Project 1

February 20, 2022

Gabe Morris

[1]:
```
# Imports
import warnings

import numpy as np
import pandas as pd
from IPython.display import display, Latex
from msu_esd import Pipe, hardy_cross
from scipy.optimize import fsolve

warnings.filterwarnings('ignore', category=FutureWarning)  # Some random␣
 ↪warning with pandas
```

Contents

1 Introduction

A central chiller system for the Orlando International Airport is to be investigated. The airport consists of 4 concourses (Airside 1, 2, 3, and 4) shown in Figure 1. The main terminal building is not included in the chiller system. The working fluid of this system is Therminol D-12TM, a common heat transfer fluid.

Figure 1: Concourse Layout

In addition, the following specifications must be met:

- Provide valves to isolate each concourse, pump, and supply and return lines.
- Minimize the pipe length by placing lines in the same tunnel when feasible.
- Provide 50 feet of pipe in each mechanical room (one per concourse).
- Use a "Z" network (shown in Figure 3).
- Avoid pipe velocities much in excess of 9 feet per second.
- Access tunnels are 20 feet below the entrance to each concourse, mechanical rooms are 20 feet above the surface.
- The 2 air handling units in concourse 4 are to be placed in parallel.
- Avoid running lines under the main terminal building.
- Flow rates across the air handling units must not exceed 3.5% of the minimum required.

2 Given

The energy requirements across the air handling units are,

Table 1: AHU Energy Requirements

Concourse	Tons	K_{AHU}
1	480	4.5
2	480	4.5
3	480	4.5
4 (each)	225	10

where 1 ton is $12,000\ \frac{Btu}{hr}$. The head loss through the air handling units in each terminal is $K_{AHU}Q^2$ where Q is the flow rate in cubic feet per second. The units for K_{AHU} are $\frac{ft\,lbf}{lbm}$. The head loss across the chiller is taken to be $0.1Q^2$. The Therminol D-12 exits the chiller $25^\circ F$ cooler than it enters. The lines are well insulated.

The fluid properties are to be taken at $80^\circ F$ due to the relatively warm year-round temperatures in Orlando. The following properties were found:

$\rho = 47.3\frac{lbm}{ft^3}$
$c_p = 0.506\frac{Btu}{lbm^\circ F}$
$\mu = 2.72\frac{lbm}{ft\,hr}$

The piping material used for this system will be galvanized steel, a common piping material for chiller systems. The absolute roughness of galvanized steel is 0.0005'.

3 Solution

The first step is to obtain a physical mapping of the airport. A schematic of the airport was given to determine a rough idea of the lengths for each line. The location of the mechanical rooms were assumed to be somewhere in the center of each airside (shown in Figure 2).

3.1 Physical Mapping

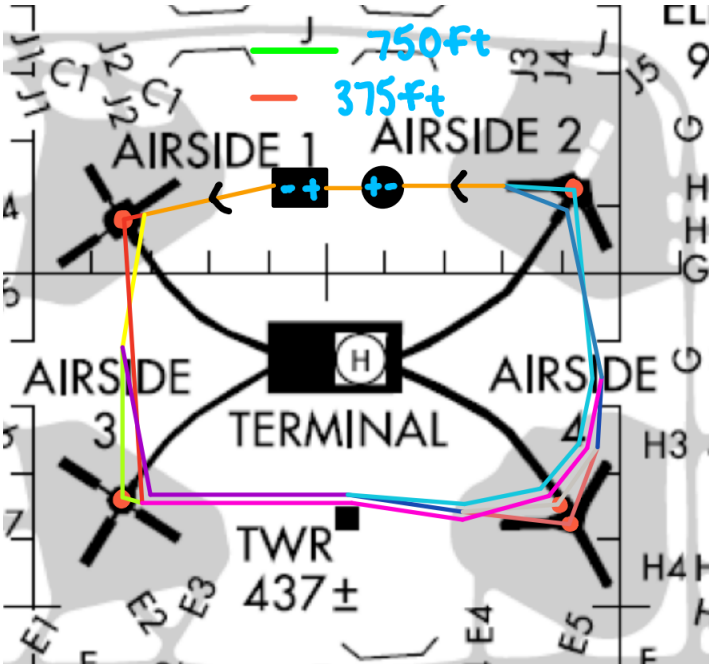

Figure 2: Physical Map of the Chiller System

The lateral distance between pipes shown above is slightly exaggerated to make it clear of the individual pipes. It is especially exaggerated around airside 4 (the parallel arrangement).

3.1.1 Pipe Lengths

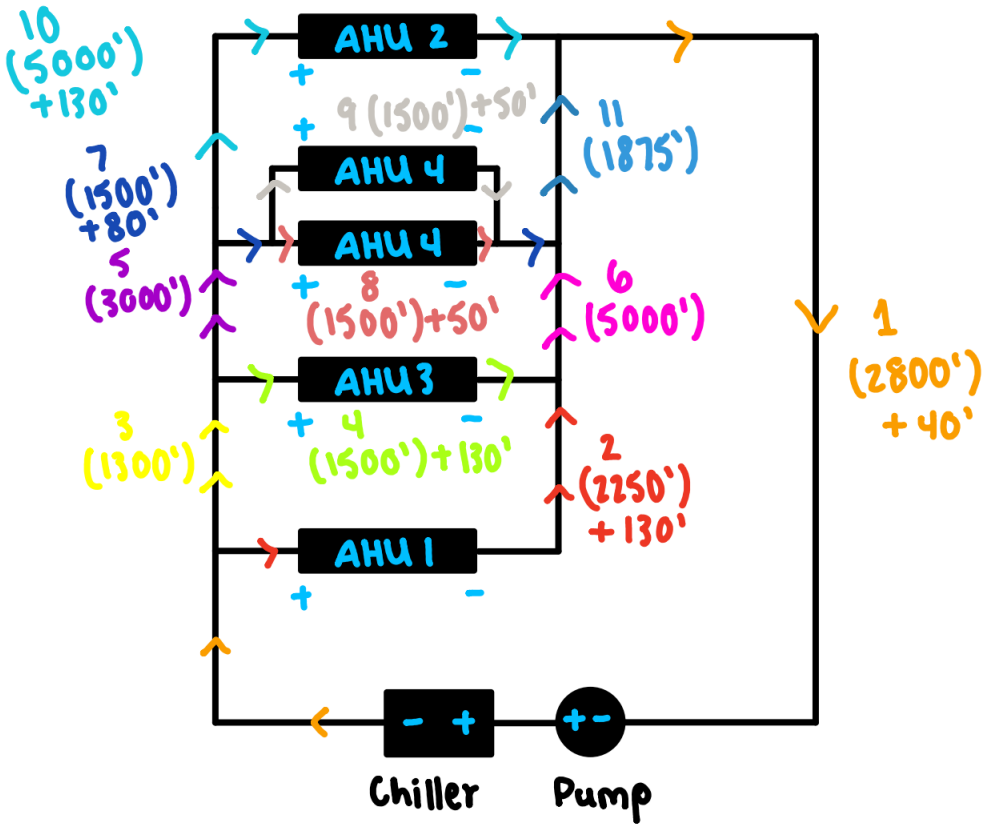

Figure 3: Z-Network Schematic of Lengths

Figure 3 shows the lengths associated with each line. The original length is shown in parentheses, and the added lengths are shown if there is a change in elevation or if the line is in a mechanical room (additional 50 feet). Here is a summary of the lengths for each of the 11 lines:

