Draining Times for Varying Volumes and Heights of a Trapezoidal Frustum

Yejun M. Kim

November 28, 2024

Contents

1 Introduction

The proposed design will simplify the process of collecting a precise amount of water from a Brita 18-cup tank by eliminating the need to verify the amount of water poured and shut off the spigot. In this design, three subsystems were integrated together to create the AquaMetric attachment, which were the electrical circuit subsystem, the solid circuit enclosure with a switch for the spigot, and the code model required for calculations. Calculating the drain time of the tank required the collection of data and adjustments to the theoretical approaches to the research question.

While this process could easily be resolved by using load cells to directly find the weight of water desired and toggling the switch when an appropriate amount has drained, the research question was deemed more interesting to solve through a theoretical approach.

The setup involves Brita LP's Extra Large 18-cup filtered water dispenser, as shown in fig. 1, and a custom circuit with a 3D-printed enclosure. For data

collection, the 2SMPP-02 pressure sensor with tubing was used to calculate the hydrostatic pressure, and a common kitchen scale for measuring volume, shown in fig. 2. The height of the water initially in the tank determines the time it takes to dispense a given amount of water. The goal was to verify the theoretical relationship between the starting height and ending height of the water in the tank and the time spent dispensing water. The theoretical assumptions made for the tank were that the flow is inviscid, incompressible, quasi-steady from points 1 to 2 with a free jet at 2, constant gravity, and uniform flow at 2. Furthermore, some simplifying assumptions were made, such as the exit hole being at the bottom of the container, the exit area being equal to the area at the end of the spigot, and the tank not currently filtering water.

$$P + \frac{1}{2}\rho v^2 + \rho g h = C \tag{1}$$

The Bernoulli equation states that the sum of the static pressure, dynamic pressure, and potential energy at one point is equal to the same sum at another point. Due to the draining of the tank, the water at the top is at rest and the exiting water has a non-zero velocity. Both points have static pressure equal to atmospheric pressure and potential energy from the height difference alone.

$$P_1+\frac{1}{2}\rho v_1^2+\rho g h_1 = P_2+\frac{1}{2}\rho v_2^2+\rho g h_2 \rightarrow \rho g(h_1-h_2) = \frac{1}{2}\rho v_2^2 \rightarrow v_2 = \sqrt{2g\Delta h} \tag{2}$$

If accounting for major and minor losses, the equation becomes a bit more complicated. Let Δp_{loss} represent the combined effects of major and minor losses, as shown below. Then the velocity at the outlet becomes:

$$\Delta p_{loss} = \frac{\rho v^2}{2}(f_D \frac{L}{D_H} + \Sigma K)$$

$$P_1 + \frac{1}{2}\rho v_1^2 + \rho g h_1 = P_2 + \frac{1}{2}\rho v_2^2 + \rho g h_2 - \Delta p_{loss}$$

$$\rho g(h_1 - h_2) = \frac{1}{2}\rho v_2^2 - \Delta p_{loss} \rightarrow \rho g(h_1 - h_2) = \frac{1}{2}\rho v_2^2(1 - (f_D \frac{L}{D_H} + \Sigma K))$$

Since we do not have the means to calculate the f_D and K as necessary, we will approximate it as the error term ϵ_v.

$$v_2 = \sqrt{\frac{2g\Delta h}{1 - (f_D \frac{L}{D_H} + \Sigma K)}} = \sqrt{\frac{2g\Delta h}{\epsilon_v}} \tag{3}$$

Given the desired volume output, the final height can be calculated by deriving and using the following equation for trapezoidal volume:

$$A_0 = w_0 d_0, A_h = (w_0 + \frac{h(w_t - w_0)}{h_t 0})(d_0 + \frac{h(d_t - d_0)}{h_t 0}), A_t = w_t d_t$$

$$V_{total} = \int_0^{h_t 0} A_h \, dh = \frac{h_t 0}{6}(w_t d_0 + w_0 d_t + 2(w_0 d_0 + w_t d_t)) \tag{4}$$

$$\begin{aligned}\Delta V_{1\to 2} = \int_{h_1}^{h_2} A_h \, dh = \frac{h_1 - h_2}{6}(6w_0 d_0 + 3(w_s d_0 h_1 \\ + w_0 d_s h_1 + w_s d_0 h_2 + w_0 d_s h_2) + 2w_s d_s (h_1^2 + h_1 h_2 + h_2^2))\end{aligned} \tag{5}$$

