Dynamical Systems Homework 2

February 10, 2025

Gabe Morris

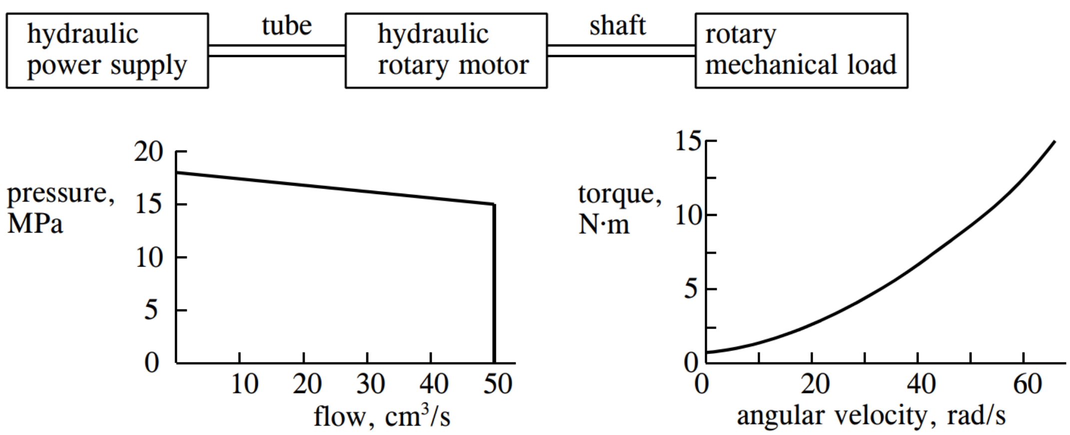
[1]:
```python
# toc
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import sympy as sp

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

Contents

Problem 1

Given

A hydraulic power supply (comprising a motor-driven pump and a relief valve) has the pressure-flow characteristic plotted on the left below. It drives a positive displacement hydraulic motor which in turn rotates a shaft that drives some mechanical equipment. Frictional and leakage losses in the hydraulic motor may be neglected. The characteristics of the mechanical load are plotted on the right below.

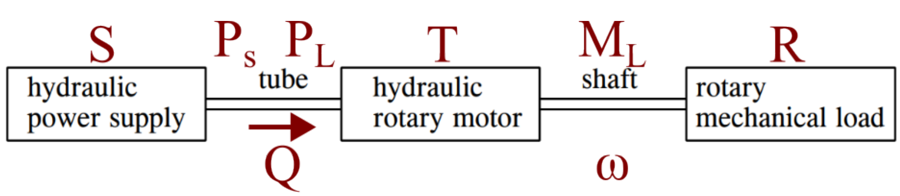


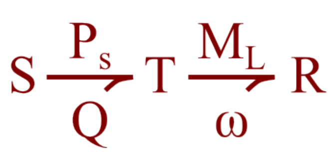
Find

a. Define key variables and place them on both the drawing and a bond-graph model of the system.
b. Find the maximum possible speed of the load.
c. Determine the volumetric displacement per revolution of the hydraulic motor.

Solution

Part A





In the above image, the source, S, causes a pressure difference, P_s, in the hydraulic rotary motor (the transformer T). This pressure difference causes a flow rate, Q, in the motor. The motor then induces