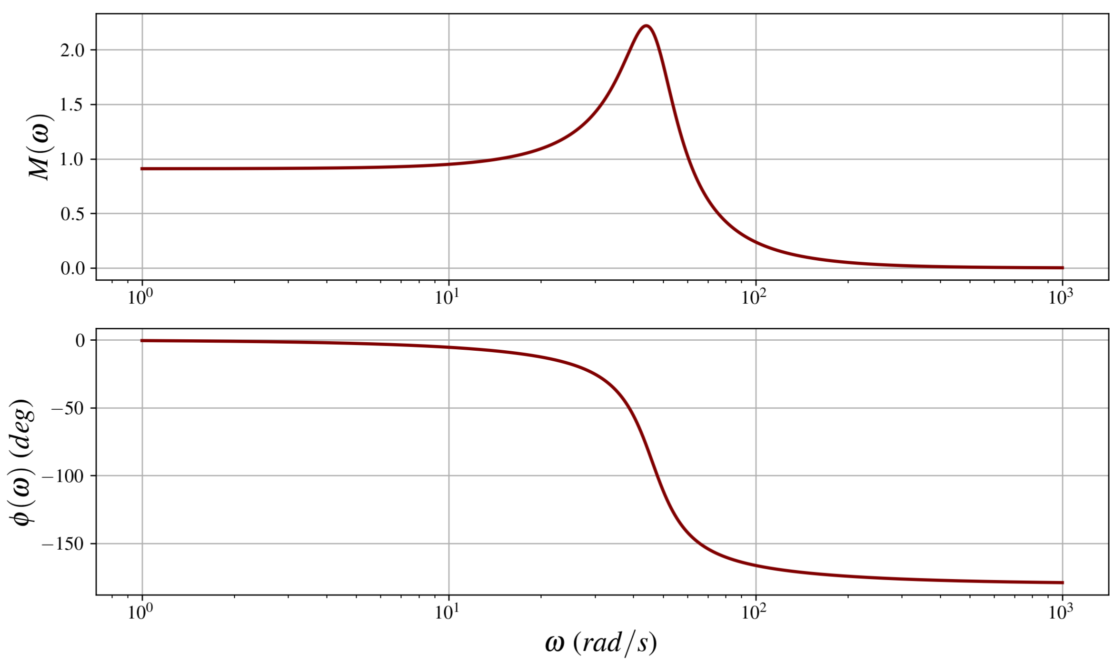
[5]:

$$\frac{100}{0.0518 s^2 + s + 110}$$

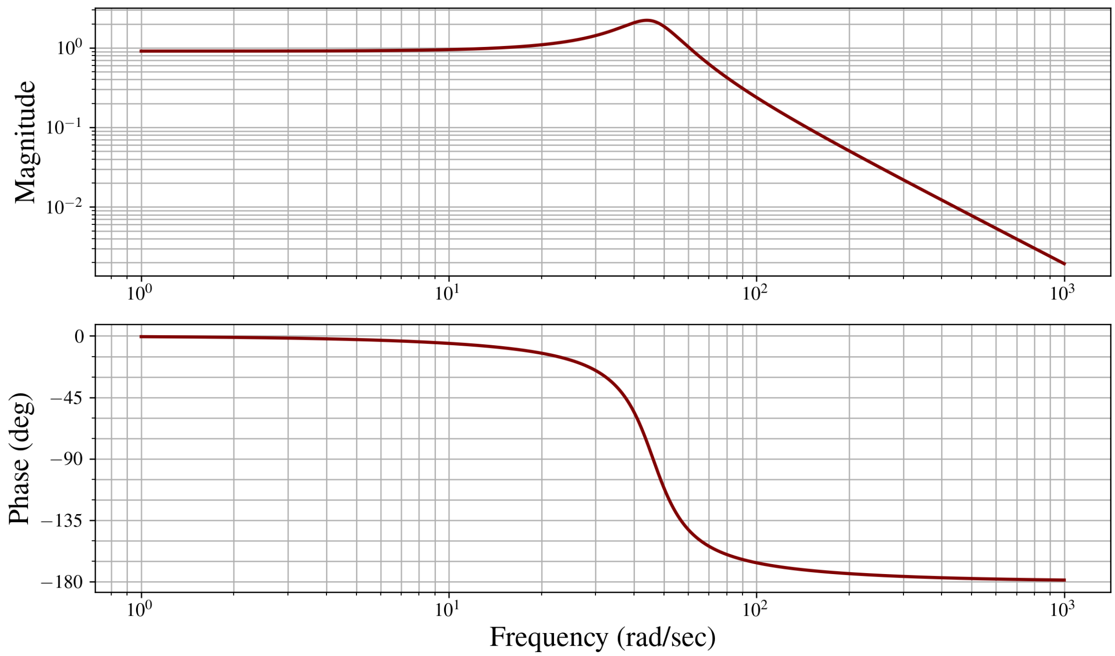
1.3.3 Part C - Plotting the Frequency Response