- Pipe 1: Original length is 2800'. Assumed to be at ground level, so add the piping that goes 20' up and 20' down to the tunnels. Total length is **2840'**
- Pipe 2: Original length is 2250'. This line runs through a mechanical room that is 40' up from the tunnels, then an additional 50' in the mechanical room, then 40' back down to the tunnels. The total length is **2380'**
- Pipe 3: There is no elevation change. The total length is **1300'**
- Pipe 4: The original length is 1500'. This line runs through a mechanical room, so an additional 130' is added to account for the elevation change and additional length within the room. The total length is **1630'**.
- Pipe 5: There is no elevation change. The total length is **3000'**.
- Pipe 6: There is no elevation change. The total length is **5000'**.
- Pipe 7: The original length on the map is 1500'. The line runs up to the mechanical room, but does not run through the mechanical room. Pipe 7 splits up prior to the parallel arrangement in airside 4, then meets up with lines 8 and 9 and returns to the tunnel. Although it is two different pipes, the flow rate is the same, so it will be analysed as if it were one pipe. The total length is **1580'**.
- Pipe 8: The original length is 1500'. It runs through the mechanical room, adding 50'. The

total length is **1550'**.

- Pipe 9: The original length is 1500'. It runs through the mechanical room, adding 50'. The total length is **1550'**.
- Pipe 10: The original length is 5000'. It runs through a mechanical room and changes elevation when doing so. The total length is **5130'**.
- Pipe 11: There is no elevation change. The total length is **1875'**.

3.1.2 Pipe Fittings and Valve Requirements

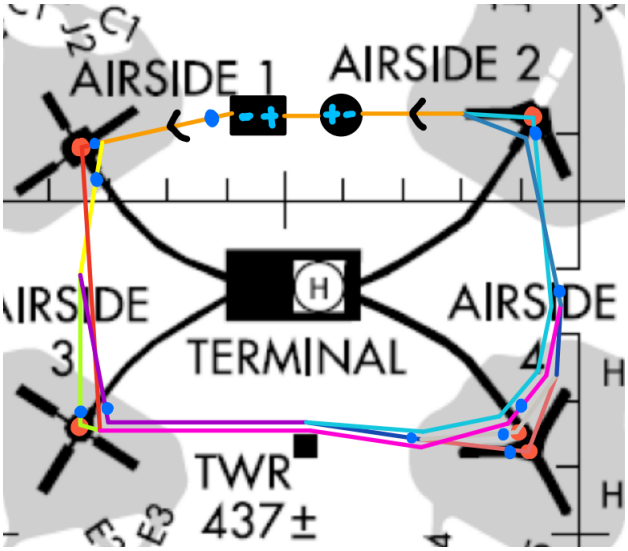

Figure 4: Physical Map of Valve Placements

Figure 4 shows the exaggerated valve placements (the blue dots). The specifications require that each line receives at least one valve.

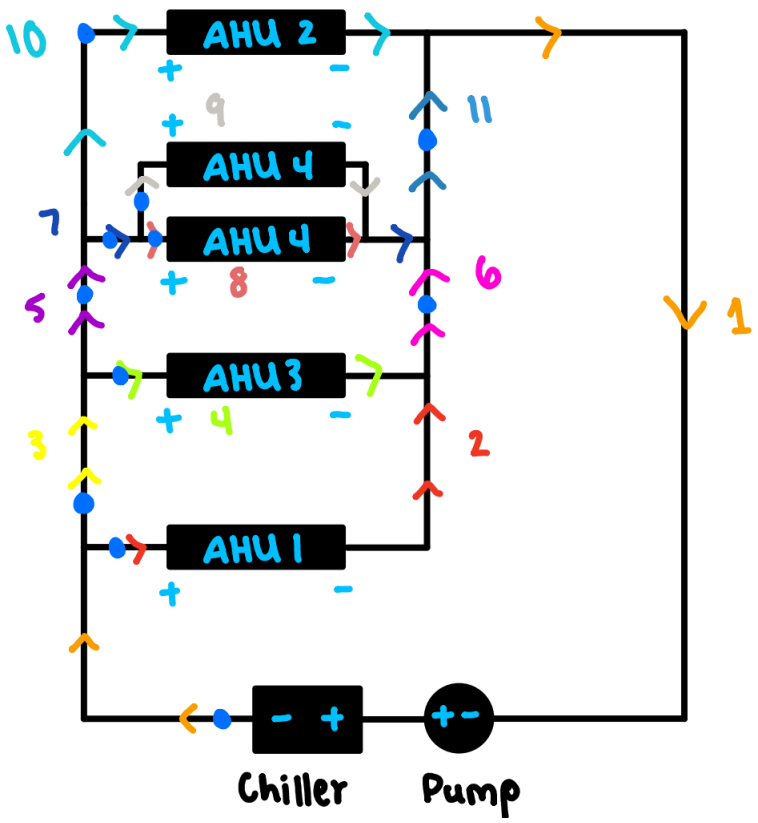

Figure 5: Schematic of Valve Placements

Figure 5 shows a clearer network of the pipe placements. The shutoff valve for line 1 is positioned after the chiller to prevent bucking of the pump. The valves will be gate valves, since it is the most common type of valve. The loss per each gate valve is $K = 8f_T$ (each line will have $C = 8$ because this is the only loss considered that gets multiplied by f_T). The fittings used for this network will be standard tee (for splitting the flow) and elbow fittings (for changing direction). There is a lot of approximation when it comes to the fittings, but the major losses and losses across the air handling units for this case should triumph over the loss produced by the fittings, since the lengths of each pipe are thousands of feet. All pipes will have a loss due to the entrance and exit ($K = 1.78$). Here is a description for the minor losses in each pipe (each fitting is visualized in Figure 4):

