Faculty of Science

Department of Physics

SIF2029 - Applied Physics

LABORATORY REPORT 3

S3. Variation of Viscosity of Liquids with Temperature

Student Name	**Student ID**
Errol Tay	23054789

Lecturer in charge:

Dr. Izlina Supa'at

Asst. Dr. Azzuliani Supangat

November 12, 2024

Abstract

The first aim of this experiment is to determine the operations of *Redwood Viscometer*. This is done by verifying the *Poiseuille* equation and also the empirical equation for viscosity of Glycerol solution of various concentrations. The data for Glycerol solution is already given by lab manual. Second, this experiment also aims to determine the relationship between temperature and viscosity of different oils. This is done by plotting data obtained from experiment using another empirical formula for viscosity, used by *"American Society for Testing of Materials"* (ASMT). Overall, this experiment emphasises the importance of viscosity, a concept used across various fields, including oil and gas, food production, pharmaceuticals, and manufacturing, where it affects efficiency, functionality, and product quality.

Contents

List of Tables

List of Figures

1 Introduction & Objectives

1.1 Introduction

Since molecules can slide around each other, a liquid has the ability to flow. The resistance to such flow is called the viscosity. Fluids that flow slowly like honey have high viscosity. [1]. Liquid viscosity os important many engineering applications such as the petroleum industry since concepts of fluid mixing: fluid flow, heat- and momentum-transfer operations take into account viscosity [2].

Viscosity is typically governed by the strength of intermolecular forces and especially by the shapes of the molecules of a liquid. Polar molecules or those that can form hydrogen bonds usually have higher viscosity than similar nonpolar substances [4]. However, temperature can also have an effect on the value of viscosity due to the change in density ρ of liquid as it is heated. Kinematic viscosity v is used in the study of change in viscosity of liquid with temperature and its relation to absolute viscosity η is [3]:

$$v = \frac{\eta}{\rho} \tag{1.1}$$

In this experiment, the *Redwood No.1 Viscometer* is used to measure viscosity of lubricating oil at various temperatures. The following oils will be used:

1. Hydraulic Oil (AWS 68)
2. Motor Oil (SAE 10W40 4T API SN JASA MA2 4T)

1.1.1 Characteristics of Redwood Viscometer

It follows from the theory of fluid flow through capillary tube and *Poiseuille* equation that [3]:

$$V = \frac{\pi r^2 P}{8\eta L} \tag{1.2}$$

$$\begin{aligned} \text{where } V &= \text{volume of liquid flowing per second} \\ r &= \text{radius of capillary tube} \\ P &= \text{pressure} \\ L &= \text{length of capillary tube} \end{aligned}$$

From Equation 1.2 and using $P = \rho g h$, we obtain:

$$V = \frac{\pi r^4 hg}{8L}\left(\frac{1}{v}\right) \tag{1.3}$$

$$\begin{aligned} \text{where } h &= \text{height of liquid column} \\ g &= \text{gravitational acceleration} = 9.81 ms^{-1} \end{aligned}$$

Therefore, average velocity of liquid flow in capillary tube is:

$$u_m = \frac{V}{\text{Cross-sectional Area of Tube}} = \frac{r^2 hg}{8L}\left(\frac{1}{v}\right). \tag{1.4}$$

Since time taken for oil to flow in capillary tube is inversely propertional to velocity, the **kinematic viscosity** is:

$$v = At \tag{1.5}$$

Since *Poiseuille* equation involves some approximation, an empirical equation is used to model liquid viscosity:

$$v = At - \frac{B}{t} \tag{1.6}$$

$$\text{where } A = \text{viscometer constant}$$
$$B = \text{apparatus constant}$$

Rewriting Equation 1.6, we obtain:

$$\frac{v}{t} = A - \frac{B}{t^2} \tag{1.7}$$

Below are the data collected for measurement of ρ against t for Glycerol at $T = 30°C$. These results will be used for calculations in Experiment 1.

Table 1.1: Provided values of viscosity of aqueous Glycerol solutions at $30°C$

Glycerol concentration (% weight)	Initial Temperature (°C)	Final Temperature (°C)	Time taken for 50 ml aqueous glycerol to flow (s)	Density ρ ($g \cdot cm^{-3}$)	Kinematic viscosity $\nu \times 10^{-2}$ ($cm^2 s^{-1}$)
80	30.2	30.0	108.0	1.205	33.9
85	30.1	30.0	174.0	1.222	58.0
90	30.2	29.8	307.7	1.223	109.0
95	30.2	29.8	681.2	1.253	237.0
98-100	30.4	32.0	1409.0	1.255	500.0

1.1.2 Viscosity Changes with Temperature

Since oil is not a pure liquid, but a mixture of liquids (complex fluids), changes of viscosity due to heat is difficult to describe. A commonly used empirical law to describe viscosity is [3]:

$$\log\{\log(v + 0.8)\} = n \log(T) + C \tag{1.8}$$

$$\text{where } n = \text{constant}$$
$$C = \text{constant}$$
$$T = \text{absolute temperature}$$

1.2 Objectives

i. Analyse and comment on readings in Table with reference to Equation 1.5 and Equation 1.7.

ii. Determine constants A and B for Redwood viscometer from Equation 1.7.

iii. Determine the constants n and C from Equation 1.8.

iv. Determine the validity of empirical formula to model viscosity of various oils at different temperatures, T.

2 Materials and Methods

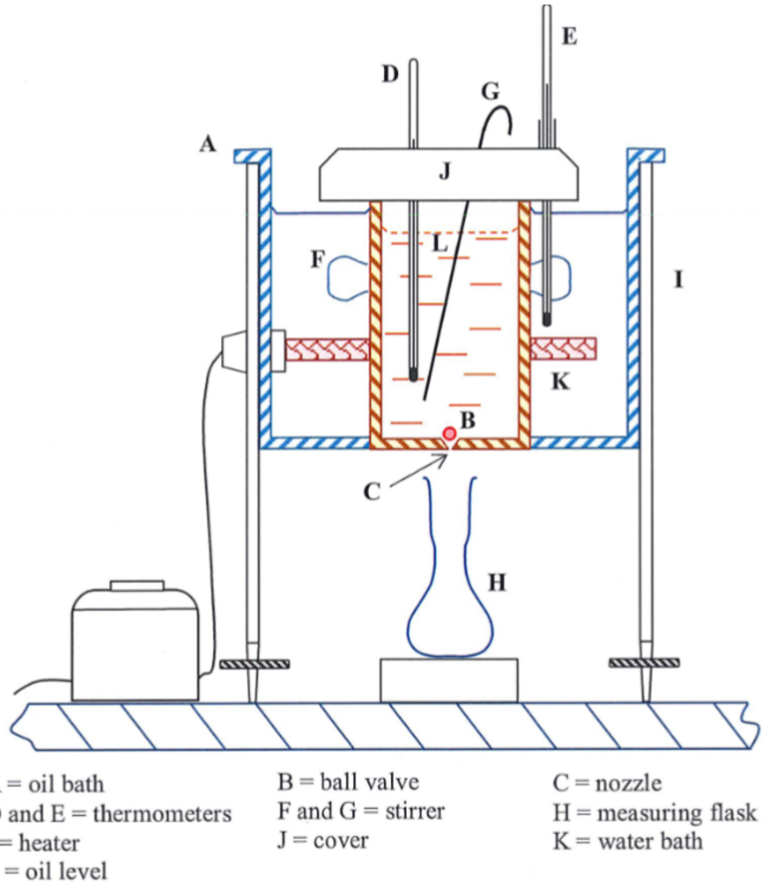


Figure 2.1: Experimental set-up of *Redwood No.1* viscometer

2.1 Materials

i. Redwood viscometer

ii. Hydraulic (Oil AWS 68)