a moment, M_L, on a shaft connected to the mechanical load, R. This in turn causes the load to rotate at a speed, ω. The load also produces a resistant pressure, P_L, on the source. This ideal machine yields the following relationships:

$$P_s = TM_L$$
$$\omega = TQ$$

Part B

We need to find the modulus that results in a maximum speed. This can be done by operating at the peak power of the source curve. First off, let's convert these graphs into functions. For the source curve, on inspection, you'll find that the relationship is linear:

$$P_s(Q) = 18 - 0.06Q \ MPa$$

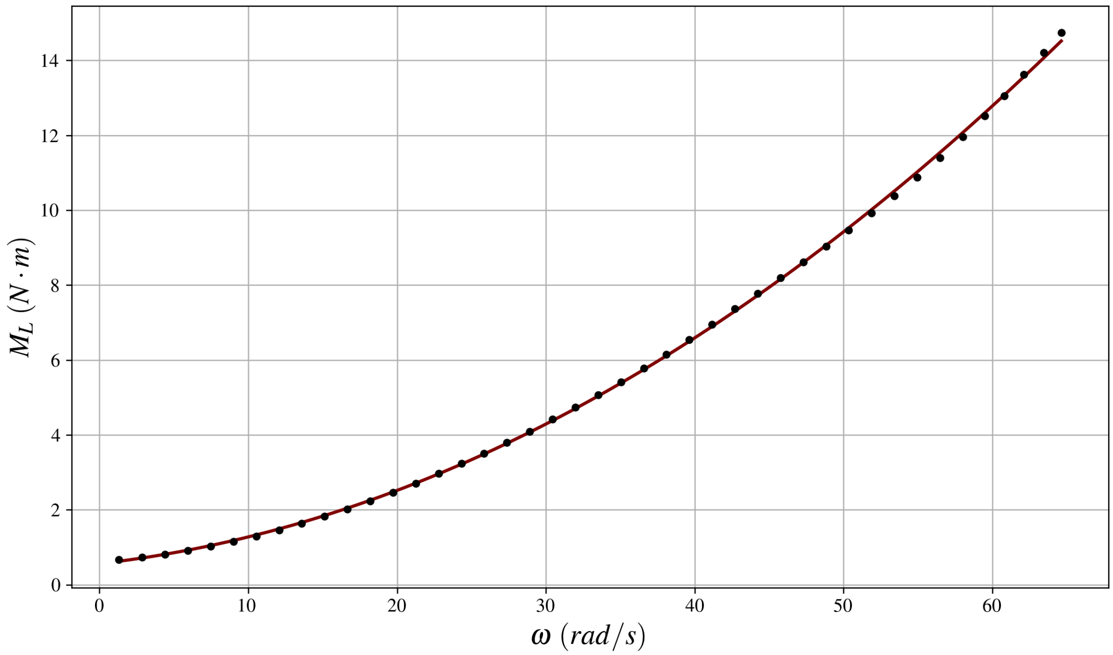
We need to ensure consistent units for the transformer modulus. This requires the pressure and flow rate to change to Pa and m^3/s, respectively. Making this change, we get:

$$P_s(Q) = 18 \cdot 10^6 - 6 \cdot 10^{10} Q \ Pa$$

This ensures that the units for the modulus come out to be $\frac{1}{m^3}$. For the load curve, this is a little more complicated. The curve can be approximated by a quadratic function that has been fitted to the data.

[2]:
```
# Finding the best fit curve for the load data
data = pd.read_csv('p1_data.csv')
omegas = np.array(data['omega'])
torques = np.array(data['torque'])