- Pipe 1 is attached to two flow through run tee connectors with a loss of 0.4 (according to Table 1-1 in the text). It is possible with the mapping shown in Figure 2 and 4 for there to be no elbows. $K_1 = 1.78 + 2(0.4) = 2.58$
- Pipe 2 is the flow through run for one tee connector and the branch flow for the other tee connector. There is one elbow to change direction toward airside 3. $K_2 = 1.78+0.4+1+0.75 = 3.93$
- Pipe 3 is the flow through run of one tee connection and the branch flow of another tee connection. $K_3 = 1.78 + 0.4 + 1 = 3.18$
- Pipe 4 is the flow through run of two tee connections and contains an elbow. $K_4 = 1.78 + 0.75 + 0.4(2) = 3.33$
- Pipe 5 is the branch flow of one tee connection and flow through for another tee connection. There are 2 elbows along its line. $K_5 = 1.78 + 0.4 + 1 + 2(0.75) = 4.68$
- Pipe 6 is the flow through run of one tee connection and the flow through branch on another.

It has 2 elbows. $K_6 = 1.78 + 2(0.75) + 0.4 + 1 = 4.68$

- Pipe 7 is the flow through run of four tee connections (remember it splits). $K_7 = 1.78 + 4(0.4) = 3.38$
- Pipe 8 is the flow through run of two tee connections and contains an elbow. $K_8 = 1.78 + 0.4(2) + 0.75 = 3.33$
- Pipe 9 is the flow through branch of two tee connections and contains two elbows (seen more clearly in Figure 5). $K_9 = 1.78 + 1(2) + 2(0.75) = 5.28$
- Pipe 10 is the flow through branch two connections and has 4 elbows. $K_{10} = 1.78 + 2(1) + 4(0.75) = 6.78$
- Pipe 11 is the flow through branch of two connections and has an elbow. $K_{11} = 1.78 + 2(0.4) + 0.75 = 3.33$

3.2 Boundary Conditions

The minimum flow rate across each air handling unit may be found using this relationship,

$$q = \dot{m} c_p \Delta T$$
$$\dot{m} = \rho Q$$

where q is the energy in tons given in Table 1. Keeping an eye on the units, Q may be solved.

$$Q_{min} = \frac{q}{c_p \Delta T \rho} \rightarrow Q_{min} = \frac{10q}{3 c_p \Delta T \rho} \frac{ft^3}{s}$$

The above expression is true for the units of c_p and ρ defined in the Given section and with q in tons. The upper bound of the flow rate is going to be $3.5\% \cdot Q_{min} + Q_{min}$.

[2]:
```
# Finding the upper and lower bounds
q = np.array([480, 480, 225, 225, 480])  # In numerical order according to␣
 ↪Figure 5
c_p, rho, del_T = 0.506, 47.3, 25  # Units are described earlier

Q_min = (10*q)/(3*c_p*del_T*rho)
Q_min
```

[2]:
```
array([2.67404257, 2.67404257, 1.25345745, 1.25345745, 2.67404257])
```

[3]:
```
Q_max = Q_min + Q_min*0.035
Q_max
```

[3]:
```
array([2.76763406, 2.76763406, 1.29732846, 1.29732846, 2.76763406])
```

The boundary conditions are,

$2.67 \leq Q_2 < 2.77$
$2.67 \leq Q_4 < 2.77$
$1.25 \leq Q_8 < 1.30$
$1.25 \leq Q_9 < 1.30$
$2.67 \leq Q_{10} < 2.77$

where Q is in $\frac{ft^3}{s}$. The only other boundary condition is that the velocity of the fluid in all the pipes has to be less than $9 \frac{ft}{s}$.

3.3 Diameter Selections

The minimum diameter for the pipes containing the air handling units may be found using,

$$Q = VA = V\frac{\pi}{4}D^4$$
$$D^4 = \frac{4Q}{V\pi}$$
$$D_{min} = \sqrt[4]{\frac{4Q_{min}}{V_{max}\pi}}$$

where Q_{min} was solved above, and the maximum velocity is $9\frac{ft}{s}$.

[4]:
```
# Calculating the minimum diameters
D_min = (4*Q_min/(9*np.pi))**0.25
D_min  # In ft
```

[4]:
```
array([0.78425804, 0.78425804, 0.64892469, 0.64892469, 0.78425804])
```

If the minimum diameters are that large, it is acceptable to choose a lower schedule piping. Schedule 5 piping will be considered for this analysis, since it encompasses the larger diameters. A chart for schedule 5 piping can be found here.

[5]:
```
# Getting a table of values to choose from
nps = np.array([8, 10, 12, 14, 16, 18, 20])  # Nominal Pipe Size
od = np.array([8.625, 10.75, 12.75, 14, 16, 18, 20])  # Outer Diameter
t = np.array([0.109, 0.134, 0.156, 0.156, 0.165, 0.165, 0.188])  # Thickness
id_ = od - 2*t  # Inner Diameter
id_ft = id_/12

df = pd.DataFrame({'NPS': nps, 'Outer Diameter (in)': od, 'Thickness (in)': t,␣
 ↪'Inner Diameter (in)': id_, 'Inner Diameter (ft)': id_ft})
# df
display(Latex(df.to_latex(index=False)))
```

NPS	Outer Diameter (in)	Thickness (in)	Inner Diameter (in)	Inner Diameter (ft)
8	8.625	0.109	8.407	0.700583
10	10.750	0.134	10.482	0.873500
12	12.750	0.156	12.438	1.036500
14	14.000	0.156	13.688	1.140667
16	16.000	0.165	15.670	1.305833
18	18.000	0.165	17.670	1.472500
20	20.000	0.188	19.624	1.635333

There are four diameters that need to be considered: the main line, the supply lines, the lines across air handling units 1, 2, and 3, and the lines across air handling units in airside 4. The main pipeline (line 1) needs to be the largest diameter because it will have the largest flow rate. Increasing the diameter will help to decrease the velocity. The lines with the air handling units need to be smaller, and air handling units in airside 4 can have smaller diameters than the lines that run through the other units because its flow rate doesn't have to be as high (as seen in the Boundary Conditions section).