The initial height h_1 will be found using the pressure sensor and change in volume $V_{1\to 2}$ will be provided as input. All other values are known in equation (4) so h_2 is solved for numerically. For clarification, the change in volume is expressed as a negative number but its magnitude is equal to the user input. The verification of the formula is provided through constraining it in two different ways to retrieve the formula of a rectangular prism and pyramidal frustum in Appendix C.

$$A_h = (w_0 + \frac{h(w_t - w_0)}{h_t 0})(d_0 + \frac{h(d_t - d_0)}{h_t 0}) - \epsilon_V$$

$$V_{total} = \int_0^{h_{t0}} A_h \, dh = \frac{h_{t0}}{6}(w_t d_0 + w_0 d_t + 2(w_0 d_0 + w_t d_t) - h_{t0}\epsilon_V) \tag{6}$$

$$\begin{aligned}\Delta V_{1\to 2} = V_{corrected} = \int_{h_1}^{h_2} A_h \, dh = \frac{h_1 - h_2}{6}(6w_0 d_0 + 3(w_s d_0 h_1 + \\ w_0 d_s h_1 + w_s d_0 h_2 + w_0 d_s h_2) + 2w_s d_s (h_1^2 + h_1 h_2 + h_2^2)) - (h_1 - h_2)\epsilon_V\end{aligned} \tag{7}$$

To reconcile the error in volume loss on the corners of the tank, a correcting factor that varies linearly with height is added to (3) and (4) to get the above equations. The correcting factor ϵ represents the slight reduction in dV caused by the four rounded corners.

$$\frac{\delta}{\delta t}\iiint_V \rho dV + \iint_S \rho \vec{v} d\vec{A} = 0 \tag{8}$$

The integral continuity equation states that the sum of the net influx of mass in the control surface and the change in mass, measured over the control volume should equal 0. In context, this equation allows us to solve for the time elapsed for a desired amount of fluid to drain given some starting height.

$$\frac{\delta}{\delta t}\iiint_V \rho dV \to \frac{dV}{dt}\rho \to \rho\frac{dh}{dt}((w_0 + w_s h)(d_0 + d_s h) - \epsilon_V), \iint_S \rho \vec{u}\cdot\vec{n} dA \to \rho\sqrt{\frac{2gh}{\epsilon_v}} A_e$$

$$\rho\frac{dh}{dt}((w_0 + w_s h)(d_0 + d_s h) - \epsilon_V) = -\rho\sqrt{\frac{2gh}{\epsilon_v}} A_e$$

$$\int_{h_1}^{h_2} \frac{w_0 d_0 - \epsilon_V}{\sqrt{h}} + (w_s d_0 + w_0 d_s)\sqrt{h} + w_s d_s h^{1.5} dh = -\int_0^t \sqrt{\frac{2g}{\epsilon_v}} A_e dt$$

$$t_{h_1 \to h_2} = \sqrt{\epsilon_v} \frac{2(w_0 d_0 - \epsilon_V)(\sqrt{h_1} - \sqrt{h_2}) + \frac{2}{3}(w_s d_0 + w_0 d_s)(h_1^{1.5} - h_2^{1.5}) + \frac{2}{5} w_s d_s (h_1^{2.5} - h_2^{2.5})}{\sqrt{2g} A_e} \tag{9}$$

This can be expressed as the difference between the general formula for drainage times from some starting height h, at starting heights h_1 and h_2. This is found by integrating from h to 0 instead:

$$t = \sqrt{\epsilon_v} \frac{2(w_0 d_0 - \epsilon_V)(\sqrt{h}) + \frac{2}{3}(w_s d_0 + w_0 d_s)(h^{1.5}) + \frac{2}{5} w_s d_s (h^{2.5})}{\sqrt{2g} A_e} \tag{10}$$

Put together, the time taken to empty the tank some volume from a starting height should correspond to the final height measurement. The combined effect of the errors ϵ_V and ϵ_v change the coefficients of the quintic polynomial of $\sqrt{h}$ representing the time equation to an unknown degree, and can only be found through regression.