[6]:
```
omega = sp.Symbol(r'\omega')
T_jw = sp.lambdify(omega, T_s.subs([
    (s, sp.I*omega),
    (k1, k1_),
```

```python
    (k2, k2_),
    (c1, c1_),
    (m1, m1_)
]))

omegas = np.linspace(1, 1000, 100_000)
mags = np.abs(T_jw(omegas))
phase = np.angle(T_jw(omegas))

fig, (ax1, ax2) = plt.subplots(nrows=2, ncols=1)
ax1.set_xscale('log')
ax2.set_xscale('log')
# ax1.set_yscale('log')

ax1.plot(omegas, mags)
ax1.set_ylabel(r'$M(\omega)$')

ax2.plot(omegas, np.rad2deg(phase))
ax2.set_ylabel(r'$\phi(\omega)$ ($deg$)')
ax2.set_xlabel(r'$\omega$ ($rad/s$)')

# fig.savefig('Problem 1 Freq Response.png')
plt.show()
```



[7]:
```
# For checking
_ = ct.bode(T, omega=omegas)  # There is unwanted behavior to where the␣
↪magnitude y-axis is logarithmic. I checked and there is no changing it.
```



1.3.4 Part D - Finding the Resonant Frequency

[8]:
```
(wn, _), (zeta, _), (root, _) = ct.damp(T)
```

```
Eigenvalue (pole)       Damping     Frequency
-9.653    +45.06j        0.2095         46.08
-9.653    -45.06j        0.2095         46.08
```