The above analysis is necessary for getting a range of diameters, but which one should be chosen for each line? There are $7^4 = 2401$ possible arrangements to choose. In the appendix, there is a considerable amount of code that loops through all the possible diameter arrangements for each pump head value (pump head values range from 50 feet to 210 feet at 10 feet increments). This brings a total of $17 * 2401 = 40,817$ outcomes that were tested. Each iteration tested to see if,

1. The velocity in all the pipes were less than $9\frac{ft}{s}$
2. $2.67 \leq Q_2 < 2.77$
3. $2.67 \leq Q_4 < 2.77$
4. $1.25 \leq Q_8 < 1.30$
5. $1.25 \leq Q_9 < 1.30$
6. $2.67 \leq Q_{10} < 2.77$

This test resulted in 1 outcome having passed 5 of the 6 tests. This iteration occurred with a pump head of $150\,ft$ with the main line being 20 NPS, the supply lines being 12 NPS, the lines across AHU 1, 2, and 3 being 10 NPS, and the lines across AHU 4 being 8 NPS. Table 2 is a summary for all the decisions made thus far.

Table 2: Summary of Piping

Pipe	L (ft)	D (in)	# Valves	# Tees	# Elbows	K	C
1	2840	19.624	1	2	0	2.58	8
2	2380	10.482	1	1	1	3.93	8
3	1300	12.438	1	1	0	3.18	8
4	1630	10.482	1	0	1	3.33	8
5	3000	12.438	1	1	2	4.68	8
6	5000	12.438	1	1	2	4.68	8
7	1580	12.438	1	2	0	3.38	8
8	1550	8.407	1	0	1	3.33	8
9	1550	8.407	1	0	2	5.28	8
10	5130	10.482	1	0	4	6.78	8
11	1875	12.438	1	0	1	3.33	8

Material: Galvanized Steel (Schedule 5)
Absolute Roughness: $\epsilon = 0.0005\,ft$
Working Fluid: Therminol D-12
Density: $\rho = 47.3\frac{lbm}{ft^3}$
Heat Capacity: $c_p = 0.506\frac{Btu}{lbm^\circ F}$
Viscosity: $\mu = 2.72\frac{lbm}{ft\,hr}$

3.4 Kirchhoff Setup

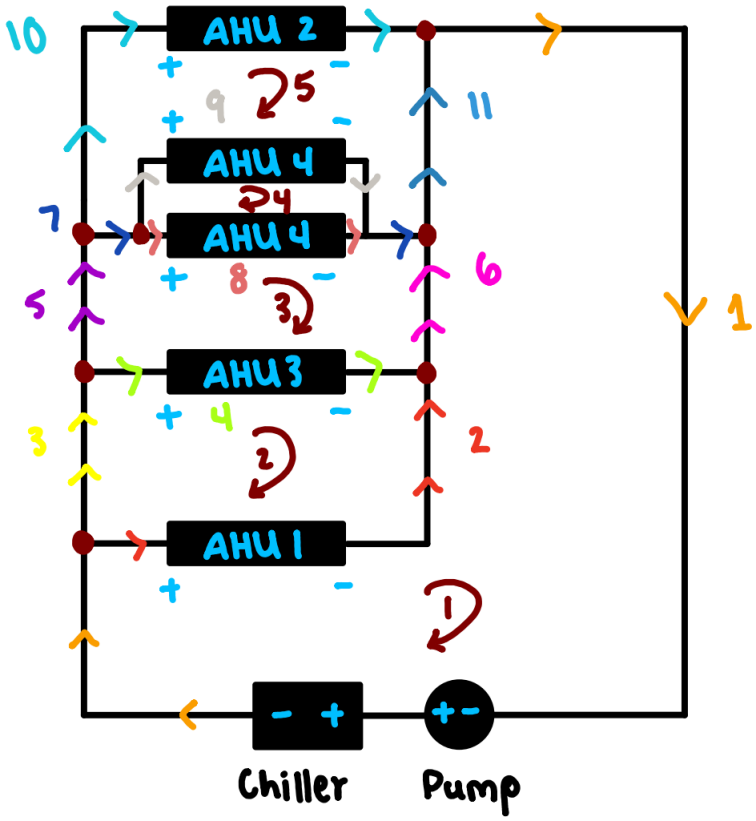

Figure 6: Kirchhoff Network

The relationship for the losses across the air handling units and the chiller result in units of feet upon converting *lbm* to *slugs* and dividing by g. This means that the rest of the terms in the system also need to be expressed in feet (make sure everything is divided by g for the loop equations). From Figure 6, the following system may be formed:

$$\begin{cases} Q_1 = Q_3 + Q_2 \\ Q_3 = Q_4 + Q_5 \\ Q_6 = Q_2 + Q_4 \\ Q_5 = Q_7 + Q_{10} \\ Q_7 = Q_8 + Q_9 \\ Q_{11} = Q_6 + Q_7 \\ h_2 + K_1 Q_2 |Q_2| + h_6 + h_{11} + h_1 - 150 + 0.1 Q_1 |Q_1| = 0 \\ h_4 + K_3 Q_4 |Q_4| - h_2 - K_1 Q_2 |Q_2| + h_3 = 0 \\ h_7 + h_8 + K_4 Q_8 |Q_8| - h_6 - h_4 - K_3 Q_4 |Q_4| + h_5 = 0 \\ h_9 + K_4 Q_9 |Q_9| - h_8 - K_4 Q_8 |Q_8| = 0 \\ h_{10} + K_2 Q_{10} |Q_{10}| - h_{11} - h_7 - h_9 - K_4 Q_9 |Q_9| = 0 \end{cases}$$

The $Q_i|Q_i|$ is very important because it allows for the losses to always work again the flow if a negative flow rate gets calculated. For this system, the flow rates should all be positive, but solvers may calculate a negative value before converging to the final result and having Q^2 may result in an

invalid solution. This same concept should be applied to the h_i functions. In the code, everywhere a `.h()` is seen, just know that this is what is being returned:

$$h = \frac{8Q|Q|}{g\pi^2 D^4}\left(\frac{L}{D}f + K + C \cdot f_T\right)$$

3.5 Unbalanced Solution

With no corrections to the system, this is the result.

