hw_2

October 15, 2025

1 AEROSP 536 Electric Propulsion: Homework 2

Jason Chen

1.1 Problem 1

1.1.1 Part (a)

We are given a table of different electric propulsion thruster types, along with their range of specific impulses, powers, and efficiencies. We know that for a given low thrust trajectory characterized by Δv and Δt, the performance metrics can be related by (as derived in lecture):

$$\Delta t = t_T = \frac{m_D (I_{sp} g_0)^2}{2\eta P_{in}} \left(e^{\Delta v / I_{sp} g_0} - 1 \right)$$

where m_D is the dry mass of the spacecraft and can be expressed as the sum of payload, structural, and power supply mass:

$$m_D = m_{PL} + m_S + m_{PS}$$

Finally note that power can be expressed here as a specific mass density α:

$$\alpha = \frac{m_{PS}}{P_{in}} = \text{ 25 kg/kW}$$

Let's assume that the structural mass is zero $m_S = 0$ because ideally the power system also acts as the spacecraft structure, and because there is no information given regarding it. This means we can rearrange the first equation:

$$\Delta t = \frac{(m_{PL} + \alpha P_{in})(I_{sp} g_0)^2}{2\eta P_{in}} \left(e^{\Delta v / I_{sp} g_0} - 1 \right)$$

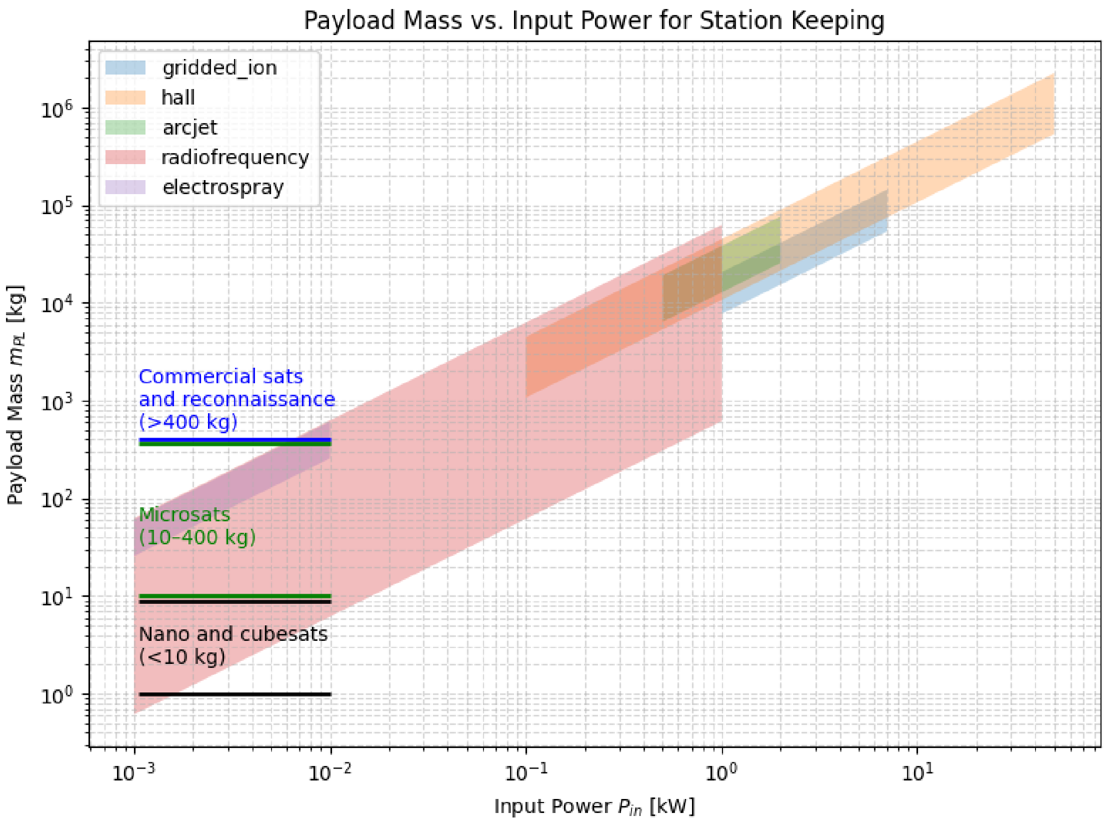
$$m_{PL} = \frac{2\eta P_{in} \Delta t}{\left(e^{\Delta v / I_{sp} g_0} - 1 \right) (I_{sp} g_0)^2} - \alpha P_{in}$$

