7　機器配置・放熱面の決定

7.1　フェアリング重量の推算

フェアリングの直径を D とする. マージンとして 150mm を確保することにすると, 衛星の対角線の長さを考慮して,

$$D > \sqrt{2}3.2m + 0.15m = 4.675m \tag{1}$$

となり、ロケットのサイジングの配布資料よりフェアリング重量 W_F は

$$W_F = \frac{2}{4^{2.2}} D^{2.2} = 2.818t \tag{2}$$

となる.

7.2　必要 ΔV の計算

必要 ΔV は

$$\Delta V = \Delta V_{PO} + \Delta V_{PK} \tag{3}$$

$$\Delta V_{PO} = \text{地上からパーキング軌道までの}\ \Delta V = V_{CE} + \Delta V_H + \Delta V_g + \Delta V_A - \Delta V_E \tag{4}$$

7.2.1　V_{CE}

これは高度 0km における円軌道速度であり, 地球の半径 6371km を用いると

$$V_{CE} = \sqrt{\frac{\mu}{R}} = \sqrt{\frac{3.986 \times 10^{14}}{6371 \times 10^3}} = 7909.8m/s \tag{5}$$

となる.

7.2.2　ΔV_H

これは高度 0km の円軌道から半径 6600km の円軌道に至るホーマン移行の ΔV である. パーキング円軌道での速度 V_{PO} は

$$V_{PO} = \sqrt{\frac{\mu}{R_{PO}}} = \sqrt{\frac{3.986 \times 10^{14}}{6600 \times 10^3}} = 7771.4m/s \tag{6}$$

ホーマン遷移軌道のアポジ点とペリジ点における速度は

$$V_{H@APG} = \sqrt{\frac{2\mu R_E}{R_{PO}(R_E + R_{PO})}} = 7702.4m/s \tag{7}$$

$$V_{H@PRG} = \sqrt{\frac{2\mu R_{PO}}{R_E(R_E + R_{PO})}} = 7979.3m/s \tag{8}$$

であるから,

$$\Delta V_H = (V_{H@PRG} - V_{CE}) + (V_{PO} - V_{H@APG}) = 138.5m/s \tag{9}$$

7.2.3　ΔV_g と ΔV_A

これはグラビティ・ロスと空気抵抗による損失である.

$$\Delta V_g + \Delta V_A = 1680m/s \tag{10}$$

7.2.4 ΔV_E

これは地球自転による速度であり, 緯度 $30°$ として,

$$\Delta V_E = 400m/s \tag{11}$$

7.2.5 ΔV_{PK}

これは GTO 投入時のペリジキックの時の値であり,

$$V_{GTO@PRG} = \sqrt{\frac{2\mu R_{GEO}}{R_G EO(R_{PO} + R_{GEO})}} = 10219.4m/s \tag{12}$$

となるので

$$\Delta V_{PK} = V_{GTO@PRG} - V_{PO} = 2448.1m/s \tag{13}$$

7.3 ΔV の合計

$$\Delta V_{PO} = V_{CE} + \Delta V_H + \Delta V_g + \Delta V_A - \Delta V_E = 11776m/s \tag{14}$$

7.4 1・2 段ロケットの推進薬の決定

自分たちの班は, 液酸液水ロケットを設計した。配布プリントの液体水素, 液体酸素の I_sp の値を用いる.

1 段目では

$$I_{sp} = 430s \tag{15}$$

2 段目では

$$I_{sp} = 455s \tag{16}$$

を用いることにする.

7.5 燃料のトータル重量の最適化

配布プリントより

$$\Delta V = g(I_{sp1} log(\frac{1}{1-\xi_1}) + I_{sp2} log(\frac{1}{1-\xi_2})) \tag{17}$$

$$\xi_2 = \frac{0.975W_{P2}}{W_{PL} + W_{A2} + \eta_2 W_{P2} + W_{P2}} \tag{18}$$

$$\xi_1 = \frac{0.995W_{P1}}{W_{PL} + W_2 + W_F + W_{A1} + \eta_1 W_{P1} + W_{P1}} \tag{19}$$

η について配布資料の片対数グラフを直線近似して考える. $W_P = 100$ のとき,$\eta = 0.105$ であり,$W_P = 1000$ のとき,$\eta = 0.085$ であることより, 通る 2 点が決まったので直線の方程式は

$$\eta_i = -0.02 \log_{10} W_{Pi} + 0.145 \tag{20}$$

となる.