A, B, C = np.polyfit(omegas, torques, 2)
print(f'A: {A}, B: {B}, C: {C}')

omega_array = np.linspace(omegas[0], omegas[-1], 300)
torque_array = A*omega_array**2 + B*omega_array + C

fig, ax = plt.subplots()
ax.scatter(omegas, torques, zorder=3, color='black', marker='.')
ax.plot(omega_array, torque_array)
ax.set_xlabel(r'$\omega$ $(rad/s)$')
ax.set_ylabel(r'$M_L$ $(N\cdot m)$')
plt.show()
```

```
A: 0.002646604774363996, B: 0.04479742996739802, C: 0.5613276032319137
```



This yields the following equation:

$$M_L(\omega) = 0.00265\omega^2 + 0.0448\omega + 0.561\ N \cdot m$$

Now we can find the peak power of the source curve where power is

$$\dot{E} = P_s \cdot Q$$

[3]:
```
# Finding the peak power
Q = sp.Symbol('Q')
Ps = int(18e6) - int(6e10)*Q
E_dot = Ps*Q
E_dot
```

[3]:

$$Q\,(18000000 - 60000000000Q)$$

[4]:
```
E_dot_diff = E_dot.diff()
Q_max = sp.solve(E_dot_diff, Q)[0]
Q_max.n()   # m^3/s
```

[4]:

0.00015

We have calculated that the ideal flow rate would be $0.00015\,m^3/s$, but this is not within the bounds of source operating conditions. Therefore, we will have to operate at the maximum possible flow rate, which is $0.00005\,m^3/s$, since this is the peak power of the source.

[5]:
```
Q_max_power = 0.00005  # m^3/s
# Q_max_power = Q_max  # m^3/s

