Vibrations and Controls Homework 10

April 6, 2022

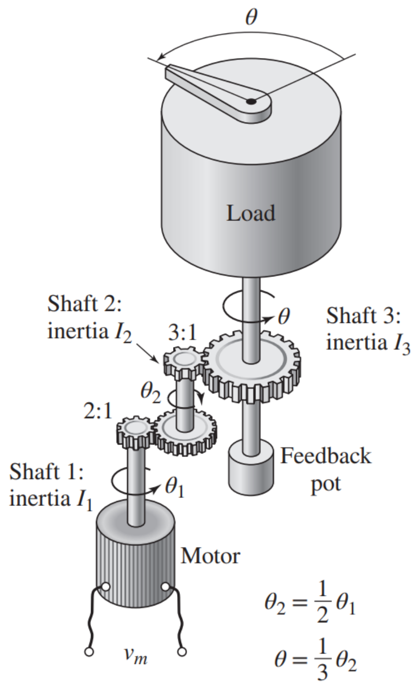
Gabe Morris

[1]:
```python
# Notebook Preamble
%config ZMQInteractiveShell.ast_node_interactivity = 'all'
import sympy as sp
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import odeint

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

t, s = sp.symbols('t s')
K_pot, K_P, K_d, K_a, K_T, L, R, I_e = sp.symbols(r'K_{pot} K_P K_d K_a K_T L R␣
↪I_e')
Th, Th_r, T = sp.Function(r'\theta')(s), sp.Function(r'\theta_r')(s), sp.
↪Function('T')(s)
```

Contents

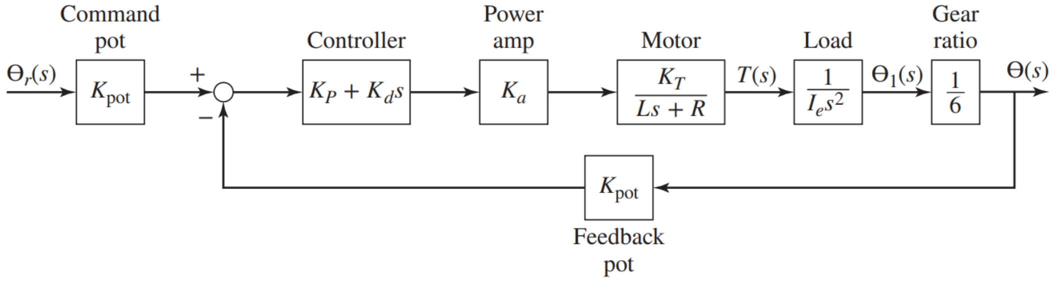
1 Problem 10.67

1.1 Given



The figure above shows a system for controlling the angular position of a load, such as an antenna. The figure below shows the block diagram for PD control of this system using a field-controlled motor. Use the following values:

- $K_a = 1\frac{V}{V}$
- $K_{pot} = 2\frac{V}{rad}$
- $I_2 = 5 \times 10^{-4}\, kg \cdot m^2$
- $R = 0.3\,\Omega$
- $K_T = 0.6\, N \cdot m/A$
- $I_1 = 0.01\, kg \cdot m^2$
- $I_3 = 0.2\, kg \cdot m^2$



The inertia I_e in the block diagram is the equivalent inertia of the entire system, as felt on the motor shaft.

1.2 Find

a. Assume that the motor inductance is very small and set $L = 0$. Compute I_e, obtain the transfer function $\frac{\theta(s)}{\theta_r(s)}$, and compute the values of the control gains K_P and K_d to meet the following specifications: $\zeta = 1$ and $\tau = 0.5\,s$.
b. Using the values of K_P and K_d computed in part (a), and the value $L = 0.015\,H$, obtain the transfer function $\frac{\theta(s)}{\theta_r(s)}$.

1.3 Solution

[2]:
```
# Define the givens here
K_a_, K_pot_, K_T_ = 1, 2, 0.6
I1, I2, I3 = 0.01, 5e-4, 0.2
R_ = 0.3
```

1.3.1 Part A

I have become untethered when it comes to calculating the equivalent inertia. The relationship is,

$$I_e = I_1 + \frac{1}{4}I_2 + \frac{1}{36}I_3$$

[3]:
```
# Getting Ie
Ie_ = I1 + sp.Rational(1, 4)*I2 + sp.Rational(1, 36)*I3
Ie_  # kg m^2
```

[3]: 0.0156805555555556

The transfer function may be obtained by using algebra from the block diagram, then solving.