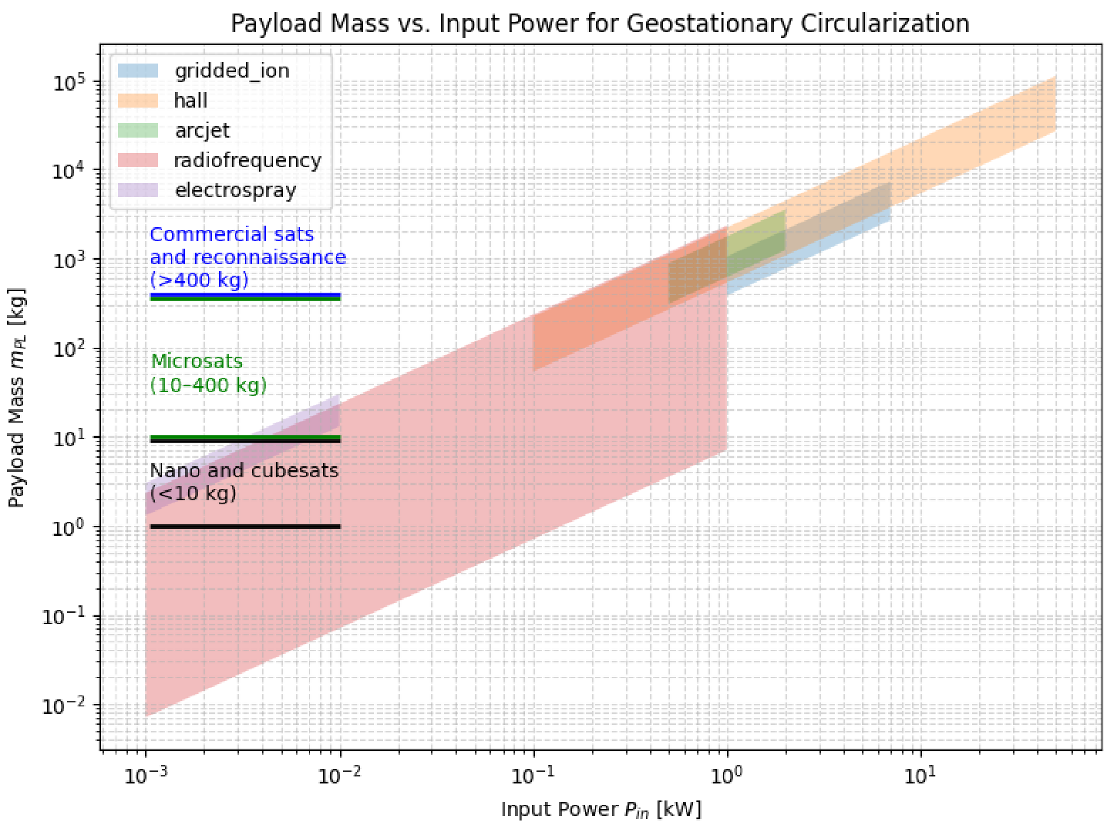
And we will plot this below.