# Find the value of T at this flow rate
omega, T = sp.symbols('omega T')
M_L = A*omega**2 + B*omega + C
omega_ = T*Q
M_L_ = M_L.subs(omega, omega_)
eq = sp.Eq(Ps, T*M_L_)
eq
```

[5]:
$$18000000-60000000000Q = T\left(0.00264660477436 4Q^2T^2 + 0.044797429967398QT + 0.561327603231914\right)$$

[6]:
```
eq = eq.subs(Q, Q_max_power)
eq.simplify()
```

[6]:
$$6.61651193590999 \cdot 10^{-12}T^3 + 2.2398714983699 \cdot 10^{-6}T^2 + 0.561327603231914T = 15000000.0$$

[7]:
```
sol = sp.solve(eq, T)
sol
```

[7]:
```
[1190301.78775134,
 -764414.663069025 - 1149032.59001054*I,
 -764414.663069025 + 1149032.59001054*I]
```

[8]:
```
# Choose only the real solution
T_best = [T_val for T_val in sol if sp.im(T_val) == 0][0]
T_best  # 1/m^3
```

[8]:
1190301.78775134

The transformer modulus of $T = 1,190,302\,\frac{1}{m^3}$ produces the maximum power output that the source can provide. The corresponding load speed is

[9]:
```
omega_max = T_best*Q_max_power
omega_max  # rad/s
```

[9]:
59.5150893875669

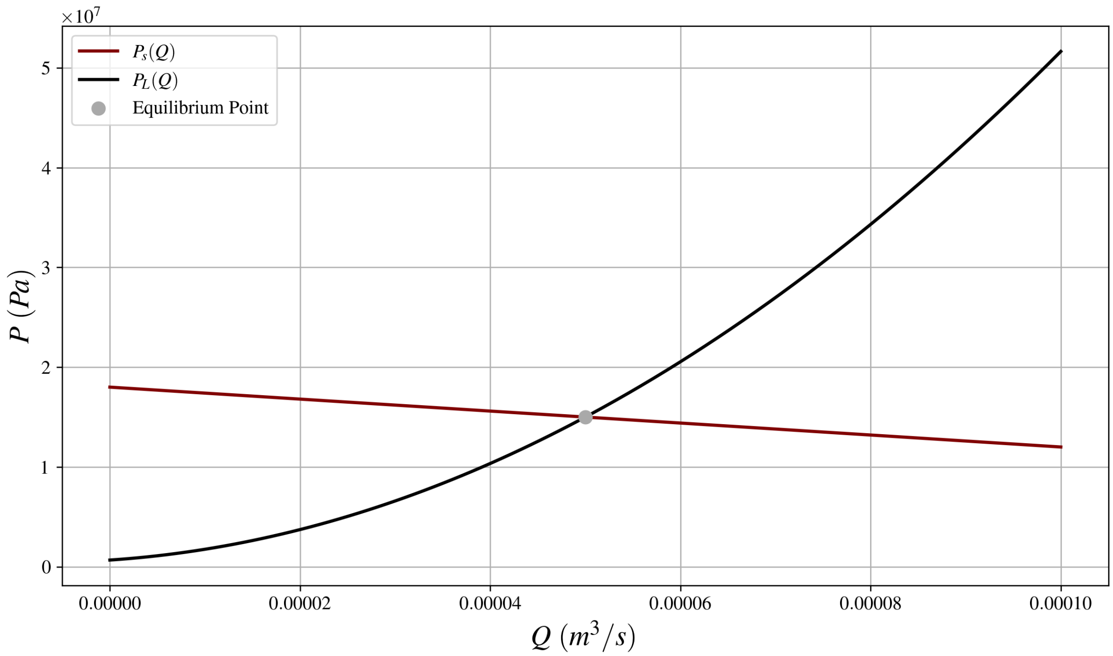
If we plot the source curve with the load curve, we should see an equilibrium point at $Q_{max} = 0.00005$. However, this load needs to be the load that acts on the source, P_L.

$$P_L = T \cdot M_L(\omega) = T \cdot M_L(T \cdot Q)$$

[10]:
```
P_L = T_best*M_L.subs(omega, T_best*Q)
P_L.simplify()
```

[10]:
$$4.46334388758242 \cdot 10^{15}Q^2 + 63469820628.0566Q + 668149.249641121$$