[4]:
```
# Getting the transfer function
eq1 = sp.Eq((Th_r*K_pot - K_pot*Th)*(K_P + K_d*s)*K_a*K_T/(L*s + R)*1/
↪(I_e*s**2)*1/6, Th)
eq1
T_sol = sp.solve(eq1.subs(Th, T*Th_r), T)[0]
sp.Eq(Th/Th_r, T_sol)
```

[4]: $$\frac{K_T K_a (K_P + K_d s)\left(-K_{pot}\theta(s) + K_{pot}\theta_r(s)\right)}{6 I_e s^2 (Ls + R)} = \theta(s)$$

[4]: $$\frac{\theta(s)}{\theta_r(s)} = \frac{K_T K_a K_{pot} (K_P + K_d s)}{6 I_e L s^3 + 6 I_e R s^2 + K_P K_T K_a K_{pot} + K_T K_a K_d K_{pot} s}$$

Now, the denominator is the characteristic equation.

[5]:
```
_, d = sp.fraction(T_sol)
poly = d.subs(L, 0)
poly
```

```python
m, c, k = sp.Poly(poly, s).coeffs()
eq1 = sp.Eq(0.5, 2*m/c)
eq2 = sp.Eq(1, c/(2*sp.sqrt(m*k)))
eq1
eq2
```

[5]:
$$6I_eRs^2 + K_PK_TK_aK_{pot} + K_TK_aK_dK_{pot}s$$

[5]:
$$0.5 = \frac{12I_eR}{K_TK_aK_dK_{pot}}$$

[5]:
$$1 = \frac{\sqrt{6}K_TK_aK_dK_{pot}}{12\sqrt{I_eK_PK_TK_aK_{pot}R}}$$

Now substituting in the values, the gains may be acquired.

[6]:
```python
sub_list = [(I_e, Ie_), (R, R_), (K_T, K_T_), (K_a, K_a_), (K_pot, K_pot_)]

gain_sol = sp.solve([eq1, eq2], (K_P, K_d), dict=True)[0]
sp.Eq(K_d, gain_sol[K_d])
sp.Eq(K_P, gain_sol[K_P])
K_P_, K_d_ = gain_sol[K_P].subs(sub_list), gain_sol[K_d].subs(sub_list)
sp.Eq(K_P, K_P_)
sp.Eq(K_d, K_d_)
```

[6]:
$$K_d = \frac{24.0I_eR}{K_TK_aK_{pot}}$$

[6]:
$$K_P = \frac{24.0I_eR}{K_TK_aK_{pot}}$$

[6]:
$$K_P = 0.0940833333333333$$

[6]:
$$K_d = 0.0940833333333333$$

The transfer function and solution with $L = 0\,H$ is,

[7]:
```python
sub_0 = sub_list + [(K_d, K_d_), (K_P, K_P_), (L, 0)]
sub_15 = sub_list + [(K_d, K_d_), (K_P, K_P_), (L, 0.015)]
T_sol_ = T_sol.subs(sub_0)
sp.Eq(Th/Th_r, T_sol_.simplify())
Th_s = T_sol_*1/s
sp.Eq(sp.Eq(Th, Th_s.simplify()), 4*(s + 1)/(s*(s**2 + 4*s + 4)),
  ↪evaluate=False)
Th_s = 4*(s + 1)/(s*(s**2 + 4*s + 4))
th_tA = (sp.inverse_laplace_transform(Th_s, s, t)/sp.Heaviside(t)).expand()
sp.Eq(sp.Function(r'\theta')(t), th_tA)
```

[7]:
$$\frac{\theta(s)}{\theta_r(s)} = \frac{0.1129\,(s+1)}{0.028225s^2 + 0.1129s + 0.1129}$$

[7]:
$$\theta(s) = \frac{0.1129\,(s+1)}{s\,(0.028225s^2 + 0.1129s + 0.1129)} = \frac{4s+4}{s\,(s^2+4s+4)}$$

[7]:
$$\theta(t) = 2te^{-2t} + 1 - e^{-2t}$$

1.3.2 Part B

With $L = 0.015\,H$ and everything else staying the same, the transfer function is,