[6]:
```
# Define known constants
K1, K2, K3, K4 = 4.5, 4.5, 4.5, 10  # Loss coefficients

rho = 47.3/32.174  # In slugs per cubic feet
mu = 2.72/(3600*32)  # In slugs per (ft s) or lbf*s per ft squared
epsilon = 0.0005  # In ft

D20, D12, D10, D8 = np.array([19.624, 12.438, 10.482, 8.407])/12  # Diameters␣
 ↪in ft

def unbalanced(x):
    Q1, Q2, Q3, Q4, Q5, Q6, Q7, Q8, Q9, Q10, Q11 = x
    # all expressions need to be set to zero
    return [
        Q1 - Q2 - Q3,
        Q6 - Q2 - Q4,
        Q3 - Q4 - Q5,
        Q5 - Q10 - Q7,
        Q7 - Q8 - Q9,
        Q11 - Q6 - Q7,
        K1*Q2*abs(Q2) + p2.h(Q2) + p6.h(Q6) + p11.h(Q11) + p1.h(Q1) - 150 + 0.
 ↪1*Q1*abs(Q1),
        p4.h(Q4) + K3*Q4*abs(Q4) - p2.h(Q2) - K1*Q2*abs(Q2) + p3.h(Q3),
        p7.h(Q7) + p8.h(Q8) + K4*Q8*abs(Q8) - p6.h(Q6) - p4.h(Q4) -␣
 ↪K3*Q4*abs(Q4) + p5.h(Q5),
        p9.h(Q9) + K4*Q9*abs(Q9) - p8.h(Q8) - K4*Q8*abs(Q8),
        p10.h(Q10) + K2*Q10*abs(Q10) - p11.h(Q11) - p9.h(Q9) - K4*Q9*abs(Q9) -␣
 ↪p7.h(Q7)
    ]

p1 = Pipe(D20, 2840, epsilon, rho, mu, K=2.58, C=8)
p2 = Pipe(D10, 2380, epsilon, rho, mu, K=3.93, C=8)
p3 = Pipe(D12, 1300, epsilon, rho, mu, K=3.18, C=8)
p4 = Pipe(D10, 1630, epsilon, rho, mu, K=3.33, C=8)
p5 = Pipe(D12, 3000, epsilon, rho, mu, K=4.68, C=8)
p6 = Pipe(D12, 5000, epsilon, rho, mu, K=4.68, C=8)
p7 = Pipe(D12, 1580, epsilon, rho, mu, K=3.38, C=8)
p8 = Pipe(D8, 1550, epsilon, rho, mu, K=3.33, C=8)
p9 = Pipe(D8, 1550, epsilon, rho, mu, K=5.28, C=8)
```

```
p10 = Pipe(D10, 5130, epsilon, rho, mu, K=6.78, C=8)
p11 = Pipe(D12, 1875, epsilon, rho, mu, K=3.33, C=8)

pipes = [p1, p2, p3, p4, p5, p6, p7, p8, p9, p10, p11]

Q_guess = np.array([5, 1, 4, 2, 2, 3, 1, 3, -2, 1, 4])  # This does satisfy␣
 ↪mass conservation
unbalanced_sol = fsolve(unbalanced, Q_guess)
unbalanced_sol
```

[6]:
```
array([9.8135717 , 2.68840988, 7.12516182, 1.87884667, 5.24631516,
       4.56725654, 2.57075517, 1.28976604, 1.28098913, 2.67555999,
       7.13801171])
```

As you can see from above, the analysis provided in the code appendix is really close to the solution required. A function can be created to capture this repetitive process of testing the flow rates to see if they lie within the boundaries.

[7]:
```
# Making a test function
# Returns a dataframe

def test(pipes_, flow_rates):
    lines = range(1, len(pipes_) + 1)
    Q_values, V_values = [], []
    AHU_1_3, AHU_4 = [], []
    V_less = []

    for pipe_, Q_, i in zip(pipes_, flow_rates, lines):
        Q_values.append(Q_)
        V_ = pipe_.V(Q_)
        V_values.append(V_)

        if V_ < 9:
            V_less.append('Yes')
        else:
            V_less.append('No')

        if i in [2, 4, 10]:
            if 2.67 <= Q_ < 2.77:
                AHU_1_3.append('Yes')
            else:
                AHU_1_3.append('No')
        else:
            AHU_1_3.append('-')

        if i in [8, 9]:
            if 1.25 <= Q_ < 1.3:
                AHU_4.append('Yes')
```

```
            else:
                AHU_4.append('No')
        else:
            AHU_4.append('-')

    return pd.DataFrame({'Pipe': lines, 'Q': Q_values, 'V': V_values, 'V<9':␣
↪V_less, '2.67<=Q<2.77': AHU_1_3, r'1.25<=Q<1.3': AHU_4})

unbalanced_test = test(pipes, unbalanced_sol)
# unbalanced_test
display(Latex(unbalanced_test.to_latex(index=False)))
```

Pipe	Q	V	V<9	2.67<=Q<2.77	1.25<=Q<1.3
1	9.813572	1.747077	Yes	-	-
2	2.688410	5.879684	Yes	Yes	-
3	7.125162	7.860092	Yes	-	-
4	1.878847	4.109129	Yes	No	-
5	5.246315	5.787450	Yes	-	-
6	4.567257	5.038350	Yes	-	-
7	2.570755	2.835918	Yes	-	-
8	1.289766	6.816820	Yes	-	Yes
9	1.280989	6.770432	Yes	-	Yes
10	2.675560	5.851580	Yes	Yes	-
11	7.138012	7.874267	Yes	-	-

The work from the main pump is not enough especially in line 4. Adding booster pumps is to be investigated.

3.6 Balanced Solution

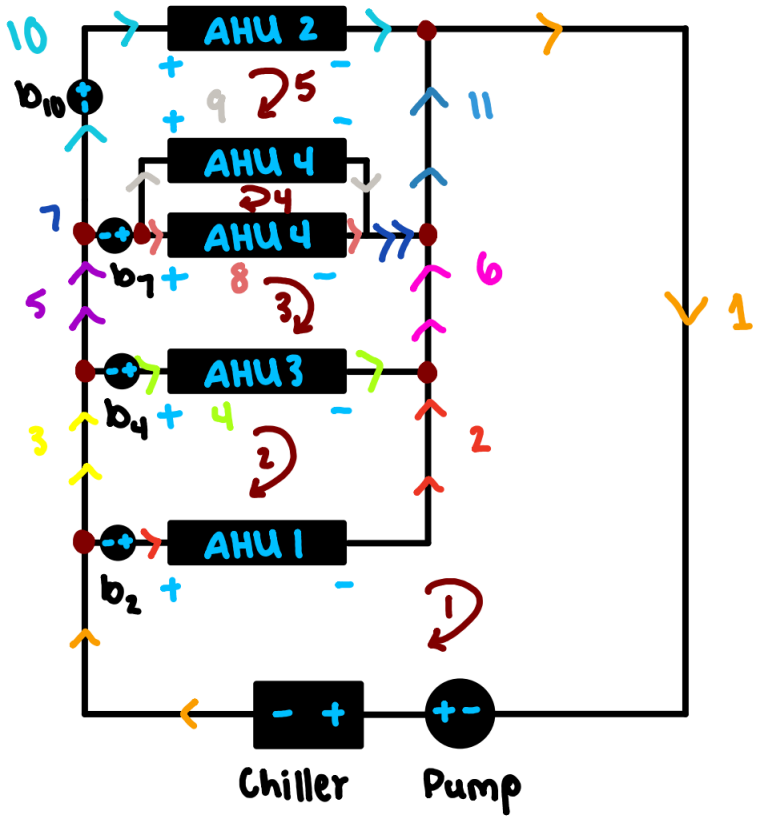

Figure 7: Booster Balance

The balanced solution will add booster pumps to line 2, 4, 7, and 10. This changes the system of equations to,