```
[11]: # Plotting
P_L_lamb = sp.lambdify(Q, P_L, modules='numpy')
Ps_lamb = sp.lambdify(Q, Ps, modules='numpy')

Q_array = np.linspace(0, float(Q_max_power)*2, 300)

fig, ax = plt.subplots()
ax.plot(Q_array, Ps_lamb(Q_array), label=r'$P_s(Q)$')
ax.plot(Q_array, P_L_lamb(Q_array), label=r'$P_L(Q)$')
ax.scatter(Q_max_power, Ps_lamb(Q_max_power), color='darkgrey', zorder=3,␣
 ↪label='Equilibrium Point')
ax.set_xlabel('$Q$ $(m^3/s)$')
ax.set_ylabel('$P$ $(Pa)$')
ax.legend()
plt.show()
```



Part C

The volumetric displacement per revolution would be

$$D = \frac{2\pi}{T}$$

which is just the reciprocal of the modulus. The 2π is the conversion factor from radians to revolutions.

[12]:
```python
D = 2*sp.pi/T_best
D.n(5)  # m^3/rev
```

[12]:

$5.2786 \cdot 10^{-6}$

Verification

We can verify that this modulus results in the maximum speed by considering the speed, ω, as a function of T.

[13]:
```python
eq1 = sp.Eq(Ps, T*M_L)
eq2 = sp.Eq(omega, T*Q)
display(eq1, eq2)
```

$$18000000 - 60000000000Q = T\left(0.002646604774364\omega^2 + 0.044797429967398\omega + 0.561327603231914\right)$$

$$\omega = QT$$

[14]:
```python
Q_of_T_omega = sp.solve(eq1, Q)[0]
eq3 = eq2.subs(Q, Q_of_T_omega)
eq3
```

[14]:

$$\omega = T\left(-4.41100795727333 \cdot 10^{-14}T\omega^2 - 7.46623832789967 \cdot 10^{-13}T\omega - 9.35546005386523 \cdot 10^{-12}T + 0.0003\right)$$

[15]:
```python
omega_of_T = sp.solve(eq3, omega, dict=True)
for sol in omega_of_T:
    for key, value in sol.items():
        display(sp.Eq(key, value).n(5))
```

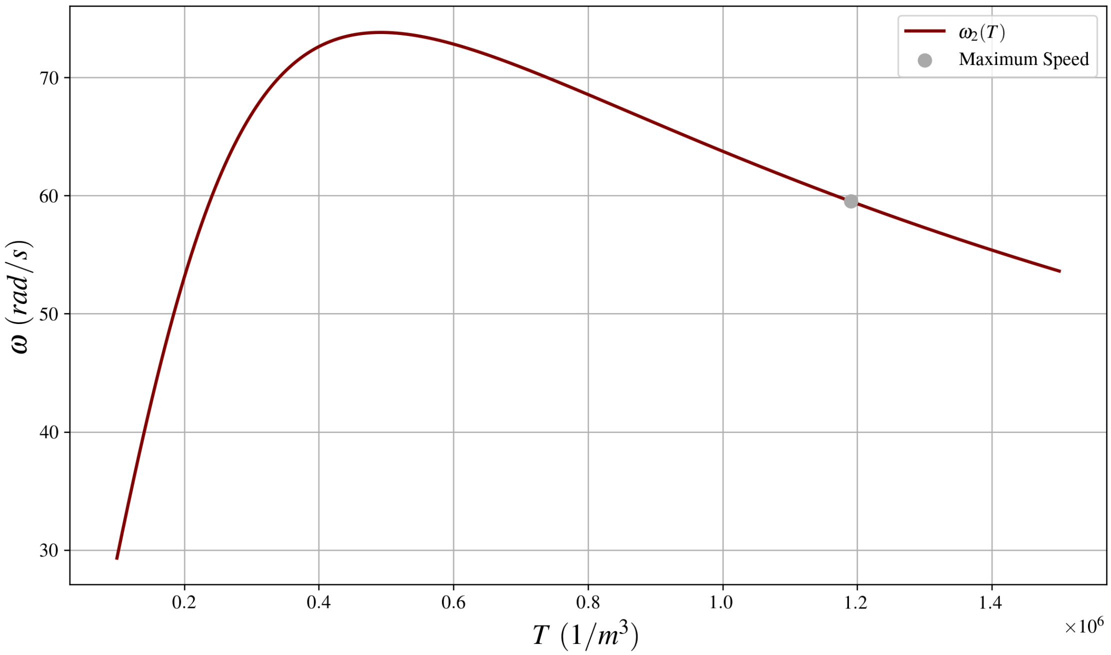
$$\omega = \frac{1.1335 \cdot 10^{-14}\left(-7.4662 \cdot 10^{14}T^2 - 1.0 \cdot 10^{27}\left(-1.0932 \cdot 10^{-24}T^4 + 5.2932 \cdot 10^{-17}T^3 + 1.4932 \cdot 10^{-12}T^2 + 1.0\right)^{0}\right.}{T^2}$$

$$\omega = \frac{1.1335 \cdot 10^{-14}\left(-7.4662 \cdot 10^{14}T^2 + 1.0 \cdot 10^{27}\left(-1.0932 \cdot 10^{-24}T^4 + 5.2932 \cdot 10^{-17}T^3 + 1.4932 \cdot 10^{-12}T^2 + 1.0\right)^{0}\right.}{T^2}$$

Only one of these solutions should result in positive values of ω for positive values of T, and it's the second solution.

[16]:
```python
omega1, omega2 = omega_of_T[0][omega], omega_of_T[1][omega]
omega1_lamb, omega2_lamb = sp.lambdify(T, omega1, modules='numpy'), sp.
 ↪lambdify(T, omega2, modules='numpy')