[8]:
```
%config ZMQInteractiveShell.ast_node_interactivity = 'last_expr'
T_sol_B = T_sol.subs(sub_15)
sp.Eq(sp.Eq(Th/Th_r, T_sol_B.simplify()), 80*(s + 1)/(s**3 + 20*s**2 + 80*s +␣
↪80), evaluate=False)
T_sol_B = 80*(s + 1)/(s**3 + 20*s**2 + 80*s + 80)
Th_s = T_sol_B*1/s
sp.Eq(Th, Th_s)
# sp.inverse_laplace_transform(sp.apart(Th_s).expand(), s, t)
# th_tB = sp.inverse_laplace_transform(Th_s, s, t)
# th_tB
th_tB = 1.16*sp.exp(-1.56*t) - 2.64*sp.exp(-3.4*t) + 0.477*sp.exp(-15*t) + 1
sp.Eq(sp.Function(r'\theta')(t), th_tB)
```

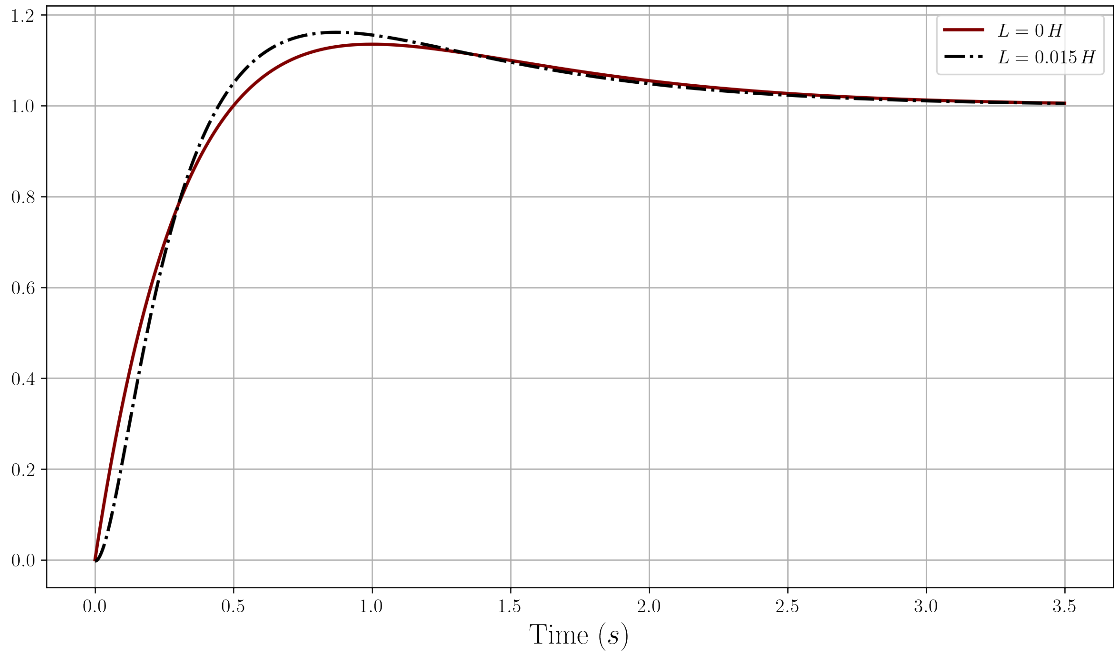
[8]:
$$\frac{\theta(s)}{\theta_r(s)} = \frac{0.1129\,(s+1)}{0.00141125s^3 + 0.028225s^2 + 0.1129s + 0.1129} = \frac{80s+80}{s^3+20s^2+80s+80}$$

[8]:
$$\theta(s) = \frac{80s+80}{s\,(s^3+20s^2+80s+80)}$$

[8]:
$$\theta(t) = 1 - 2.64e^{-3.4t} + 1.16e^{-1.56t} + 0.477e^{-15t}$$

For verification, here is a plot.

[9]:
```
# Plotting
time = np.linspace(0, 3.5, 1000)
th_tA_lamb, th_tB_lamb = sp.lambdify(t, th_tA, modules='numpy'), sp.lambdify(t,␣
↪th_tB, modules='numpy')
plt.plot(time, th_tA_lamb(time), label=r'$L=0\,H$')
plt.plot(time, th_tB_lamb(time), label=r'$L=0.015\,H$', ls='-.')
plt.xlabel('Time ($s$)')
plt.legend()
plt.show()
```



$L = 0\,H$
$L = 0.015\,H$
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
Time (s)