iii. Motor Oil (SAE 10W40 4T API SN JASA MA2 4T)

iv. 10V Power source

v. Stopwatch

vi. Tweezer

vii. Retort stand and clamp

viii. Dishwashing liquid

ix. Tissue paper

2.2 Methods and Procedure

2.2.1 Precautionary Step

1. Made sure the same person operated both stopwatch and ball valve to prevent delay.
2. Made sure oil level L was constant for every single time experiment was carried out.
3. Used new oil when the experiment was repeated.
4. Aligned the thermometer placed in oil such that it is roughly at the center and not touching the walls of the viscometer.
5. Made sure voltage is not set too high to allow precise control of water temperature.
6. Used tweezers to handle B (ball valve) because of the hot handle.

2.2.2 Procedure

Experiment 1

1. Using Equation 1.5 and Equation 1.7, analyse and comment on the data from Table 1.1.
2. Subsequently, using data from Table 1.1, plot graphs to determine constants A and B for Redwood viscometer.

Experiment 2

1. Filled the water bath of viscometer with tap water.
2. Marked the 50 $m\ell$ point of beaker and placed it beneath the viscometer. Then, covered the hole of the viscometer using ball valve.
3. Poured 100 $m\ell$ of motor oil into viscometer and placed two thermometers into the water bath and oil respectively. For thermometer placed in oil, used retort stand to hold it.
4. Switched on the power supply and set it low. Allow for steady increase in water temperature.
5. Heat the water until 30°C. Right as the water reaches, 25°C, switched off the power supply as the temperature of water will keep increasing due to the heat capacity of viscometer. Then, wait to allow for oil and water to reach equilibrium. Carefully adjusted the voltage as necessary until temperature reached desired amount,
6. Once desired temperature is reached, quickly recorded initial temperature T of oil and water. Then, simultaneously removed ball valve and started stopwatch. Stop the stopwatch once oil level in breaker reaches 50 $m\ell$. Then quickly recorded final temperature of oil and water.
7. Cover the hole again using the ball valve and pour the oil back into the viscometer. Then wash the beaker with the diswashing soap.
8. Repeat steps 6-8 for temperatures 40°C, 50°C, 60°C, 70°C, 80°C.
9. Repeat steps 3-9 with the same oil for second reading.
10. Repeat steps 1-10 with hydraulic oil.
11. Plotted a graph using Equation 1.8.

3 Data Collection & Observations

3.1 Data of Characteristics of Redwood Viscometer

Table 3.1: Selected data of viscosity of aqueous Glycerol solutions at 30°C from Table 1.1

Glycerol Concentration (% weight)	Time taken for aqueous glycerol to flow, $t(s)$	Kinematic viscosity, $v \times 10^{-2}$ $(cm^2 s^{-1})$	$\frac{1}{t^2} \times 10^{-5}$	$\frac{v}{t} \times 10^{-2}$
80	108.0	33.9	8.573	0.3139
85	174.0	58.0	3.303	0.3333
90	307.7	109.0	1.056	0.3542
95	681.2	237.0	0.2155	0.3479
98 − 100	1409.0	500.0	0.05037	0.3549

3.2 Data Obtained for Motor Oil

Table 3.2: Data of initial T_i and final temperature T_f for both water and oil, alongside average temperature of oil, T_{avg} and time taken for 50 $m\ell$ of oil to flow, t

Reading 1						
Ideal Temperature (°C)	Water (±0.5°C)		Oil (±0.5°C)		Average Temperature of Oil, T_1	Time, t_1 (±0.1s)
	T_i	T_f	T_i	T_f		
28	27.0	27.0	28.0	28.0	28.00	698.4
40	43.0	41.0	38.0	43.0	40.50	370.8
50	50.5	53.0	47.5	53.0	50.25	266.0
60	60.0	59.5	59.0	60.5	59.75	172.9
70	69.5	69.0	69.5	70.5	70.00	126.3
80	79.0	79.0	79.0	80.0	79.50	100.8
Reading 2						
Ideal Temperature (°C)	Water (±0.5°C)		Oil (±0.5°C)		Average Temperature of Oil, T_2	Time, t_2 (±0.1s)
	T_i	T_f	T_i	T_f		
28	28.0	28.0	28.0	28.0	28.00	664.5
40	45.0	50.0	42.0	46.0	44.00	359.0
50	56.0	58.0	50.0	55.5	52.75	259.0
60	66.0	63.5	60.0	61.0	60.50	202.0
70	70.0	77.0	70.0	76.5	73.25	142.5
80	79.0	79.0	79.0	80.0	79.50	100.2

Table 3.3: Data of average values of temperature, T_{avg} and and time, t_{avg} for both Readings 1 and Reading 2, alongside the values of $\log(T_{avg})$, kinematic viscosity, v and $\log\{\log(v+0.8)\}$

Temperature (K)			Time (s)			$\log(T_{avg})$	Kinetic viscosity, v (cm^2s^{-1})	$\log\{\log(v+0.8)\}$
T_1	T_2	T_{avg}	t_1	t_2	t_{avg}			
301.00	301.00	301.00	698.4	664.5	681.45	2.479	2.400	−0.2965
313.50	317.00	315.25	370.8	359.0	364.90	2.499	1.276	−0.4987
323.25	325.75	324.50	266.0	259.0	262.50	2.512	0.9088	−0.6332
332.75	333.50	333.13	172.9	202.0	187.45	2.523	0.6361	−0.8036
343.00	346.25	344.63	126.3	142.5	134.40	2.537	0.4384	−1.0322
352.50	352.50	352.50	100.8	100.2	100.50	2.547	0.3063	−1.3577

3.3 Data Obtained for Hydraulic Oil

Table 3.4: Data of initial T_i and final temperature T_f for both water and oil, alongside average temperature of oil, T_{avg} and time taken for 50 $m\ell$ of oil to flow, t