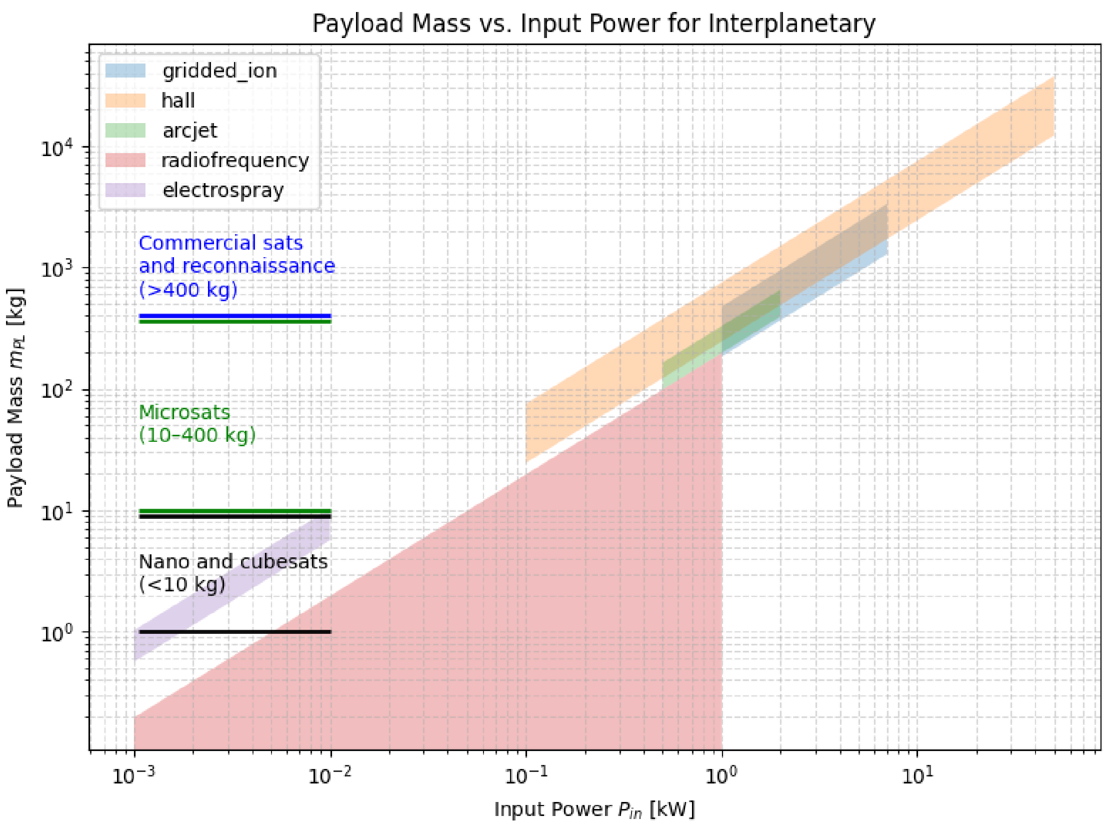
[1]:
```python
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from pint import UnitRegistry
ureg = UnitRegistry(auto_reduce_dimensions=True)
Q_ = ureg.Quantity
```

```python
missions = {
    'station_keeping': (Q_(100, 'm/s'), Q_(1, 'year')),
    'geostationary circularization': (Q_(1.4, 'km/s'), Q_(9, 'months')),
    'interplanetary': (Q_(10, 'km/s'), Q_(3, 'years'))
}
thrusters = {
    "gridded_ion":    {"isp": (2500, 5000), "P_in": (1.0, 7.0),    "eta": (0.
 ↪60, 0.80)},
    "hall":           {"isp": (1000, 3000), "P_in": (0.1, 50.0),   "eta": (0.
 ↪50, 0.70)},
    "arcjet":         {"isp": (500, 1000),  "P_in": (0.5, 2.0),    "eta": (0.
 ↪20, 0.30)},
    "radiofrequency": {"isp": (200, 1000),  "P_in": (0.001, 1.0),  "eta": (0.
 ↪01, 0.20)},
    "electrospray":   {"isp": (1000, 2000), "P_in": (0.001, 0.01), "eta": (0.
 ↪80, 0.95)},
}
alpha = Q_(25, 'kg/kW').magnitude
g0 = 9.81

for mission_name, (delta_v, delta_t) in missions.items():
    delta_v = delta_v.to('m/s').magnitude
    delta_t = delta_t.to('s').magnitude
    plt.figure(figsize=(8, 6))

    for thruster, params in thrusters.items():
        isp_min, isp_max = params['isp']
        P_in_min, P_in_max = params['P_in']
        eta_min, eta_max = params['eta']

        # Create meshgrid for P_in and Isp
        P_in_vals = np.logspace(np.log10(P_in_min), np.log10(P_in_max), 100)  #␣
 ↪kW
        isp_vals = np.array([isp_min, isp_max])
        eta_vals = np.array([eta_min, eta_max])

        # Calculate m_PL for all corners of the parameter space
        m_PL = np.zeros((2, 2, len(P_in_vals)))
        for i, isp in enumerate(isp_vals):
            for j, eta in enumerate(eta_vals):