T_array = np.linspace(100_000, 1.5e6, 5000)
omega1_array, omega2_array = omega1_lamb(T_array), omega2_lamb(T_array)

fig, ax = plt.subplots()
# ax.plot(T_array, omega1_array, label=r'$\omega_1(T)$')
ax.plot(T_array, omega2_array, label=r'$\omega_2(T)$')
ax.set_xlabel(r'$T$ $(1/m^3)$')
ax.set_ylabel(r'$\omega$ $(rad/s)$')
```

```
ax.scatter(float(T_best), float(omega_max), color='darkgrey', zorder=3,␣
 ↪label='Maximum Speed')
ax.legend()
plt.show()
```

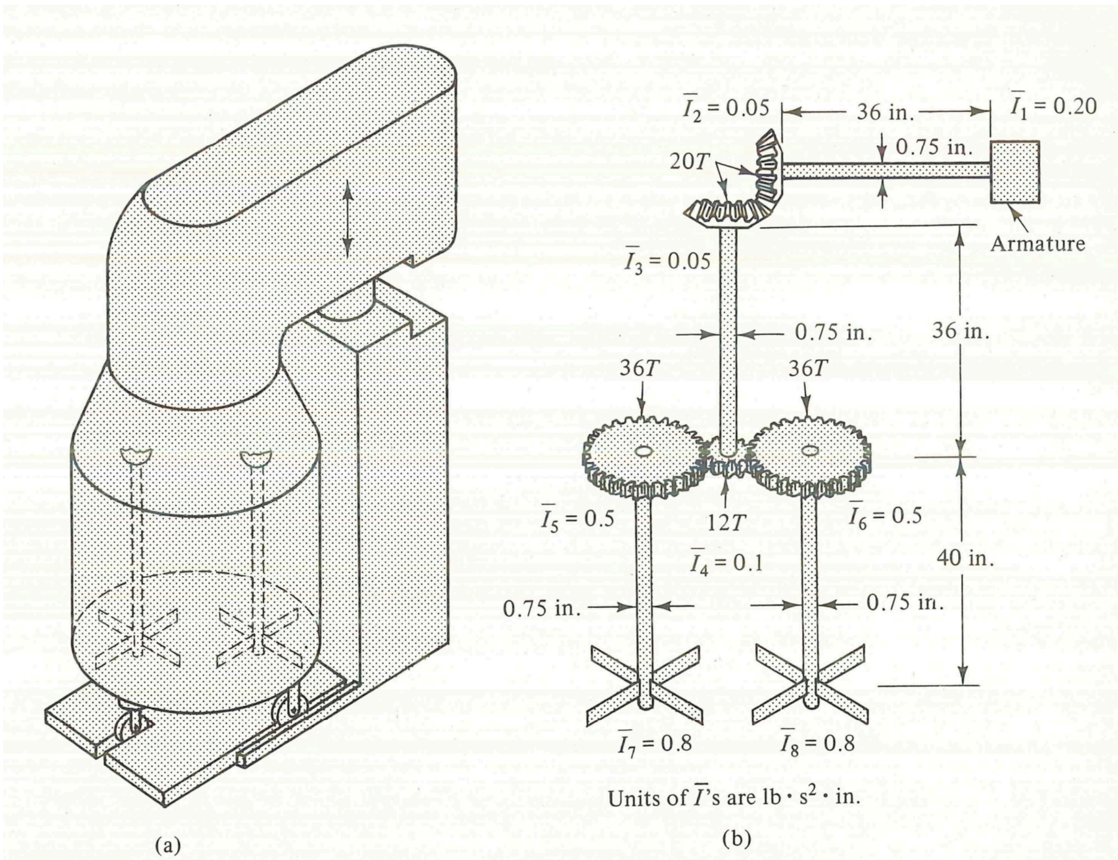


If there was not a limitation on the source, then the best transformer modulus would be $T = 492,104\,\frac{1}{m^3}$, resulting in a speed of $73.8\,rad/s$. However, this occurs at a flow rate of $Q_{max} = 0.00015$, which is not within the operating conditions of the source. If you wanted to see the results without this limitation, then you can uncomment the line `Q_max_power = Q_max  # m^3/s` in the code cell 5.

You should also recognize that in order to get that dot to move to the left, you need to keep increasing the flow rate Q. Looking at the graph, you can see that the output shaft speed decreases with increasing modulus, so we can definitely say that this is the maximum speed.

Problem 2

Given



For the figure shown above assume the armature has a moment-speed curve of $M_a(\omega_1) = -315\omega_1 + 2000$ when the motor is on and acts as a damper when the motor is off with $M_a(\omega_1) = -14.4\omega_1$. In both cases the armature speed is in rad/s and the moment is in $lbf \cdot in$.

Assume that all shafts are made of steel ($G = 1.15 \cdot 10^7\,psi$) and that the diameter of the bevel gears is 2 inches. Model each shaft as a rotational spring $\left(k = \frac{\pi r^4 G}{2L}\right)$. Model the resistance felt by each of the paddles (inertias 7 and 8 shown at the bottom of the industrial mixer) with rotational dampers of a coefficient of $B = 2880\,lbf \cdot in \cdot s$. The number of teeth of each gear (10T, 12T, and 36T) is given in the figure.

Find

Obtain the following:

a. A detailed free body diagram including your choice of reference coordinates.
b. A complete set of differential equations in state variable form. Notice that there are 8 inertias and 4 springs, but not all inertias are independent.
c. Solve the state variable equations using a 4th order Runge-Kutta method for a time period of 20 seconds. Assume that all initial states are zero and that the motor armature is on for

the first 10 seconds and off for the last 10 seconds. It is recommended that a very small time step is used. Try $\Delta t = 0.0005s$.

d. Verify using energy conservation and include a discussion on the results with the following points:
   - When the motor is turned on and at steady-state (ω_1 is constant), does it operate at its maximum power?
   - Also, consider the mechanical failure modes that might occur with the plotted speed results.