Reading 1						
Ideal Temperature (°C)	Water (±0.5°C)		Oil (±0.5°C)		Average Temperature of Oil, T_1	Time, t_1 (±0.1s)
	T_i	T_f	T_i	T_f		
28	28.0	30.0	30.0	30.0	30.00	496.8
40	42.0	42.0	39.0	44.0	41.50	293.5
50	50.5	50.5	49.5	52.0	50.75	194.2
60	64.0	65.0	58.0	67.0	62.50	140.9
70	71.5	71.5	68.0	73.0	70.50	111.6
80	78.0	76.5	79.5	79.0	79.25	81.4
Reading 2						
Ideal Temperature (°C)	Water (±0.5°C)		Oil (±0.5°C)		Average Temperature of Oil, T_2	Time, t_2 (±0.1s)
	T_i	T_f	T_i	T_f		
28	30.0	30.0	30.0	30.0	30.00	489.5
40	42.0	41.0	41.5	42.0	41.75	275.8
50	50.0	48.5	49.0	50.0	49.50	213.6
60	65.0	64.0	62.0	63.5	62.75	143.4
70	71.0	71.5	68.5	71.0	69.75	112.6
80	79.0	80.0	78.0	81.0	79.50	81.7

Table 3.5: Data of average values of temperature, T_{avg} and and time, t_{avg} for both Readings 1 and Reading 2, alongside the values of $\log(T_{avg})$, kinematic viscosity, v and $\log\{\log(v+0.8)\}$

Temperature (K)			Time (s)			$\log(T_{avg})$	Kinetic viscosity, v (cm^2s^{-1})	$\log\{\log(v+0.8)\}$
T_1	T_2	T_{avg}	t_1	t_2	t_{avg}			
303.00	303.00	303.00	496.8	489.25	493.025	2.481	1.7320	−0.3942
314.50	314.75	314.63	293.5	275.8	284.65	2.498	0.9885	−0.5978
323.75	322.50	323.13	194.2	213.6	203.9	2.509	0.6964	−0.7569
335.50	335.75	335.63	140.9	143.4	142.15	2.526	0.4678	−0.9870
343.50	342.75	343.13	111.6	112.6	112.1	2.535	0.3523	−1.2105
352.25	352.50	352.38	81.4	81.7	81.55	2.547	0.2280	−1.9205

*Values of viscosity, v in Table 3.3 and Table 3.5, calculated using values A_2 and B from Section 4.1.2. Assumed A and B do not change as same viscometer is used throughout the experiment.

3.4 Graphs of Viscosity, v against Temperature, T

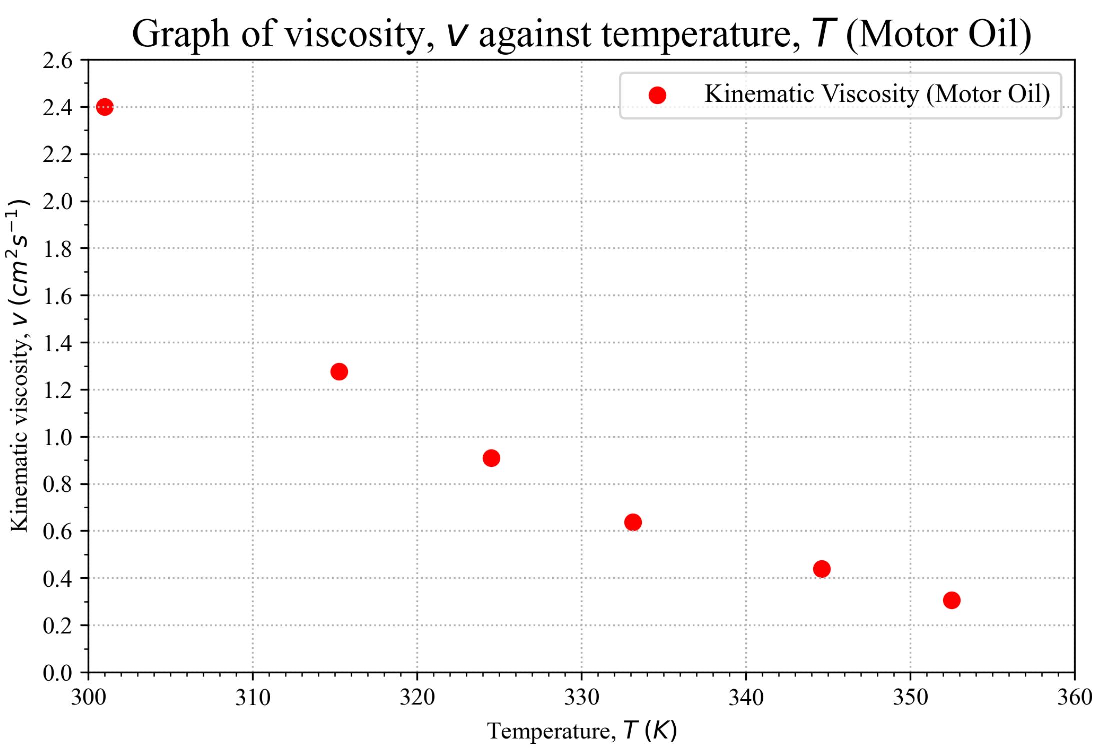


Figure 3.1: Graph of kinematic viscosity, v against temperature, T (Motor Oil)

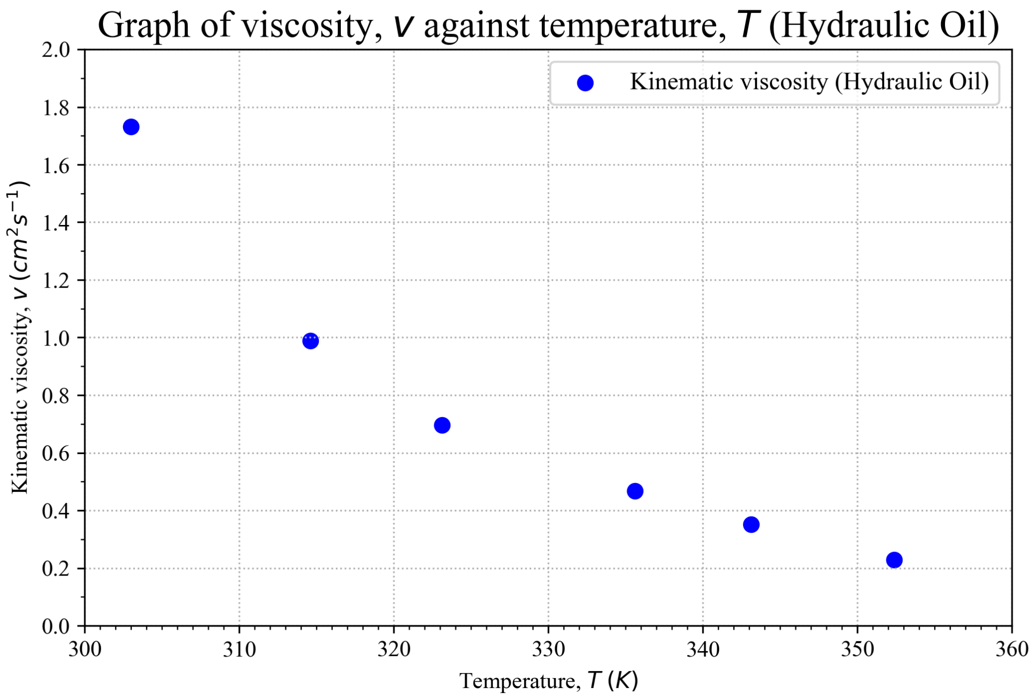


Figure 3.2: Graph of kinematic viscosity, v against temperature, T (Hydraulic Oil)