                exp_term = np.exp(delta_v / (isp * g0)) - 1
                denom = exp_term * (isp * g0) ** 2
                m_PL[j, i, :] = ((2 * eta * P_in_vals * 1e3 * delta_t) / denom)␣
 ↪- (alpha * P_in_vals)
```

```
        # Find min/max payload mass for each P_in
        m_PL_min = np.min(m_PL.reshape(-1, len(P_in_vals)), axis=0)
        m_PL_max = np.max(m_PL.reshape(-1, len(P_in_vals)), axis=0)

        plt.fill_between(P_in_vals, m_PL_min, m_PL_max, alpha=0.3,␣
↪label=thruster)

    plt.xscale('log')
    plt.yscale('log')
    plt.xlabel('Input Power $P_{in}$ [kW]')
    plt.ylabel('Payload Mass $m_{PL}$ [kg]')
    plt.title(f'Payload Mass vs. Input Power for {mission_name.replace("_", "␣
↪").title()}')
    plt.hlines(1, P_in_vals[2], P_in_vals[-1], colors='black', linestyles='-',␣
↪linewidth=2)
    plt.hlines(9, P_in_vals[2], P_in_vals[-1], colors='black', linestyles='-',␣
↪linewidth=2)
    plt.hlines(10, P_in_vals[2], P_in_vals[-1], colors='green', linestyles='-',␣
↪linewidth=2)
    plt.hlines(360, P_in_vals[2], P_in_vals[-1], colors='green',␣
↪linestyles='-', linewidth=2)
    plt.hlines(400, P_in_vals[2], P_in_vals[-1], colors='blue', linestyles='-',␣
↪linewidth=2)
    plt.text(P_in_vals[2], 3, "Nano and cubesats\n(<10 kg)", va='center',␣
↪ha='left', fontsize=10, color='black')
    plt.text(P_in_vals[2], 50, "Microsats\n(10-400 kg)", va='center',␣
↪ha='left', fontsize=10, color='green')
    plt.text(P_in_vals[2], 1000, "Commercial sats\nand reconnaissance\n(>400␣
↪kg)", va='center', ha='left', fontsize=10, color='blue')
    plt.legend()
    plt.grid(True, which='both', ls='--', alpha=0.5)
    plt.tight_layout()
    plt.show()
```