Chapter5 の結果と設定条件より

$$W_{PL} = W_{GTO} = 3203.1kg \tag{21}$$

$$W_{A1} = 200kg \tag{22}$$

$$W_{A2} = 400kg \tag{23}$$

である．また

$$W_1 = W_{A1} + \eta_1 W_{P1} + W_{P1} = 0.2 + (1.145 - 0.02log_{10}W_{P1})W_{P1}[t] \tag{24}$$

$$W_2 = W_{A2} + \eta_1 W_{P2} + W_{P2} = 0.4 + (1.145 - 0.02log_{10}W_{P2})W_{P2}[t] \tag{25}$$

フェアリング重量 W_F は以前の議論により

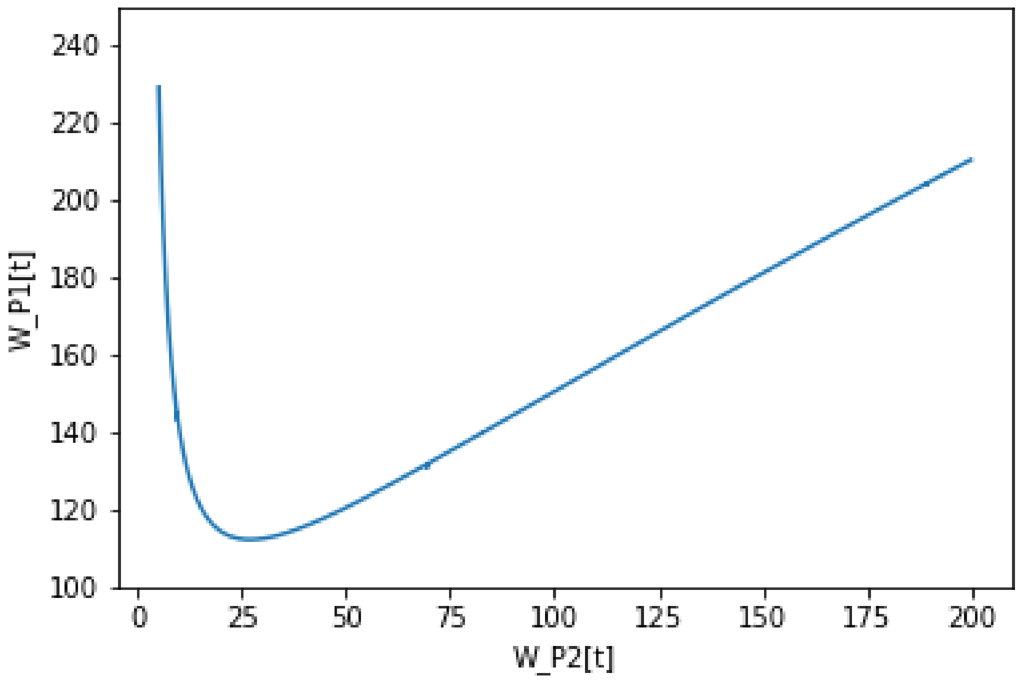
$$W_F = 2.818t \tag{26}$$

$$\xi_2 = \frac{0.975W_{P2}}{3.611 + (1.145 - 0.02log_{10}W_{P2})W_{P2}} \tag{27}$$

$$\xi_1 = \frac{0.995W_{P1}}{6.229 + (1.145 - 0.02log_{10}W_{P2})W_{P2} + (1.145 - 0.02log_{10}W_{P1})W_{P1}} \tag{28}$$

式 (8.17) に ξ_1 と ξ_2 を代入すると,W_{P1} と W_{P2} の関係式になる．これより 2 段目の燃料重量から 1 段の燃料重量を計算する．その結果とソースコードは以下の通り．横軸は W_{P2} で縦軸は W_{P1} である．

図 1　機器配置図 (水平方向)