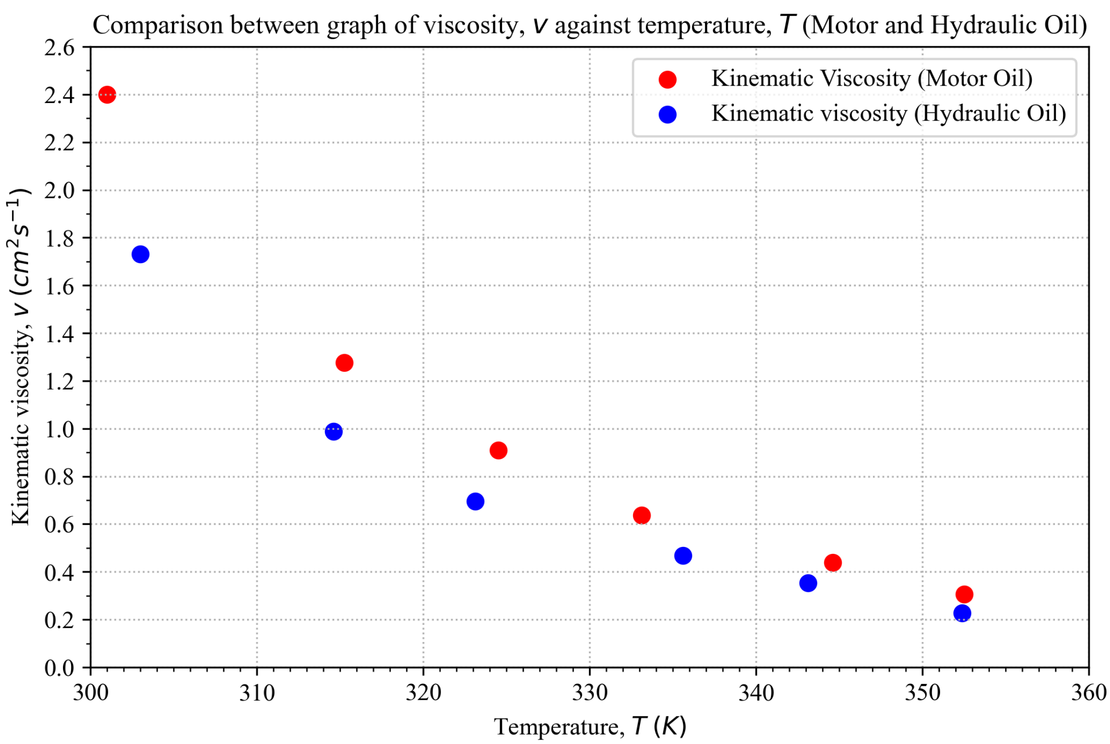


Figure 3.3: Comparison between graphs of viscosity, v against temperature, T (Motor and Hydraulic Oil)

4 Results & Analysis

4.1 Characteristics of Redwood Viscometer

4.1.1 Determination of A

With reference to Equation 1.5, used least-square method to obtain best-fit gradient of viscosity, v against time for 50 $m\ell$ of oil to flow, t. Set $x = t$ while $y = v$. Data was selected from Table 3.1 which are given values from the lab manual.

Table 4.1: Data of x, y, x^2, y^2 and xy alongside their summations for Glycerol

x	y	x^2	y^2	xy
108.0	0.339	11664	0.1149	36.61
174.0	0.580	30276	0.3364	100.92
307.7	1.090	94679	1.1881	335.39
681.2	2.370	464033	5.6169	1614.44
1409.0	5.000	1985281	25.0000	7045.00
$\sum x = 2679.9$	$\sum y = 9.379$	$\sum x^2 = 2585933$	$\sum y^2 = 32.2563$	$\sum xy = 9132.36$

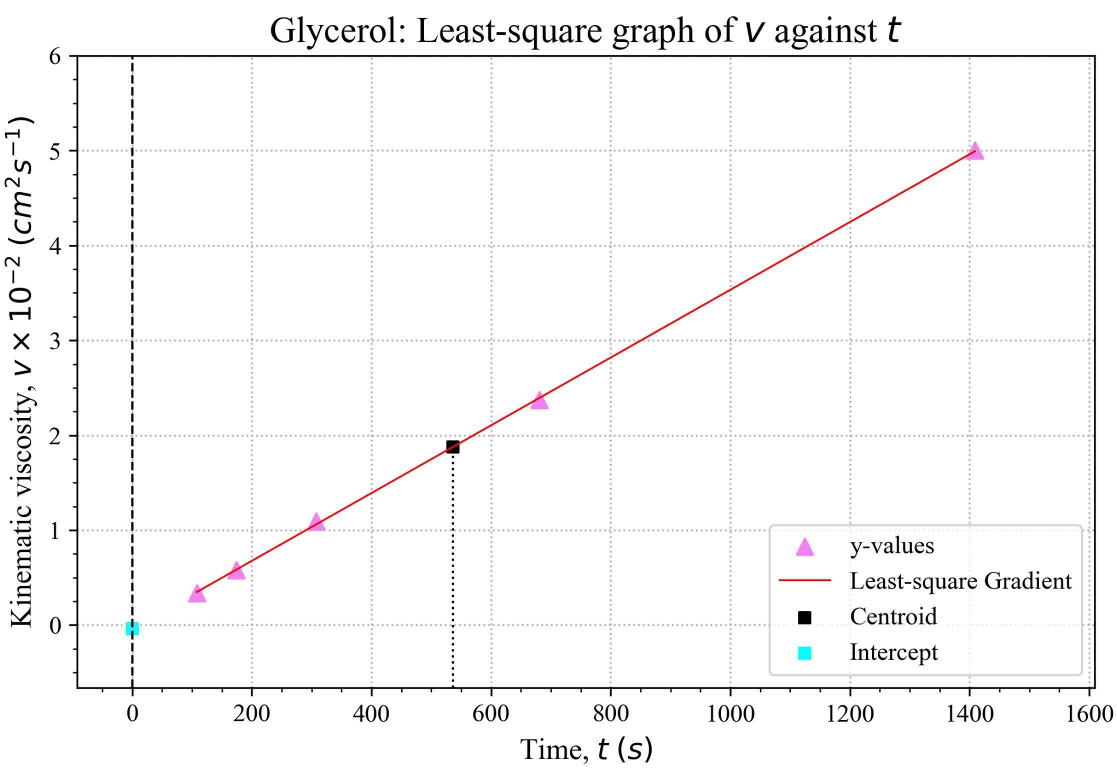


Figure 4.1: Least-squares graph of v against t for Glycerol

Gradient was calculated to be:

$$m_1 = (3.5713 \pm 0.021) \times 10^{-3} = (3.58 \pm 0.02) \times 10^{-3} cm^2 s^{-2}$$