Payload Mass vs. Input Power for Station Keeping
gridded_ion
hall
arcjet
radiofrequency
electrospray
Commercial sats
and reconnaissance
(>400 kg)
Microsats
(10–400 kg)
Nano and cubesats
(<10 kg)
Payload Mass m_{PL} [kg]
Input Power P_{in} [kW]
10^{0}
10^{1}
10^{2}
10^{3}
10^{4}
10^{5}
10^{6}
10^{-3}
10^{-2}
10^{-1}
10^{0}
10^{1}


Payload Mass vs. Input Power for Geostationary Circularization
gridded_ion
hall
arcjet
radiofrequency
electrospray
Commercial sats and reconnaissance (>400 kg)
Microsats (10–400 kg)
Nano and cubesats (<10 kg)
Payload Mass m_{PL} [kg]
Input Power P_{in} [kW]
10^{-2}
10^{-1}
10^{0}
10^{1}
10^{2}
10^{3}
10^{4}
10^{5}
10^{-3}
10^{-2}
10^{-1}
10^{0}
10^{1}



1.1.2 Part (b)

Hall thrusters are becoming the dominant form of electric propulsion because they have the highest ceiling of payload mass deliverable, and generally are extremely competitive when considering payload mass for a given input power above 1 kW (which is a reasonable size for a power supply on microsats and commercial satellites), only losing out narrowly to RF in stationkeeping. Additionally, as power supply technology inevitably improves and becomes lighter, hall thrusters are able to utilize that extra input power to deliver more payload mass. This is true for all domains of EP, as shown above in the previous 3 graphs.

1.1.3 Part (c)

For the GTO transfer, the launch mass (i.e. wet mass of the entire system) for the given problem statement can be found via the equation we already have for dry mass, and then using the rocket equation to find the wet mass required at the beginning of the mission.

[2]:
```
delta_v = missions['geostationary circularization'][0].m_as('m/s')
delta_t = missions['geostationary circularization'][1].m_as('s')
isp, eta, P_in = 2000, 0.7, 5
exp_term = np.exp(delta_v / (isp * g0)) - 1
denom = exp_term * (isp * g0) ** 2
```

```
m_dry = ((2 * eta * P_in * 1e3 * delta_t) / denom)
m_wet = m_dry * np.exp(delta_v / (isp * g0))
print(f"Launch mass for GTO transfer: {m_dry*1e-3:.2f} tonnes")
```

```
Launch mass for GTO transfer: 5.82 tonnes
```

1.1.4 Part (d)

For chemical systems, the amount of kinetic energy available within the chemical bonds of the propellant is related to total enthalpy, which is called specific energy here:

$$v_e = \sqrt{2\eta h_0}$$

where v_e is the effective exhaust velocity, and is related to the specific impulse through $v_e = I_{sp}g_0$. With this, we can plug back into the rocket equation (noting this is an impulsive maneuver and therefore do not need to integrate) and we get the dry mass of the vehicle. Since this is a chemical system, there is no significant power system contribution to total mass, and to keep the comparison consistent, we will not consider the structural mass of the vehicle. Therefore, the dry mass is simply the paylaod mass m_{PL}.

```
[3]: delta_v = 1.36e3
h0_hydrolox, eta_hydrolox = 20e6, 0.3
h0_hydrazine, eta_hydrazine = 2e6, 0.1
v_hydrolox = np.sqrt(2 * eta_hydrolox * h0_hydrolox)
v_hydrazine = np.sqrt(2 * eta_hydrazine * h0_hydrazine)
print(f"Payload mass for LH2/LOX: {m_wet * np.exp(-delta_v/v_hydrolox):.2f} kg")
print(f"Payload mass for Hydrazine: {m_wet * np.exp(-delta_v/v_hydrazine):.2f}␣
 ↪kg")
```

```
Payload mass for LH2/LOX: 4220.22 kg
Payload mass for Hydrazine: 727.70 kg
```

1.2 Problem 2

1.2.1 Part (a)

Specific final mass of the system is defined as:

$$\alpha_f = \frac{m_f}{P_{in}}$$

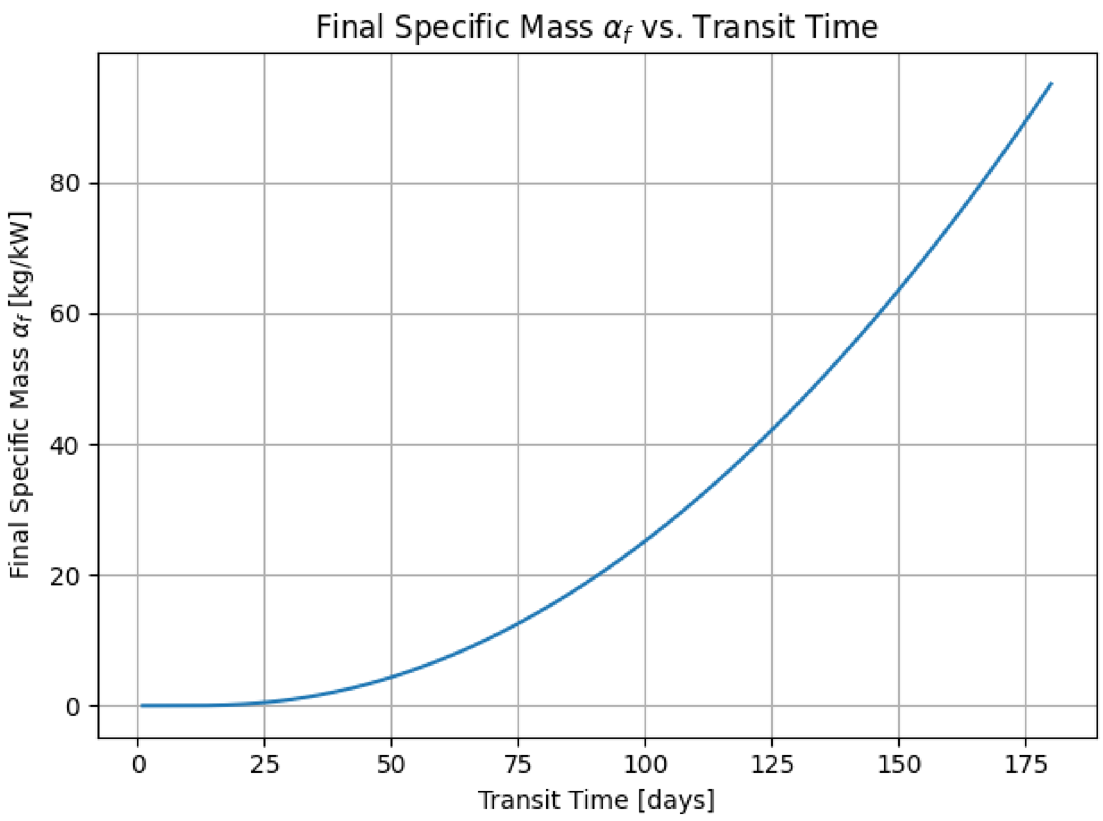
We are also given the relationship between transit time and Δv for a Mars trajectory. Thus, given a transit time, we will have a Δv requirement, and from this the only unknowns are now the input power and dry mass (since the problem statement gives us I_{sp} and η):

$$t_T = \frac{m_D (I_{sp}g_0)^2}{2\eta P_{in}} \left(e^{\Delta v / I_{sp}g_0} - 1\right)$$

Since we defined the "final mass" as the delivered mass, it is equivalent to the dry mass (includes power supply, payload, and structural mass). So we can rearrange the equation:

$$\alpha_f = \frac{2\eta t_T}{(I_{sp}g_0)^2 \left(e^{\Delta v / I_{sp}g_0} - 1\right)}$$

[13]:
```
transit_times = np.linspace(1, 180, 500)  # [days]
delta_v = (1250/transit_times + 90/(transit_times**2)) * 1e3  # [m/s]
isp, eta = 2000, 0.5
t_T = transit_times * 24 * 3600  # [s]
alpha_f = ((2*eta*t_T) / (((isp*g0)**2) * (np.exp(delta_v/(isp*g0)) - 1))) *␣
 ↪1e3  # [kg/kW]
plt.figure()
plt.plot(transit_times, alpha_f)
plt.xlabel('Transit Time [days]')
plt.ylabel(r'Final Specific Mass $\alpha_f$ [kg/kW]')
plt.title(r'Final Specific Mass $\alpha_f$ vs. Transit Time')
plt.grid(True)
plt.tight_layout()
plt.show()
```