2 Procedure

Samples at various starting heights and volume will be taken, so that the ending height is recorded and the time to drain is recorded as well. As shown in fig. 1, there is a ruler on the outside of the filter for the approximate height of the water and a collection container for the output volume. A common kitchen scale is used to measure the change in volume in mL, and a phone stopwatch is used for the draining time.

The water height in the tank is measured prior to draining, the container zeroed, and the stopwatch prepared for the trial. After the tank is drained, the water in the output container is weighed to get the exact amount of volume lost and the final height is measured as well. With the collected data, a multivariate cubic model is fit for (X = h_1, h_2) and y = V, and a quintic model is fit for (X = $\sqrt{h_1}, \sqrt{h_2}$) and y = t, both using sklearn on Python.

The correction factor ϵ_V for volume is proportional to difference between heights and is theoretically included in the coefficient of $h_1 - h_2$. By contrast, the error terms for time calculation are more complicated, and have a nonlinear impact on the coefficients of the quintic, specifically impacting the first, third and fifth order terms. Acceptable bounds for 50 mL for volume change and 1 second for time.

3 Results

The data collected are shown in Table I in Appendix B. There are four columns, with the first two representing h_1 and h_2 in mm, the third being actual volume output in mL, and the fourth the time it took for the drain from h_1 to h_2.

The two-variable cubic regression using h_1 and h_2 for volume resulted in the following model:

$$\begin{aligned} V_{model} = -53.8 - 30.7h_1 + 33.1h_2 + 0.000494h_1^2 \\ +0.0385h_1h_2 - 0.0712h_2^2 - 0.000108h_1^3 + 0.000192h_1^2h_2 \\ -0.000280h_1h_2^2 + 0.000324h_2^3 \end{aligned} \tag{11}$$

With the exception of the h_1^2 term, the coefficients of most terms of the same order are in the same approximate range. Figures 6 and 7 display visualizations of the predicted graph with corrections and the regression model, respectively, for comparison. The MSE for the regression model was found to be 195.6, compared to the MSE of the predicted graph with corrections at 452.9. Although this MSE still seems relatively large, most data points measured were for volumes between 140 to 4400, and the RMSE is 14.0, which is in an acceptable error range. The relevant code is in Appendix C.

The two-variable quintic regression using h_1 and h_2 for volume resulted in the following model:

$$\begin{aligned} t_{model} = 113 - 88.4\sqrt{h_1} + 4.18\sqrt{h_2} + 17.6h_1 + 6.30\sqrt{h_1h_2} \\ -0.672h_2 - 3.45\sqrt{h_1^3} + 1.76h_1\sqrt{h_2} - 1.90\sqrt{h_1}h_2 \\ +0.549\sqrt{h_2^3} - 0.0206h_1^2 + 0.699h_1\sqrt{h_1h_2} - 0.852h_1h_2 \\ +0.473h_2\sqrt{h_1h_2} - 0.107h_2^2 - 0.00627\sqrt{h_1^5} + 0.0225h_1^2\sqrt{h_2} \\ -0.0610h_1h_2\sqrt{h_1} + 0.06535h_1h_2\sqrt{h_2} - 0.0313\sqrt{h_1}h_2^2 + 0.00608\sqrt{h_2^5} \end{aligned} \tag{12}$$

The MSE for this regression is 0.297, which implies that it is a good fit. For the average data point, it deviates by around 0.545 seconds, which can be accounted for by human error and lack of timing precision. Also, the coefficients are within appropriate ranges relative to each other, and is fairly consistent with the predicted equation. The adjusted MSE for the original was 10.2, so our model demonstrates a better fit to the data. The relevant code is in Appendix C, and figures 8 and 9 display the visualizations of the prediction with corrections and model.