Percentage uncertainty of gradient:

$$\gamma_{m_1} = \frac{\Delta m_1}{m_1} \times 100\% = 0.06\%$$

Correlation coefficient was calculated to be:

$$r_1 = 0.999$$

Coefficient of determination was calculated to be:

$${R_1}^2 = 0.999$$

From Equation 1.5, constant A_1 was determined to be:

$$A_1 \equiv m_1 = (3.58 \pm 0.02) \times 10^{-3} cm^2 s^{-2}$$

Percentage uncertainty of constant A_1:

$$\gamma_{A_1} = \frac{\Delta A_1}{A_1} \times 100\% = 0.06\%$$

4.1.2 Determination of B

With reference to Equation 1.7, used least-square method to obtain best-fit gradient of viscosity over time $\frac{v}{t}$ against one over time squared $\frac{1}{t^2}$. Set $x = \frac{1}{t^2}$ while $y = \frac{v}{t}$. Data was selected from Table 3.1 which was processed from the given values from the lab manual.

Table 4.2: Data of x, y, x^2, y^2 and xy alongside their summations for Glycerol

$x \times 10^{-5}$	$y \times 10^{-2}$	$x^2 \times 10^{-10}$	$y^2 \times 10^{-4}$	$xy \times 10^{-7}$
8.573	0.3139	73.503	0.09853	2.691
3.303	0.3333	10.909	0.1111	1.101
1.056	0.3542	1.116	0.1255	0.3741
0.2155	0.3479	0.04644	0.1210	0.07498
0.05037	0.3549	0.002537	0.1259	0.01787
$\sum x = 13.19787$	$\sum y = 1.7042$	$\sum x^2 = 85.576977$	$\sum y^2 = 0.58203$	$\sum xy = 4.25895$

Gradient was calculated to be:

$$m_2 = (-4.72 \pm 0.71) = (-4.8 \pm 0.7)\, cm^2$$

Percentage uncertainty of gradient:

$$\gamma_{m_2} = \frac{\Delta m_2}{m_2} \times 100\% = 14.29\%$$

Intercept of y-axis was determined to be:

$$b_2 = 3.533 \times 10^{-3} cm^2 s^{-2}$$

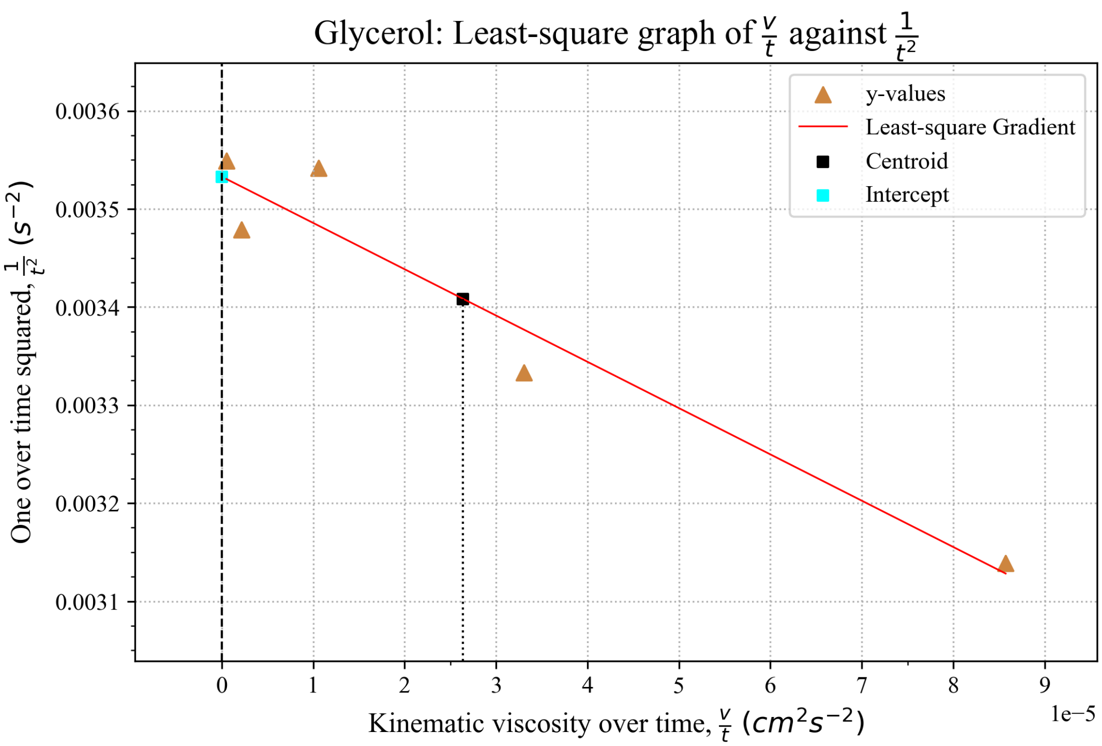


Figure 4.2: Least-squares graph of $\frac{v}{t}$ against $\frac{1}{t^2}$ for Glycerol

Least squares equation was determined to be:

$$y = -4.9x + 3.533 \times 10^{-3}$$

Correlation coefficient was calculated to be:

$$r_2 = -0.9673$$

Coefficient of determination was calculated to be:

$${R_2}^2 = 0.9356$$

With reference to Equation 1.7, values of A and B can be determined from least squares equation.

Constant A_2 was determined to be:

$$A_2 \equiv b_2 = 3.533 \times 10^{-3} cm^2 s^{-2}$$

Comparing the difference in values of A_1 from the previous part:

$$dA = \frac{A_1 - A_2}{A_1} = 4.70\%$$

Constant B was determined to be:

$$B \equiv m_2 = -\left(4.9 \pm 0.7\right) cm^2$$

Percentage uncertainty of constant B:

$$\gamma_B = \frac{\Delta B}{B} \times 100\% = 14.29\%$$

4.2 Motor Oil

With reference to Equation 1.8, used least-square method to obtain best-fit gradient of $\log\{\log(v+0.8)\}$ against $\log(T)$. Set $x = \log(T)$ and $y = \log\{\log(v+0.8)\}$. Data was selected from Table 3.3.