[9]:
```
wr = wn*np.sqrt(1 - 2*zeta**2)
wr  # rad/s
```

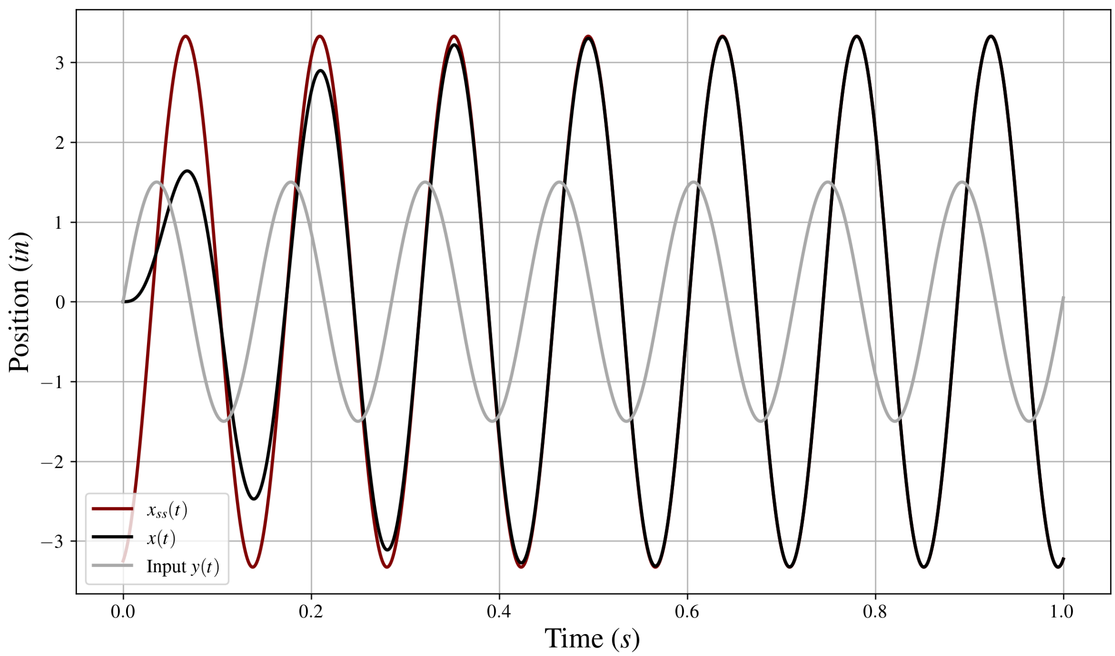
[9]: 44.01375056012086

[10]:
```
# Could also do something like this
omegas[max(mags) == mags][0]
```

[10]: 44.017370173701735

1.3.5 Part E - Finding the Response at the Resonant Frequency

[11]:
```
phi = np.angle(T_jw(wr))
B = 1.5*np.abs(T_jw(wr))
x_ss = lambda t_: B*np.sin(wr*t_ + phi)

t_array = np.linspace(0, 1, 1000)
_, x_ = ct.forced_response(T, T=t_array, U=1.5*np.sin(wr*t_array))

fig, ax = plt.subplots()

ax.plot(t_array, x_ss(t_array), label='$x_{ss}(t)$')
ax.plot(t_array, x_, label='$x(t)$')
ax.plot(t_array, 1.5*np.sin(wr*t_array), label='Input $y(t)$')

ax.legend()
ax.set_xlabel('Time ($s$)')
ax.set_ylabel('Position ($in$)')
# fig.savefig('Problem 1 Response.png')
plt.show()
```



2 Problem 2

2.1 Given

A certain series RLC circuit has the following transfer function.

$$T(s) = \frac{I(s)}{V(s)} = \frac{Cs}{LCs^2 + RCs + 1}$$

Suppose that $L = 300\,H$, $R = 10^4\,\Omega$, and $C = 10^{-6}\,F$.

2.2 Find

Determine the filtering properties by

a. Deriving the equations for the magnitude and the phase.
b. Plot the magnitude ($m(\omega)$) and phase and use the analytical solution above for checking.
c. Find the bandwidth from ω_1 to ω_2. What kind of filter is this (i.e. low-pass, band-pass, or high-pass)? Note: The relationship for finding ω_r is not valid for this example because of the presence of the s in the numerator of the transfer function. So, just find the peak M_{peak} by using the `max()` function.

2.3 Solution

2.3.1 Part A - Finding Magnitude and Phase Equations

[12]:
```
C, L, R, s, w = sp.symbols(r'C L R s \omega', positive=True, real=True)
C_, L_, R_ = 1e-6, 300, 1e4

sub_values = [
    (L, 300),
    (R, sp.S('1e4')),
    (C, sp.S('1e-6'))
]

T_s = C*s/(L*C*s**2 + R*C*s + 1)
T_s
```

[12]:
$$\frac{Cs}{CLs^2 + CRs + 1}$$

[13]:
```
T_jw = T_s.subs(s, sp.I*w)
T_jw
```

[13]:
$$\frac{iC\omega}{-CL\omega^2 + iCR\omega + 1}$$

[14]:
```python
num, den = sp.fraction(T_jw)
num_im = sp.im(num)
den_re = sp.re(den)
den_im = sp.im(den)

mag = num_im/sp.sqrt(den_re**2 + den_im**2)
mag
```