After correcting both models using a linear error term, the MSE is increases to 1451 for volume and drops to 0.640 for time. The effect of error terms ϵ_V and

combined effects of ϵ_V and ϵ_v and factored in assuming a linear relationship. Comparing the errors of the regression models and linear correction factors, it is clear that the regression was more effective in capturing the behavior of the tank.

A major source of error is the lack of accuracy of data. The setup currently does not allow for accurate height measurements, as it is dependent on the position of the pressure sensor and is subject to significant human error. Also, due to the roughly cylindrical section descending from the top of the tank, it may be more accurate to use separate models for the volume and the time for heights with the cylindrical section and another set for heights without the cylindrical section, because the cylinder introduces a significant volume displacement. The exit hole is not located at the bottom of the tank, and this means the equations are only consistent for heights where the hole is submerged. As expected from this, the drainage time behavior changes drastically near the bottom, and the tank actually fails to drain completely, regardless of how much time is given.

4 Assembly

Using the corrected model, the second height is determined by the Secant root finding method given the change in volume, provided as input by the user, and the initial height, provided by the sensor. With the final height, initial height, and correcting factor, the time is calculated directly and in ms for the Arduino.

The user declares the desired volume via the potentiometer, followed by a button to initiate the program. The height is measured and fed as input to calculate the draining time t required, and a servo operates the spigot to turn it on and off for t seconds. A picture of the circuit enclosure is included in fig. 3-5, and a circuit diagram is provided in fig. 10.

In practice, the error for the volume was around 19mL, which is within acceptable bounds. This can be attributed to the error propagation of the volume model into the drainage time model. Overall, the project successfully delivered its goal of dispensing the correct amount of water given user input. Further information on the project can be found at the following github repo:

`https://github.com/Sagittarius-B-Astro/AquaMetric`

5 Appendix

5.1 A (Figures)

Figure 1: Tank with external ruler for height measurement

Figure 2: Scale used for weight measurement as shown on Amazon

Figure 3: Project prototype disassembled view

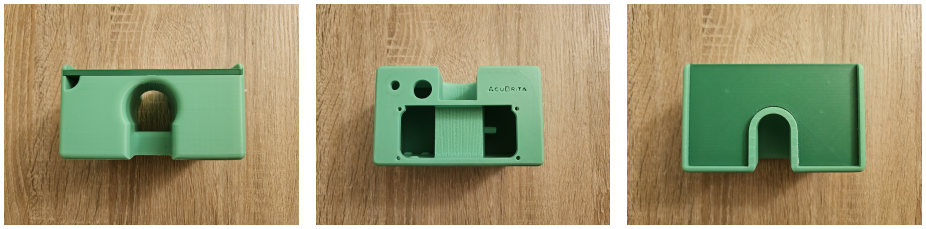

Figure 4: Project prototype top, front and back views

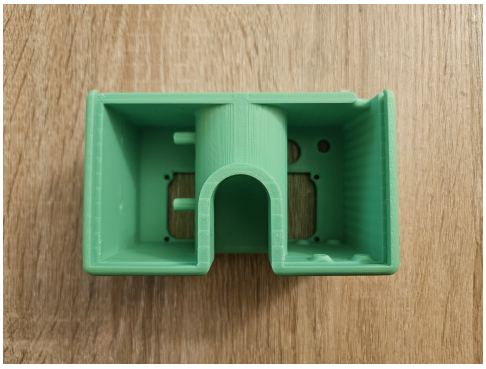

Figure 5: Internal view displaying electronic standoffs

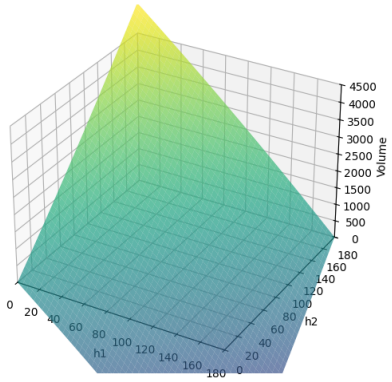

Figure 6: Plot of predicted Volume with error corrections

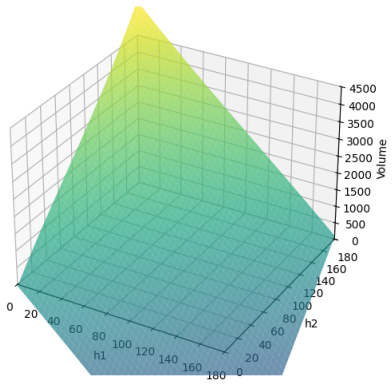

Figure 7: Plot of cubic regression model of Volume

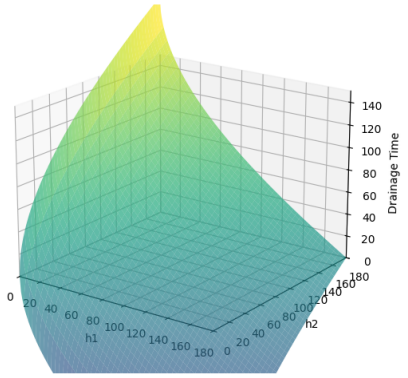

Figure 8: Plot of predicted Time with error corrections

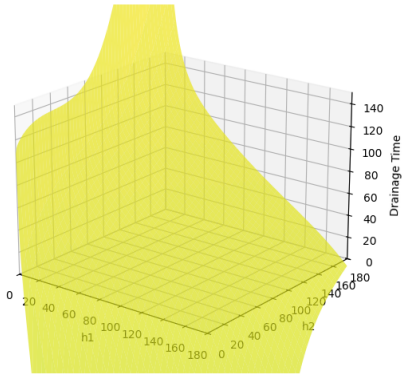

Figure 9: Plot of quintic regression model of Time

5.2 B (Tables)

Table I: Raw Volume and Time Data

h1 (mm)	h2 (mm)	Volume (mL)	Time (s)
85.5	114	825	22.54
7	153	4305	158.78
27	153	3698	102.40
47	153.5	3116	79.67
66	153	2550	61.27
87	153.5	1968	44.82
104	153	1477	32.52
126	153.5	838	17.77
147	153.5	214	4.22
18.5	143	3583	107.45
37.5	143	3031	81.92
57.5	142	2425	61.33
77.5	143.5	1908	45.14
97	143.5	1357	30.98
117	143	783	16.85
137	143.5	214	4.41
6.5	133	3637	130.63
27	132.5	3067	88.78
48	133	2432	64.48
66.5	133	1888	45.24
87	133	1314	30.97
106	133	769	17.31
123.5	132.5	274	5.84
18	123	3011	93.00
38	123	2440	68.84
57.5	123	1861	48.69
77	123	1306	32.49
97	122.5	748	17.26
116.5	122.5	182	4.01
8	113	3018	107.38
28.5	113	2417	74.25
44.5	113	1967	55.31
65	113	1377	35.98
85	113	792	19.63
106.5	113	188	4.42
16	104	2518	83.10
37	103.5	1920	56.74
53.5	103.5	1424	39.16

Continued on next page

Table continued from previous page

h1 (mm)	h2 (mm)	Volume (mL)	Time (s)
78	103.5	735	18.80
97	103	180	4.28
9.5	92.5	2382	87.21
26	92.5	1898	61.13
46	92	1304	38.99
68	92	734	20.38
85	92.5	192	4.87
18	83	1866	65.34
34.5	83	1396	44.64
58	83.5	727	21.40
76	83	216	5.87
9	73	1842	74.45
26.5	73	1347	46.90
48	73	710	22.34
67	73	171	4.92
17	63.5	1361	52.13
37	63.5	776	26.49
56.5	63.5	203	6.20
8.5	53	1301	60.22
27.5	53	753	28.27
46	53	201	6.83
17.5	42.5	728	30.91
36	42.5	201	7.42
7.5	32.5	711	42.19
25	32.5	217	9.04
14.5	22	201	9.93
7	12	145	15.94