Table 4.3: Data of x, y, x^2, y^2 and xy alongside their summations for motor oil

x	y	x^2	y^2	xy
2.479	−0.2965	6.1454	0.0879	−0.7351
2.499	−0.4987	6.2450	0.2487	−1.2462
2.512	−0.6332	6.3101	0.4010	−1.5907
2.523	−0.8036	6.3655	0.6457	−2.0274
2.537	−1.0322	6.4364	1.0654	−2.6186
2.547	−1.3577	6.4872	1.8434	−3.4581
$\sum x = 15.097$	$\sum y = -4.6219$	$\sum x^2 = 37.9896$	$\sum y^2 = 4.2921$	$\sum xy = -11.6761$

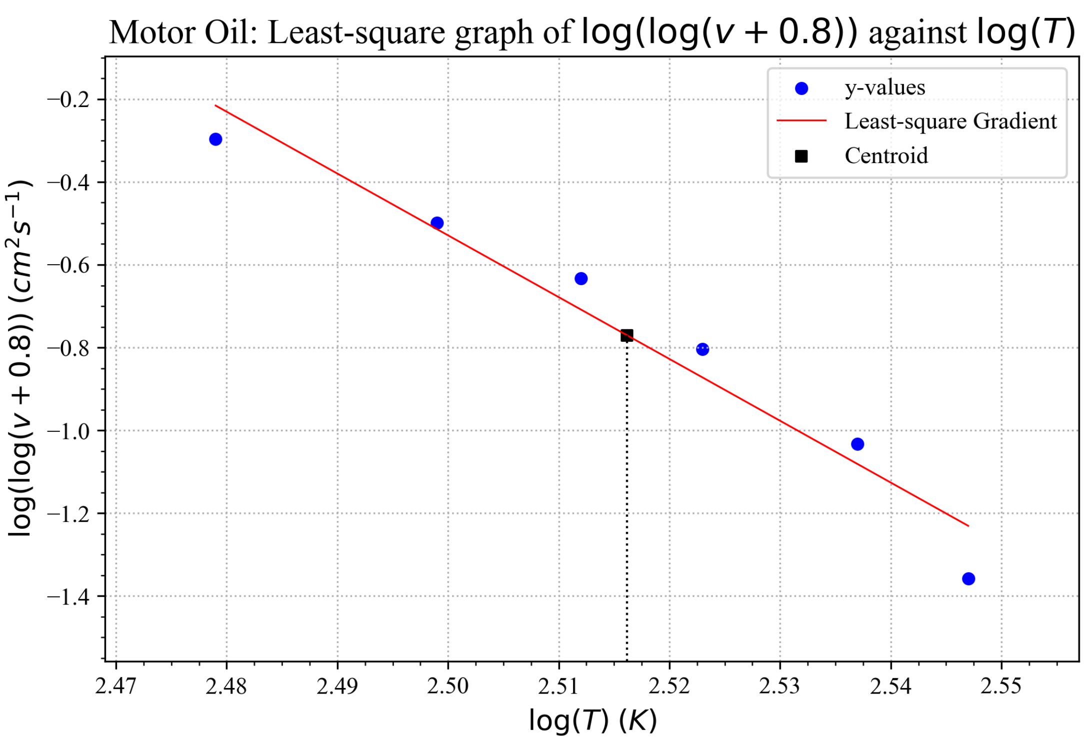


Figure 4.3: Least-squares graph of $\log\{\log(v+0.8)\}$ against $\log(T_{avg})$ for motor oil

Gradient was calculated to be:

$$m_3 = -(14.93 \pm 1.7) = -(14 \pm 2)\, cm^2 s^{-1} K^{-1}$$

Percentage uncertainty of gradient:

$$\gamma_{m_3} = \frac{\Delta m_3}{m_3} \times 100\% = 14.28\%$$

Intercept of y-axis was determined to be:

$$b_3 = 36.78\ cm^2 s^{-1}$$

Least squares equation was determined to be:

$$y = -14x + 36.78$$

Correlation coefficient was calculated to be:

$$r_3 = -0.9753$$

Coefficient of determination was calculated to be:

$$R_3{}^2 = 0.9512$$

With reference to Equation 1.8, values of n and C can be determined from least squares equation.

Constant n_3 is determined to be:

$$n_3 \equiv m_3 = -\left(14 \pm 2\right) cm^2 s^{-1} K^{-1}$$

Percentage uncertainty of constant n_3:

$$\gamma_{n_3} = \frac{\Delta n_3}{n_3} \times 100\% = 14.28\%$$

Constant C_3 was determined to be:

$$C_3 \equiv b_3 = 36.78\ cm^2 s^{-1}$$

4.3 Hydraulic Oil

With reference to Equation 1.8, used least-square method to obtain best-fit gradient of $\log\left\{\log\left(v + 0.8\right)\right\}$ against $\log\left(T\right)$. Set $x = \log\left(T\right)$ and $y = \log\left\{\log\left(v + 0.8\right)\right\}$. Data was selected from Table 3.4.

Table 4.4: Data of x, y, x^2, y^2 and xy alongside their summations for motor oil

x	y	x^2	y^2	xy
2.481	−0.3942	6.1554	0.1554	−0.9780
2.498	−0.5978	6.2400	0.3573	−1.4932
2.509	−0.7569	6.2951	0.5728	−1.8990
2.526	−0.9870	6.3807	0.9742	−2.4932
2.535	−1.2105	6.4262	1.4654	−3.0687
2.547	−1.9205	6.4872	3.6882	−4.8915
$\sum x = 15.096$	$\sum y = -5.8669$	$\sum x^2 = 37.9846$	$\sum y^2 = 7.2134$	$\sum xy = -14.8235$