[14]:
$$\frac{C\omega}{\sqrt{C^2R^2\omega^2 + \left(-CL\omega^2 + 1\right)^2}}$$

[15]:
```python
# The numeric result
mag.subs(sub_values)
```

[15]:
$$\frac{1.0 \cdot 10^{-6}\omega}{\sqrt{0.0001\omega^2 + \left(1 - 0.0003\omega^2\right)^2}}$$

For the phase angle,

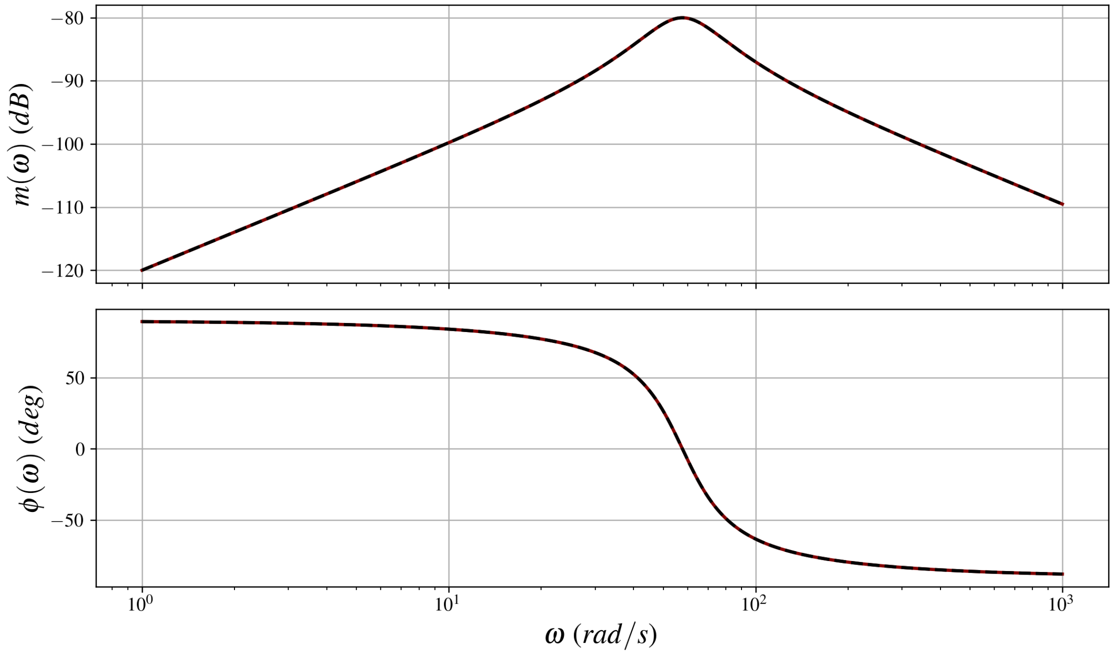
$$\angle T_{j\omega} = \tan^{-1}(C\omega/0) - \tan^{-1}\left(\frac{CR\omega}{1 - CL\omega^2}\right)$$
$$\angle T_{j\omega} = \frac{\pi}{2} - \tan^{-1}\left(\frac{CR\omega}{1 - CL\omega^2}\right)$$

Making sure the angle is properly returned, we use the atan2 function instead.