5.3 C (Proofs and Code)

Prf. 1

For a rectangular prism, ws and ds both are equal to 0, since the values of the sides of a rectangular prism do not change with height. The formula for a rectangular prism is Hw_0d_0, where w_0 and d_0 are assumed to be the width and height of the rectangular base, and H = $h_1 - h_2$.

$$V_{h_1 \to h_2} = \int_{h_1}^{h_2} A_h \, dh = w_0 d_0 (h_1 - h_2) = H w_0 d_0$$

Prf. 2

For a pyramidal frustum, the constraint is that the ratio of the sides do not change with change in height. In algebraic terms, $\frac{w_1}{d_1} = \frac{w_2}{d_2} \to w_1 d_2 = w_2 d_1$. This further yields: $w_1 d_2 = w_2 d_1 \to (w_0 + w_s h_1)(d_0 + d_s h_2) = (w_0 + w_s h_2)(d_0 + d_s h_1) \to w_s d_0 h_2 + w_0 d_s h_2 = w_s d_0 h_1 + w_0 d_s h_1$

The formula for a pyramidal frustum is $V_{pyramid} = \frac{H}{3}(A_1 + A_2 + A_1 A_2)$, where $A_h = (w_s h + w_0)(d_s h + d_0)$ and $H = h_1 - h_2$.

$$V_{h_1 \to h_2} = \frac{h_1 - h_2}{6}(6w_0d_0 + 3(w_s d_0 h_1 + w_0 d_s h_1 + w_s d_0 h_2 + w_0 d_s h_2) + 2w_s d_s (h_1^2 + h_1 h_2 + h_2^2)$$

$$= \frac{H}{6}(2(w_0 d_0 + w_0 d_s h_1 + w_s d_0 h_1 + w_s d_s h_1^2) + 2(w_0 d_0 + w_0 d_s h_2 + w_s d_0 h_2 + w_s d_s h_2^2)$$

$$+ (2w_0 d_0 + 2w_s d_s h_1 h_2 + w_s d_0 h_2 + w_0 d_s h_2 + w_s d_0 h_1 + w_0 d_s h_1)$$

$$= \frac{H}{6}(2A_1 + 2A_2 + 2(w_0 d_0 + w_s d_s h_1 h_2 + w_0 d_s h_2 + w_s d_0 h_1)$$

$$= \frac{H}{3}(A_1 + A_2 + (w_s h_1 + w_0)(d_s h_2 + d_0) = \frac{H}{3}(A_1 + A_2 + w_1 d_2) = \frac{H}{3}(A_1 + A_2 + \sqrt{(w_1 d_2)^2})$$

$$= \frac{H}{3}(A_1 + A_2 + \sqrt{w_1 d_1 w_2 d_2}) = \frac{H}{3}(A_1 + A_2 + \sqrt{A_1 A_2})$$

```python
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.preprocessing import PolynomialFeatures
from sklearn import linear_model
from sklearn.metrics import mean_squared_error

# Constants
bottom_thick, wall_thick, ht0 = 18.5, 3, 176.7
Ae, g = 28.2743, 9807 # Circular Exit Area w/ d = 6mm, Gravity in mm/s^2
ws, ds = 12/(ht0-bottom_thick), 13/(ht0-bottom_thick)
w0, d0 = 110-2*wall_thick, 277-2*wall_thick

data = pd.read_csv("Brita Volume and Time Data.csv")
data = data[:-1]
h1_data = data["h1, mm"].values
h2_data = data["h2, mm"].values
vol_data = data["Volume, mL \r\n(actual)"].values
time_data = data["Time, sec \r\n(actual)"].values
```