1.2.2 Part (b)

Given that $\alpha_{PS} = 10$ kg/kW and assuming that the dry mass is 100% the power supply system $\alpha_{PS} = m_D/P_{in} = \alpha_f$, the fastest transit time that can be achieved can be found by:

$$t_T = \frac{\alpha_{PS}(I_{sp}g_0)^2}{2\eta}\left(e^{\Delta v/I_{sp}g_0} - 1\right)$$

Or, we can simply sample the data we've already found above for $\alpha_{PS} = 10$ kg/kW.

```
[ ]: print(f"Fastest transit time achievable for nuclear EP: {transit_times[np.
     ↪argmin(np.abs(alpha_f - 10))]:.1f} days")
```

```
Fastest transit time achievable for nuclear: 68.4 days
```

We can see that this is significantly shorter than the Hohmann transfer time of 260 days, but is idealistic in that it assumes essentially no other dry mass on the vehicle. Additionally, it is not leaving time for the required deceleration burn.

1.2.3 Part (c)

Assuming we have a nuclear EP mission with $t_T = 200$ days and $m_{PL} = 50,000$ kg, we can find the required power input as:

$$t_T = \frac{(m_{PL} + \alpha_{PS} P_{in})(I_{sp} g_0)^2}{2\eta P_{in}} \left(e^{\Delta v / I_{sp} g_0} - 1\right)$$

$$P_{in} = m_{PL} \left(\frac{t_T 2\eta}{(I_{sp} g0)^2 \left(e^{\Delta v / I_{sp} g_0} - 1\right)} - \alpha_{PS} \right)^{-1}$$

```
[19]: delta_v = (1250/200 + 90/(200**2)) * 1e3  # [m/s]
      t_T = 200 * 24 * 3600  # [s]
      p_in = 50000 * ((t_T * 2 * eta) / ((isp*g0)**2 * (np.exp(delta_v/(isp*g0))-1))␣
       ↪- 10/1e3) ** -1
      print(f"Required power input for 200 day transit with 50,000 kg payload: {p_in/
       ↪1e3:.1f} kW")
```

```
Required power input for 200 day transit with 50,000 kg payload: 456.2 kW
```

1.2.4 Part (d)

Assuming we can deliver 50,000 kg per launch, we need to find the wet mass of the LEO to Mars mission to determine the number of launches from Earth necessary.

We will assume the 200 day transit requirement and thus we can solve directly for the dry mass. Using the rocket equation again (which will not consider the gravity drag of the Sun, but is adequate for this estimate), we can get the wet mass of the transfer vehicle.

```
[22]: m_wet = (50000 + 10 * p_in/1e3) * np.exp(delta_v/(isp*g0))
      print(f"Wet mass for transfer vehicle: {m_wet:.2f} kg")
      print(f"Number of launches required: {int(np.ceil(m_wet/50000))}")
```

```
Wet mass for transfer vehicle: 75038.14 kg
Number of launches required: 2
```

1.2.5 Part (e)

We can repeat the process for chemical rockets, using the rocket equation.

```
[23]: delta_v, isp = 7e3, 400
      m_wet = 50000 * np.exp(delta_v/(isp*g0))
      print(f"Wet mass for transfer vehicle: {m_wet:.2f} kg")
      print(f"Number of launches required: {int(np.ceil(m_wet/50000))}")
```

```
Wet mass for transfer vehicle: 297649.61 kg
Number of launches required: 6
```

1.2.6 Part (f)

Comparing the architectures, there's a nontrivial trade depending on the assumptions made on current technology. Purely based of the number of launches and assumming all else equal, EP is a clear winner. Due to the long distance between Earth and Mars, the EP system can beat the chemical system in both mass fraction or transit time, depending on what is held constant. However, there is the element of complexity of designing and flying a half-megawatt nuclear reactor in space to be adequately reliable and safe. This engineering effort may or may not be more difficult than 4 more launches, depending on the reliability of the launch vehicle. There is also the element of in-space assembly as well.

Overall, based on mass, I would choose the EP system, but a clearer answer can be made given more information on the involved technologies.

1.3 Problem 6