$$
\begin{cases}
Q_1 = Q_3 + Q_2 \\
Q_3 = Q_4 + Q_5 \\
Q_6 = Q_2 + Q_4 \\
Q_5 = Q_7 + Q_{10} \\
Q_7 = Q_8 + Q_9 \\
Q_{11} = Q_6 + Q_7 \\
h_2 + K_1 Q_2 |Q_2| + h_6 + h_{11} + h_1 - 150 + 0.1 Q_1 |Q_1| - b_2 = 0 \\
h_4 + K_3 Q_4 |Q_4| - h_2 - K_1 Q_2 |Q_2| + h_3 + b_2 - b_4 = 0 \\
h_7 + h_8 + K_4 Q_8 |Q_8| - h_6 - h_4 - K_3 Q_4 |Q_4| + h_5 + b_4 - b_7 = 0 \\
h_9 + K_4 Q_9 |Q_9| - h_8 - K_4 Q_8 |Q_8| = 0 \\
h_{10} + K_2 Q_{10} |Q_{10}| - h_{11} - h_7 - h_9 - K_4 Q_9 |Q_9| + b_7 - b_{10} = 0
\end{cases}
$$

[8]:
```
def balanced(x, b2, b4, b7, b10):
    Q1, Q2, Q3, Q4, Q5, Q6, Q7, Q8, Q9, Q10, Q11 = x
    # all expressions need to be set to zero
    return [
        Q1 - Q2 - Q3,
```

```
        Q6 - Q2 - Q4,
        Q3 - Q4 - Q5,
        Q5 - Q10 - Q7,
        Q7 - Q8 - Q9,
        Q11 - Q6 - Q7,
        K1*Q2*abs(Q2) + p2.h(Q2) + p6.h(Q6) + p11.h(Q11) + p1.h(Q1) - 150 + 0.
 ↪1*Q1*abs(Q1) - b2,
        p4.h(Q4) + K3*Q4*abs(Q4) - p2.h(Q2) - K1*Q2*abs(Q2) + p3.h(Q3) + b2 -␣
 ↪b4,
        p7.h(Q7) + p8.h(Q8) + K4*Q8*abs(Q8) - p6.h(Q6) - p4.h(Q4) -␣
 ↪K3*Q4*abs(Q4) + p5.h(Q5) + b4 - b7,
        p9.h(Q9) + K4*Q9*abs(Q9) - p8.h(Q8) - K4*Q8*abs(Q8),
        p10.h(Q10) + K2*Q10*abs(Q10) - p11.h(Q11) - p9.h(Q9) - K4*Q9*abs(Q9) -␣
 ↪p7.h(Q7) + b7 - b10
    ]

b = 26, 55, 16, 9
balanced_solution = fsolve(balanced, Q_guess, args=(*b, ))
display(Latex(test(pipes, balanced_solution).to_latex(index=False)))
```

Pipe	Q	V	V<9	2.67<=Q<2.77	1.25<=Q<1.3
1	10.560735	1.880092	Yes	-	-
2	2.675529	5.851513	Yes	Yes	-
3	7.885206	8.698532	Yes	-	-
4	2.681127	5.863755	Yes	Yes	-
5	5.204080	5.740858	Yes	-	-
6	5.356656	5.909172	Yes	-	-
7	2.523334	2.783605	Yes	-	-
8	1.265973	6.691066	Yes	-	Yes
9	1.257361	6.645548	Yes	-	Yes
10	2.680746	5.862922	Yes	Yes	-
11	7.879989	8.692777	Yes	-	-

The unbalanced solution with a main pump head of 150 feet was not enough to reach the minimum requirements for the flow rates. The analysis above shows that adding a head of 26 feet to line 2, 55 feet to line 4, 16 feet to line 7, and 9 feet to line 10 is just enough to satisfy the requirements.

Another option for reaching the requirements is to increase the head to an extent that the flow rates go over the maximum limit, then add losses until the flow rate is within its appropriate bounds. Changing the head to 300 ft (double) will be enough for all flow rates to be either within bounds, or over the bounds.

[9]:
```
# Changing the head to 300 ft

def overload(x):
    Q1, Q2, Q3, Q4, Q5, Q6, Q7, Q8, Q9, Q10, Q11 = x
    # all expressions need to be set to zero
```

```
    return [
        Q1 - Q2 - Q3,
        Q6 - Q2 - Q4,
        Q3 - Q4 - Q5,
        Q5 - Q10 - Q7,
        Q7 - Q8 - Q9,
        Q11 - Q6 - Q7,
        K1*Q2*abs(Q2) + p2.h(Q2) + p6.h(Q6) + p11.h(Q11) + p1.h(Q1) - 300 + 0.
 ↪1*Q1*abs(Q1),
        p4.h(Q4) + K3*Q4*abs(Q4) - p2.h(Q2) - K1*Q2*abs(Q2) + p3.h(Q3),
        p7.h(Q7) + p8.h(Q8) + K4*Q8*abs(Q8) - p6.h(Q6) - p4.h(Q4) -␣
 ↪K3*Q4*abs(Q4) + p5.h(Q5),
        p9.h(Q9) + K4*Q9*abs(Q9) - p8.h(Q8) - K4*Q8*abs(Q8),
        p10.h(Q10) + K2*Q10*abs(Q10) - p11.h(Q11) - p9.h(Q9) - K4*Q9*abs(Q9) -␣
 ↪p7.h(Q7)
    ]

overload_sol = fsolve(overload, Q_guess)
overload_test = test(pipes, overload_sol)
# overload_test
display(Latex(overload_test.to_latex(index=False)))
```

Pipe	Q	V	V<9	2.67<=Q<2.77	1.25<=Q<1.3
1	13.960621	2.485362	Yes	-	-
2	3.822000	8.358901	Yes	No	-
3	10.138621	11.184376	No	-	-
4	2.675616	5.851703	Yes	Yes	-
5	7.463005	8.232782	Yes	-	-
6	6.497616	7.167817	Yes	-	-
7	3.657563	4.034825	Yes	-	-
8	1.835057	9.698856	No	-	No
9	1.822506	9.632517	No	-	No
10	3.805442	8.322688	Yes	No	-
11	10.155179	11.202642	No	-	-

The unbalance of overloading the main pump is so severe that balancing won't occur until adding values of 50 to most of the line's loss coefficient. For this reason, this option will not be chosen.