Code 1: Initialization of variables and csv data

```python
def volume_pred(inputs):
    h1, h2 = inputs
    volume = 1 / 1000 * (h2 - h1) / 6 * (
        6 * w0 * d0 +
        3 * (d0 * ws * h2 + w0 * ds * h2 + d0 * ws * h1 + w0 * ds * h1) +
        2 * (ws * ds * h1**2 + ws * ds * h1 * h2 + ws * ds * h2**2))
    return volume

def time_pred(height):
    h1, h2 = height
    time = -(2 * w0 * d0 * (h1 - h2) + 2/3 * (ws * d0 + w0 * ds) * (h1**3 - h2**3) +
        2/5 * ws * ds * (h1**5 - h2**5)) / ((2 * g)**0.5 * Ae)
    return time

def create_volume_model(coef):
    def volume_model(inputs):
        h1, h2 = inputs
        vars = [np.ones_like(h1), h1, h2, h1**2, h1*h2, h2**2, h1**3, h1**2*h2, h1*h2**2, h2**3]
        result = sum(c * v for c, v in zip(coef, vars))
        return result
    return volume_model

def create_time_model(coef):
    def time_model(inputs):
        h1, h2 = inputs
        vars = [np.ones_like(h1), h1, h2, h1**2, h1*h2, h2**2, h1**3, h1**2*h2, h1*h2**2, h2**3, h1**4, h1**3*h2,
                h1**2*h2**2, h1*h2**3, h2**4, h1**5, h1**4*h2, h1**3*h2**2, h1**2*h2**3, h1*h2**4, h2**5]
        result = sum(c * v for c, v in zip(coef, vars))
        return result
    return time_model
```

Code 2: Code of volume and time equations

```python
def vol_multivar_cubic_reg():
    volX = np.array([h1_data, h2_data]).T
    voly = np.array(vol_data)
    volume_poly_model = PolynomialFeatures(degree = 3)
    volX_ = volume_poly_model.fit_transform(volX)

    volmod = linear_model.LinearRegression()
    volmod.fit(volX_, voly)
    predictions = volmod.predict(volX_)

    print(mean_squared_error(voly, predictions))
    coef = np.hstack(([volmod.intercept_], volmod.coef_[1:]))
    return coef

def time_multivar_quintic_reg():
    h1_sqrt = np.sqrt(h1_data)
    h2_sqrt = np.sqrt(h2_data)
    timeX = np.array([h1_sqrt, h2_sqrt]).T
    timey = np.array(time_data)
    time_poly_model = PolynomialFeatures(degree = 5)
    timeX_ = time_poly_model.fit_transform(timeX)

    timemod = linear_model.LinearRegression()
    timemod.fit(timeX_, timey)
    predictions = timemod.predict(timeX_)

    print(mean_squared_error(timey, predictions))
    coef = np.hstack(([timemod.intercept_], timemod.coef_[1:]))
    return coef
```

Code 3: Cubic and quintic regression code of volume and time

```python
def plot(func, zmax, zlabel, mode = "volume"):
    h1 = np.linspace(0, 180, 200)  # Start heights
    h2 = np.linspace(0, 180, 200)  # End heights
    h1, h2 = np.meshgrid(h1, h2)  # Create a 2D grid
    fig = plt.figure(figsize=(8, 6))
    ax = fig.add_subplot(111, projection='3d')
    if mode == "time":
        ax.plot_surface(h1, h2, func((np.sqrt(h1), np.sqrt(h2))), cmap='viridis', alpha=0.7)
    else: ax.plot_surface(h1, h2, func((h1, h2)), cmap='viridis', alpha=0.7)
    ax.set_xlim(0, 180)  # h1 range
    ax.set_ylim(0, 180)  # h2 range
    ax.set_zlim(0, zmax)  # Volume range, 4500, Time range, 150
    ax.set_xlabel('h1')
    ax.set_ylabel('h2')
    ax.set_zlabel(zlabel) #'Volume', 'Drainage Time'
    plt.show()

def __main__():
    volmod_coef = vol_multivar_cubic_reg()
    timemod_coef = time_multivar_quintic_reg()

    print(volmod_coef)
    print(timemod_coef)

    volume_model = create_volume_model(volmod_coef)
    time_model = create_time_model(timemod_coef)

    plot(volume_pred, 4500, "Volume")
    plot(volume_model, 4500, "Volume")

    plot(time_pred, 150, "Drainage Time", "time")
    plot(time_model, 150, "Drainage Time", "time")

__main__()
```

Code 4: Plotting of equations and main function