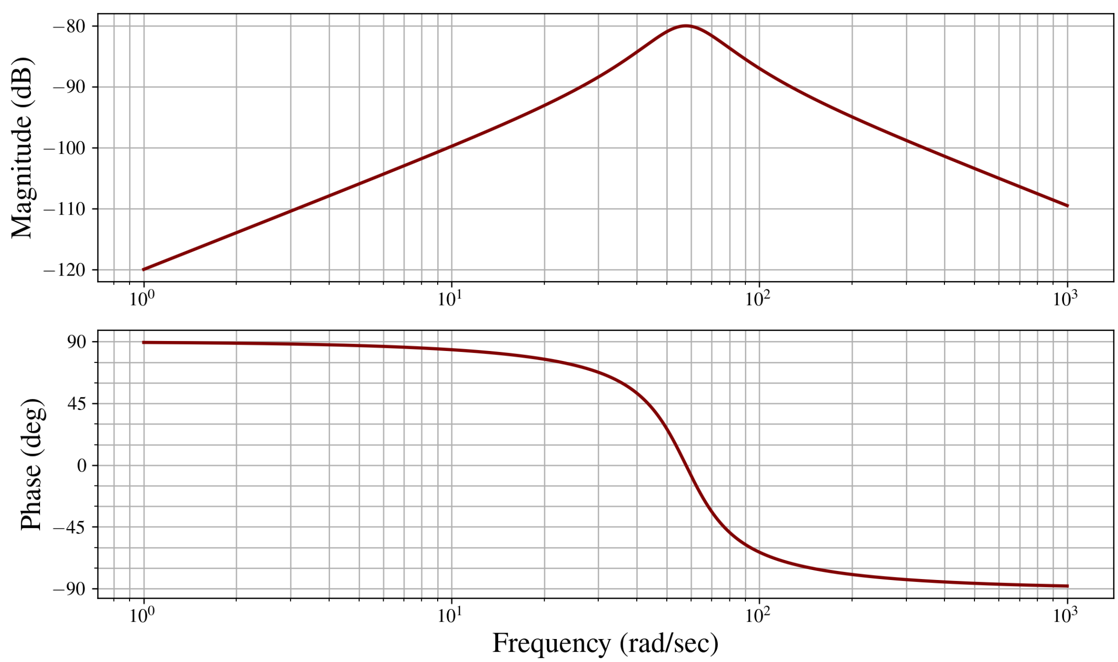
2.3.2 Part B - Plotting the Frequency Response

[16]:
```python
mag_lamb = sp.lambdify(w, mag.subs(sub_values), modules='numpy')

omegas = np.linspace(1, 1000, 100_000)
T_jw = lambda w_: C_*1j*w_/(C_*L_*(1j*w_)**2 + C_*R_*1j*w_ + 1)
c_nums = T_jw(omegas)
mags = np.abs(c_nums)
phase = np.angle(c_nums)
phase_ana = np.pi/2 - np.arctan2(C_*R_*omegas, 1 - C_*L_*omegas**2)

fig, (ax1, ax2) = plt.subplots(nrows=2, ncols=1, sharex=True)
ax1.set_xscale('log')

ax1.plot(omegas, 20*np.log10(mags))
ax1.plot(omegas, 20*np.log10(mag_lamb(omegas)), ls='--')
ax1.set_ylabel(r'$m(\omega)$ ($dB$)')

ax2.plot(omegas, np.rad2deg(phase))
ax2.plot(omegas, np.rad2deg(phase_ana), ls='--')
ax2.set_ylabel(r'$\phi(\omega)$ ($deg$)')
ax2.set_xlabel(r'$\omega$ ($rad/s$)')
```

```
# fig.savefig('Problem 2 Freq Response.png')
plt.show()
```




```
[17]: T = ct.tf([C_, 0], [C_*L_, C_*R_, 1])
      _ = ct.bode(T, omega=omegas, dB=True, wrap_phase=True)
```



2.3.3 Part C - Finding the Bandwidth

[18]:
```
M_peak = max(mags)
wr = omegas[M_peak == mags][0]
wr  # rad/s
```

[18]: 57.73377733777338

[19]:
```
def find_band(w_):
    M = mag_lamb(w_)[0]
    return M - M_peak/np.sqrt(2)


w1 = fsolve(find_band, np.array([wr - 1, ]))[0]
w1  # rad/s
```

[19]: 43.425854523361906

[20]:
```
w2 = fsolve(find_band, np.array([wr + 1, ]))[0]
w2
```

[20]: 76.759188044074

[21]:
```
# You can check to see that the distance from the m_peak to the end points is 3.
↪01
20*np.log10(M_peak) - 20*np.log10(mag_lamb(w1))
```

[21]: 3.010299956639841

Thus, the bandwidth is $43\, rad/s \leq \omega < 77\, rad/s$. This is a band-pass filter because $\omega_1 > 0$.