3.7 Power Consumption

The power added to the fluid may be calculated using,

$$power = \rho Q W_s$$

W_s needs to be in units of $\frac{ft^2}{s^2}$, so it is necessary to multiply by $g\,\frac{ft}{s^2}$, but if ρ is in $\frac{lbm}{ft^3}$, then it won't be necessary to multiply by g. For the power consumed using the main pump only with a head of 300 feet is,

[10]:
```
# power for main pump only
# using the flow rate in line 1 that was calculated above
power_A = 47.3*13.96*300
power_A/550  # In hp
```

[10]:
```
360.168
```

The power for the booster pump option is the summation of all the pump powers for each line.

[11]:
```
# Getting the power for the booster pump option
power_B = 47.3*(10.560735*150 +  2.675529*26 + 2.681127*55 + 2.523334*16 + 2.
↪680746*9)
power_B/550
```

[11]:
```
160.444700042
```

4 Verification

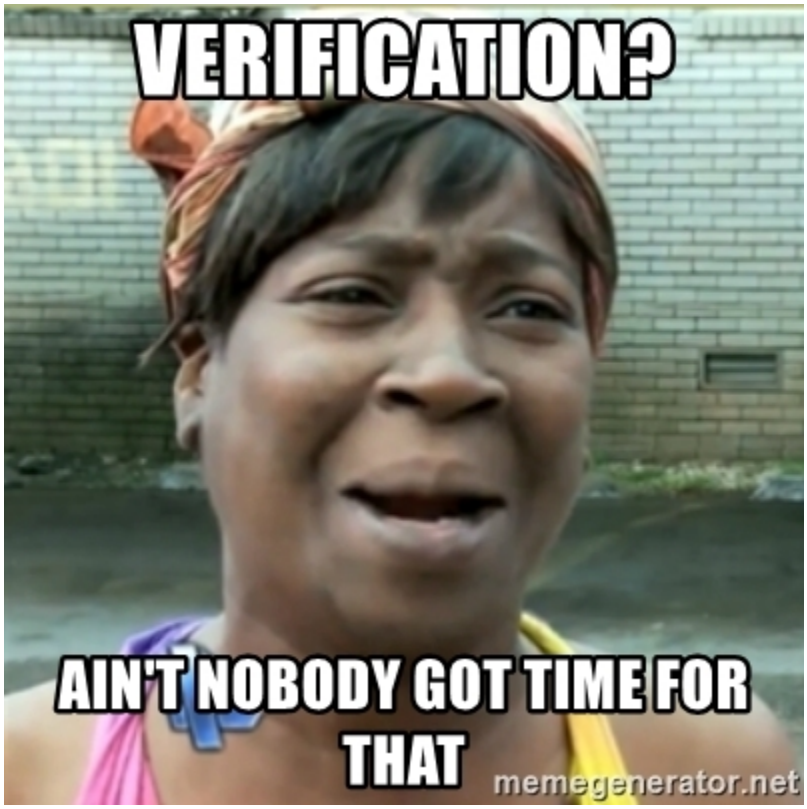

Figure 8: Me Right Now

The math for the booster pump analysis can be confirmed with a hardy cross solution.

[12]:
```
# Getting the hardy cross solution

# Connection matrix
N = np.transpose([
```

```
    [1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1],
    [0, -1, 1, 1, 0, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, -1, 1, -1, 1, 1, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, -1, 1, 0, 0],
    [0, 0, 0, 0, 0, 0, -1, 0, -1, 1, -1]
])

# Losses/gains in each line
b2_, b4_, b7_, b10_ = b
h = [
    lambda Q: 0.1*Q*abs(Q) - 150,
    lambda Q: K1*Q*abs(Q) - b2_,
    lambda Q: 0,
    lambda Q: K3*Q*abs(Q) - b4_,
    lambda Q: 0,
    lambda Q: 0,
    lambda Q: -b7_,
    lambda Q: K4*Q*abs(Q),
    lambda Q: K4*Q*abs(Q),
    lambda Q: K2*Q*abs(Q) - b10_,
    lambda Q: 0
]

dh = [
    lambda Q: 2*0.1*abs(Q),
    lambda Q: 2*K1*abs(Q),
    lambda Q: 0,
    lambda Q: 2*K3*abs(Q),
    lambda Q: 0,
    lambda Q: 0,
    lambda Q: 0,
    lambda Q: 2*K4*abs(Q),
    lambda Q: 2*K4*abs(Q),
    lambda Q: 2*K2*abs(Q),
    lambda Q: 0
]

hardy_solution = hardy_cross(pipes, Q_guess, N, h=h, dh=dh)
display(Latex(test(pipes, hardy_solution).to_latex(index=False)))
```

Pipe	Q	V	V<9	2.67<=Q<2.77	1.25<=Q<1.3
1	10.560432	1.880038	Yes	-	-
2	2.675533	5.851522	Yes	Yes	-
3	7.884899	8.698193	Yes	-	-
4	2.681110	5.863719	Yes	Yes	-
5	5.203789	5.740538	Yes	-	-
6	5.356643	5.909158	Yes	-	-
7	2.523206	2.783464	Yes	-	-
8	1.265945	6.690919	Yes	-	Yes
9	1.257261	6.645019	Yes	-	Yes
10	2.680583	5.862567	Yes	Yes	-
11	7.879849	8.692622	Yes	-	-

The Hardy Cross solution is the same as the Kirchhoff solution, but the tolerance of the Hardy Cross solution is greater than the numerical solver.

5 Results and Discussion

The recommended choice for this piping network is the option of adding booster pumps. There are some downsides for adding booster pumps, such as additional maintenance and initial cost. However, adding booster pumps allows for more control over the system, and consumes far less power than the alternative choice of overloading the main pump. It is better to keep adding power until it's just enough to reach the minimum requirements than to overshoot.