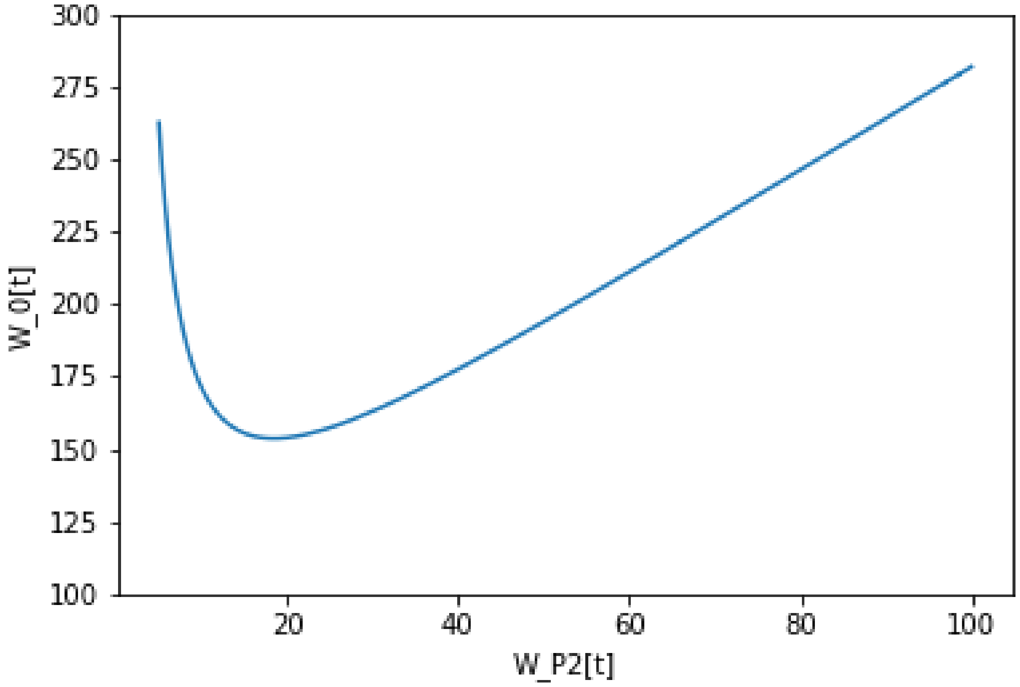


```python
import matplotlib.pyplot as plt
import math
import numpy as np

W_P2 = np.linspace(5, 200, 7000)
W_P1 = np.array([])


def cal_xi2(w_p2):
    return 0.975 * w_p2 / (3.611 + (1.145 - 0.02 * math.log10(w_p2)) * w_p2)


def cal_xi1(w_p1, w_p2):
    return 0.995 * w_p1 / (6.299 + (1.145 - 0.02 * math.log10(w_p2)) *
    w_p2 + (1.145 - 0.02 * math.log10(w_p1)) * w_p1)


def delta(w_p1, w_p2):
    return 11776 - 9.8 * (430 * math.log(1 / (1 - cal_xi1(w_p1, w_p2))) +
                          455 * math.log(1 / (1 - cal_xi2(w_p2))))


a_list = [10, 1000]

for j in W_P2:
    while(delta(a_list[0], j) >= 0 and delta(a_list[1], j) <= 0):
        if(abs(delta(a_list[1], j)) > 0.001):
            if(delta((a_list[0] + a_list[1]) / 2, j) > 0):
                a_list[0] = (a_list[0] + a_list[1]) / 2
            if(delta((a_list[0] + a_list[1]) / 2, j) < 0):
                a_list[1] = (a_list[0] + a_list[1]) / 2

        else:
            print(delta((a_list[0] + a_list[1]) / 2, j))
            print(a_list)
            W_P1 = np.append(W_P1, [(a_list[0] + a_list[1]) / 2])
            a_list = [10, 1000]
            break

plt.plot(W_P2, W_P1)
plt.xlabel("W_P2[t]")
plt.ylabel("W_P1[t]")
plt.show()
```

また打ち上げ総重量

$$W_0 = W_{PL} + W_F + W_1 + W_2 = 5.327 + (1.145 - 0.02log_{10}W_{P2})W_{P2} + (1.145 - 0.02log_{10})W_{P1} \tag{29}$$

であり, 直前の議論で W_{P1} と W_{P2} のペアの値が計算できるのでそれを用いて W_0 の最適化を行うと結果とソースコードは以下の通り. 横軸は W_{P2} で縦軸は W_{P0} である.


300
275
250
225
200
175
150
125
100
W_0[t]
20
40
60
80
100
W_P2[t]

```
import matplotlib.pyplot as plt
import math
import numpy as np

W_P2 = np.linspace(5, 100, 7000)
W_P1 = np.array([])
W_0 = np.array([])


def cal_xi2(w_p2):
    return 0.975 * w_p2 / (3.611 + (1.145 - 0.02 * math.log10(w_p2)) * w_p2)


def cal_xi1(w_p1, w_p2):
    return 0.995 * w_p1 / (6.299 + (1.145 - 0.02 * math.log10(w_p2)) *
     w_p2 + (1.145 - 0.02 * math.log10(w_p1)) * w_p1)


def delta(w_p1, w_p2):
    return 11776 - 9.8 * (430 * math.log(1 / (1 - cal_xi1(w_p1, w_p2))) +
                          455 * math.log(1 / (1 - cal_xi2(w_p2))))


a_list = [10, 1000]

for j in W_P2:
    while(delta(a_list[0], j) >= 0 and delta(a_list[1], j) <= 0):
        if(abs(delta(a_list[1], j)) > 0.001):
            if(delta((a_list[0] + a_list[1]) / 2, j) > 0):
                a_list[0] = (a_list[0] + a_list[1]) / 2
            if(delta((a_list[0] + a_list[1]) / 2, j) < 0):
                a_list[1] = (a_list[0] + a_list[1]) / 2

        else:
            W_P1 = np.append(W_P1, [(a_list[0] + a_list[1]) / 2])
            W_0 = np.append(W_0, [5.327 + (1.145 - 0.02 * math.log10(j))
             * j + (1.145 - 0.02 * math.log10((a_list[0] + a_list[1]) / 2))
              * (a_list[0] + a_list[1]) / 2])

            a_list = [10, 1000]
            break

print(np.min(W_0))
num = np.argmin(W_0)
print(W_P1[num])
print(W_P2[num])
plt.plot(W_P2, W_0)
plt.xlabel("W_P2[t]")
plt.ylabel("W_0[t]")
plt.show()
```

$$W_{0min} = 153.72t \tag{30}$$

この時,

$$W_{P1} = 115.87t \tag{31}$$

$$W_{P2} = 18.315t \tag{32}$$

最後にこの時のペイロード比は

$$\Lambda = \frac{3.203}{153.72} = 0.0208 \tag{33}$$

となる．これは少ない気がするが、考えられる原因としては