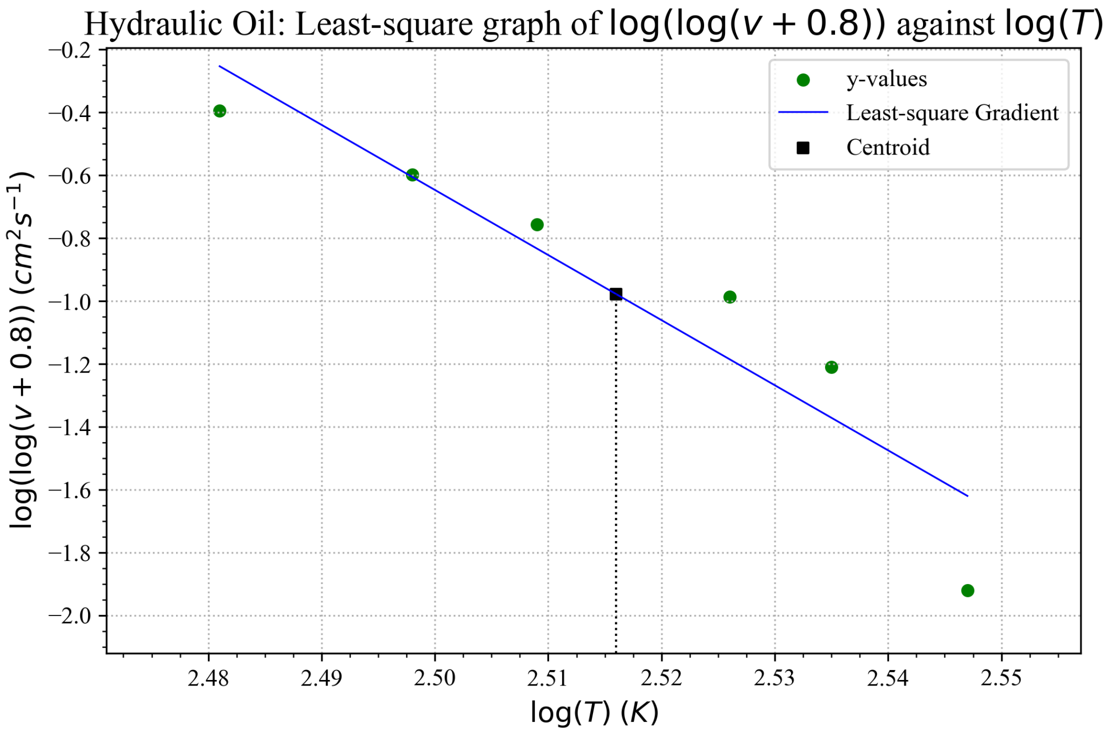


Figure 4.4: Least-squares graph of $\log\{\log(v+0.8)\}$ against $\log(T_{avg})$ for hydraulic oil

Gradient was calculated to be:

$$m_4 = -(20.71 \pm 3.8) = -(20 \pm 4)\, cm^2 s^{-1} K^{-1}$$

Percentage uncertainty of gradient:

$$\gamma_{m_4} = \frac{\Delta m_4}{m_4} \times 100\% = 20.00\%$$

Intercept of y-axis was determined to be:

$$b_4 = 51.13\ cm^2 s^{-1}$$

Least squares equation was determined to be:

$$y = -20x + 51.13$$

Correlation coefficient was calculated to be:

$$r_4 = -0.9366$$

Coefficient of determination was calculated to be:

$$R_4{}^2 = 0.8773$$

With reference to Equation 1.8, values of n and C can be determined from least squares equation.

Constant n_4 is determined to be:

$$n_4 \equiv m_4 = -(20 \pm 4)\, cm^2 s^{-1} K^{-1}$$

Percentage uncertainty of constant n_4:

$$\gamma_{n_4} = \frac{\Delta n_4}{n_4} \times 100\% = 20.00\%$$

Constant C_4 was determined to be:

$$C_4 \equiv b_4 = 51.13\ cm^2 s^{-1}$$

4.4 Comparison between Motor and Hydraulic Oil

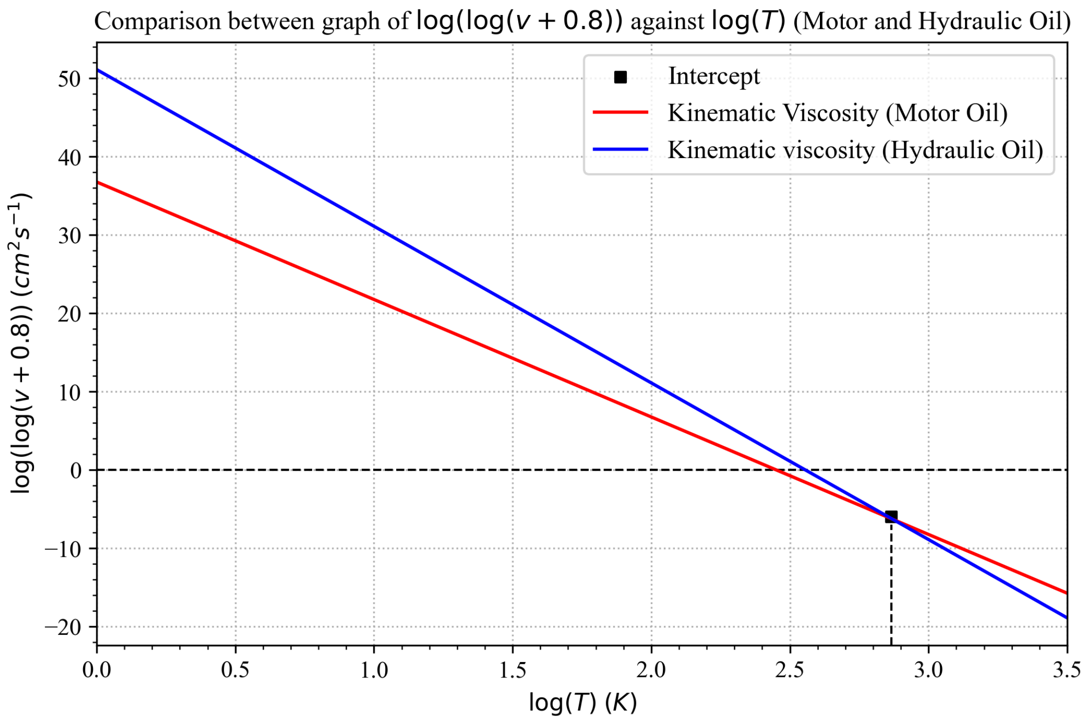


Figure 4.5: Comparison between graphs of $\log(\log(v + 0.8))$ against $\log(T)$ (Motor and Hydraulic Oil)

Analytically, intersection of the two graphs occur at $\log(T) \approx 2.865K$. Converting into absolute temperature, $T = 732.82K$. It is important to note that this graph, relates the kinematic viscosity, v inversely with temperature, T due to the double log. This means, comparing $P = \log(\log(v + 0.8)) = -0.2$ and $G = \log(\log(v + 0.8)) = -0.3$, when we convert back into viscosity, v, viscosity P will be lower compared to viscosity Q.

4.5 Analysis and Interpretation

Referring to Section 3.4, graphs of kinematic viscosity, v against temperature, T was plotted. For both types of oils, data points show that the graphs follow a negative exponential relationship, $f(x) = e^{-x}$. Moreover, when plotting the two data points together, it is noticed that the kinematic viscosity, v for motor oil seems to decrease at a faster rate than hydraulic oil at increasing temperature, T.

Referring to Section 4.1.1 and 4.1.2, from both least-squares graph, we found values of constant A that were nearly identical with a percentage difference of only $dA = 4.70\%$. This means that the determined value of A is dependable and can be used for further calculation such as in Section 4.2 and Section 4.3. Meanwhile, for Section 4.1.2 we managed to plot a linear regression graph with a high correlation coefficient, indicating a good correlation between $\frac{v}{t}$ and $\frac{1}{t^2}$ of the *Poiseuille* equation.