The option of overloading the pump, then adding loss was so out of balance that doubling the main pump was the only means to push the fluid hard enough for line 4 before adding losses elsewhere. This resulted in a power consumption that is so out of reach that it should not be considered. Although the flow rate used in the calculation was before the loss was added, it is reasonable to assume this because reducing the flow rate above by 1/2 will still be considerably larger than the booster pump choice (180 hp).

6 Code Appendix

The following code was used to get a good initial idea of the diameters to be selected as well as a starting pump head for the main line.

[ ]:
```
from msu_esd import Pipe
from scipy.optimize import fsolve
import itertools
import numpy as np
import csv


def network(x, head):
    Q1_, Q2_, Q3_, Q4_, Q5_, Q6_, Q7_, Q8_, Q9_, Q10_, Q11_ = x

    return [
        Q1_ - Q2_ - Q3_,
        Q6_ - Q2_ - Q4_,
        Q3_ - Q4_ - Q5_,
        Q5_ - Q10_ - Q7_,
        Q7_ - Q8_ - Q9_,
        Q11_ - Q6_ - Q7_,
        K_ahu1*Q2_*abs(Q2_) + p2.h(Q2_) + p6.h(Q6_) + p11.h(Q11_) + p1.h(Q1_) -␣
 ↪head + 0.1*Q1_*abs(Q1_),
        p4.h(Q4_) + K_ahu3*Q4_*abs(Q4_) - p2.h(Q2_) - K_ahu1*Q2_*abs(Q2_) + p3.
 ↪h(Q3_),
        p7.h(Q7_) + p8.h(Q8_) + K_ahu4*Q8_*abs(Q8_) - p6.h(Q6_) - p4.h(Q4_) -␣
 ↪K_ahu3*Q4_*abs(Q4_) + p5.h(Q5_),
        p9.h(Q9_) + K_ahu4*Q9_*abs(Q9_) - p8.h(Q8_) - K_ahu4*Q8_*abs(Q8_),
        p10.h(Q10_) + K_ahu2*Q10_*abs(Q10_) - p11.h(Q11_) - p9.h(Q9_) -␣
 ↪K_ahu4*Q9_*abs(Q9_) - p7.h(Q7_)
    ]


K_ahu1, K_ahu2, K_ahu3, K_ahu4 = 4.5, 4.5, 4.5, 10

rho = 47.3/32.174
mu = 2.72/(3600*32)
epsilon = 0.0005

D = np.array([8.407, 10.482, 12.438, 13.688, 15.67, 17.67, 19.624])/12

p1 = Pipe(D[-1], 2840, epsilon, rho, mu, K=2.58, C=8)
p2 = Pipe(D[1], 2380, epsilon, rho, mu, K=3.93, C=8)
p3 = Pipe(D[2], 1300, epsilon, rho, mu, K=3.18, C=8)
p4 = Pipe(D[1], 1630, epsilon, rho, mu, K=3.33, C=8)
p5 = Pipe(D[2], 3000, epsilon, rho, mu, K=4.68, C=8)
```

```
p6 = Pipe(D[2], 5000, epsilon, rho, mu, K=4.68, C=8)
p7 = Pipe(D[2], 1580, epsilon, rho, mu, K=3.38, C=8)
p8 = Pipe(D[0], 1550, epsilon, rho, mu, K=3.33, C=8)
p9 = Pipe(D[0], 1550, epsilon, rho, mu, K=5.28, C=8)
p10 = Pipe(D[1], 5130, epsilon, rho, mu, K=6.78, C=8)
p11 = Pipe(D[2], 1875, epsilon, rho, mu, K=3.33, C=8)

pipes = [eval(f'p{i}') for i in range(1, 12)]

Q_guess = np.array([5, 1, 4, 2, 2, 3, 1, 3, -2, 1, 4])

head_test_file = open('Tests/Overall.csv', 'w', newline='')

for pump_head in range(50, 211, 10):
    print('[Starting]', pump_head)
    products = itertools.product(D, repeat=4)
    tests = []
    count = 0
    for diameters in products:
        count += 1

        d1, d2, d3, d4 = diameters
        p1.D = d1
        p3.D, p5.D, p6.D, p7.D, p11.D = d2, d2, d2, d2, d2
        p2.D, p4.D, p10.D = d3, d3, d3
        p8.D, p9.D = d4, d4

        # noinspection PyTupleAssignmentBalance
        Q1, Q2, Q3, Q4, Q5, Q6, Q7, Q8, Q9, Q10, Q11 = fsolve(network, Q_guess,␣
↪args=(pump_head, ))
        Q = [eval(f'Q{i}') for i in range(1, 12)]

        test = [diameters]
        # The order of the list is:
        # 1) All the velocities are under 9 ft/s
        # 2) 2.67 <= Q2 < 2.77
        # 3) 2.67 <= Q4 < 2.77
        # 4) 1.25 <= Q8 < 1.3
        # 5) 1.25 <= Q9 < 1.3
        # 6) 2.67 <= Q10 < 2.77

        # Test the velocities
        velocity_test = all([pipe.V(Q_value) < 9 for pipe, Q_value in␣
↪zip(pipes, Q)])
        test.append(velocity_test)

        # Test the flow rates
```

```
        Q2_test = 2.67 <= Q2 < 2.77
        Q4_test = 2.67 <= Q4 < 2.77
        Q8_test = 1.25 <= Q8 < 1.3
        Q9_test = 1.25 <= Q9 < 1.3
        Q10_test = 2.67 <= Q10 < 2.77
        for test_ in [Q2_test, Q4_test, Q8_test, Q9_test, Q10_test]:
            test.append(test_)

        tests.append(test)

    with open(f'Tests/Test{pump_head}.csv', 'w', newline='') as f:
        writer = csv.writer(f)
        writer.writerows(tests)

    head_tests = [f'{pump_head} ft']
    gt_3, gt_4, gt_5 = [], [], []
    for i, test in enumerate(tests):
        if test.count(True) > 3:
            gt_3.append(i)

        if test.count(True) > 4:
            gt_4.append(i)

        if test.count(True) > 5:
            gt_5.append(i)

    for gt in [gt_3, gt_4, gt_5]:
        head_tests.append(gt)

    if any(gt_3):
        print(f'{len(gt_3)} indices greater than 3 true values.')

    if any(gt_4):
        print(f'{len(gt_4)} indices greater than 4 true values.')

    if any(gt_5):
        print(f'{len(gt_5)} indices greater than 5 true values.')

    head_writer = csv.writer(head_test_file)
    head_writer.writerow(head_tests)

    print('[Ending]', pump_head)

head_test_file.close()
```