Meanwhile, referring to Sections 4.3 and Section 4.4, we find that n_4 is lower than n_3. From the graph in Section 4.4, we see that the magnitude of gradient of motor oil is lower compared to hydraulic oil, and they intercept at $\log(T) \approx 2.865K \Rightarrow T = 732.82K$. Referring back to the discussion in Section 4.4, this means that viscosity, v of motor oil decreases faster with increasing temperature compared to hydraulic oil.

Additionally, obtained high values of correlation coefficient ~ 0.95 for both motor oil and hydraulic oil. This indicates a good correlation between x and y values used in Equation 1.8.

5 Error Analysis

5.1 Precautionary Steps

First, made sure the same person operated both stopwatch and ball valve to prevent delay. Second, made sure oil level L was constant for every single time experiment was carried out. Third, used new oil when the experiment was repeated. Fourth, aligned the thermometer placed in oil such that it is roughly at the center and not touching the walls of the viscometer. Fifth, made sure voltage is not set too high to allow precise control of water temperature. Sixth, made sure temperature of oil is stabalised at all times so that measurement of average temperature is consistent.

5.2 Sources of Error

The first source of error is the excess heat from heating water bath. The metals/heater of the water bath can contain excess heat after the power is turned off. This will cause the temperature of the system to further increase even though power supply is off. Second, the time taken for heat transfer between water and oil. Sometimes, water temperature may be $75°C$ while oil is only $67°C$. But towards the end, it may end up as $73°C$ and $74°C$ respectively. This causes average temperature of oil to fluctuate a lot. Seven, made sure to fill in water in the water bath, not oil. Oil has about half the heat capacity of water [5]. Therefore, by using water, ensured even heating of oil and also due to high heat capacity of water, change in temperature does not rise too quickly from heating.

5.3 Steps to Overcome Weakness

First, made sure to not set the voltage of power supply too high. This ensured that not too much heat is leftover in the heating element which causes oil temperature to rise much further beyond intended temperature. Second, after heating the oil up to $80°C$ in the first Reading, used new oil for Reading 2 as the chemical structure of the oil molecules may change due to heating, resulting in different viscosity. Third, carried out heating slowly and at lowe voltages so that there is enough time for heat in the water bath to transfer into oil, resulting in stable temperatures.

5.4 Limitations of Experiment

The first limitation is the of maximum temperature of oil. Since water has a boiling point of $100°C$, the maximum temperature of oil is limited to $100°C$ as well. Another limitation is the heat loss to surroundings. During the experiment, there was no shielding to cover the surface of water bath and oil, resulting in heat loss through evaporation and conduction. Thus, the highiest temperature able to be reached in this experiment is about $90°C$.

5.5 Suggestions to Improve Experiment

First, replace the water bath of this experiment with a proper laboratory water bath as in Figure 5.1. A proper water bath allows the user to set a fixed temperature to maintain the water at, rather than turning a voltage knob and hoping for the best. Second, include a cover for the set-up so that heat loss to the surroundings can be reduced. This allows for a higher maximum temperature of oil and also stabalises the temperature of the oil.

Figure 5.1: Weight holder at its max capacity

6 Discussion & Interpretation

6.1 Characteristics of Redwood Viscometer

6.1.1 Determination of A

From plotting least squares graph of v against t, the value of constant A_1 was found to be $A_1 = (3.58 \pm 0.02) \times 10^{-3} cm^2 s^{-2}$ with percentage uncertainty of $\gamma_{A_1} = \frac{\Delta A_1}{A_1} \times 100\% = 0.06\%$ and correlation of determination ${R_1}^2 = 0.999$ indicating a very accurate value of A. Furthermore, a near-1 value of correlation coefficient $r_1 = 0.999$ was obtained, indicating a very good linear correlation between v and t. Thus, Equation 1.5 is valid to model liquid viscosity.

6.1.2 Determination of B

From plotting least squares graph of $\frac{v}{t}$ against $\frac{1}{t^2}$, the value of constant A_2 and B were found. First the value of B was determined to be $B = -(4.9 \pm 0.7)\, cm^2$ with percentage uncertainty of $\gamma_B = \frac{\Delta B}{B} \times 100\% = 14.29\%$ and correlation of determination ${R_2}^2 = 0.9356$ which indicate value of B is acceptable.

Meanwhile, the value of A_2 was found to be $A_2 = 3.533 \times 10^{-3} cm^2 s^{-2}$. Comparing the values of A_2 with the previous section, we obtain a difference of $dA = 4.70\%$. This means that the value of A for both experiments are only off by only $\sim 5\%$, which is within acceptable range. Furthermore, the correlation coefficient was found to be $r_2 = -0.9673$ which is close to -1, meaning a good negative linear correlation between $\frac{v}{t}$ and $\frac{1}{t^2}$. Thus, these two results show us that the empirical Equation 1.6 is also valid to model liquid viscosity.

6.2 Motor Oil

From plotting least squares graph of $\log\{\log(v + 0.8)\}$ against $\log(T)$, the value of constant n was found to be $n_3 = -(14 \pm 2)\, cm^2 s^{-1} K^{-1}$ with percentage uncertainty of $\gamma_{n_3} = \frac{\Delta n_3}{n_3} \times 100\% = 14.28\%$ and correlation of determination ${R_3}^2 = 0.9512$ indicating an accurate value of n_3. Meanwhile, C_3 was found to be $C_3 = 36.78\ cm^2 s^{-1}$. Moreover, correlation coefficient was calculated to be $r_3 = -0.9753$ which is near to value -1. This indicates a good negative linear correlation between $\log\{\log(v + 0.8)\}$ against $\log(T)$. Therefore, the results supports that Equation 1.8 is suitable to be used to model viscosity of a liquid.

6.3 Hydraulic Oil

From plotting least squares graph of $\log\{\log(v + 0.8)\}$ against $\log(T)$, the value of constant n was found to be $n_4 = -(20 \pm 4)\, cm^2 s^{-1} K^{-1}$ with percentage uncertainty of $\gamma_{n_4} = \frac{\Delta n_4}{n_4} \times 100\% = 20.00\%$ and correlation of determination ${R_4}^2 = 0.9366$ indicating an accurate value of n_4. Meanwhile, C_4 was found to be $C_4 = 51.13\ cm^2 s^{-1}$. Moreover, correlation coefficient was calculated to be $r_4 = -0.8773$ which is near to value -1. This indicates a good negative linear correlation between $\log\{\log(v + 0.8)\}$ against $\log(T)$. Therefore, the results again supports that Equation 1.8 is suitable to be used to model viscosity of a liquid.

6.4 Comparison between Motor and Hydraulic Oil

Since we found that n_4 is lower than n_3. From data in the tables, we observed that in general, hydraulic oil takes less time for 50 $m\ell$ to flow compared to motor oil. Therefore, this implies that lower/more negative values of n corresponds to lower viscosity.

6.5 Questions

From the graph in Section 4.4, we see that the magnitude of gradient of motor oil is lower compared to hydraulic oil, and they intercept at $\log(T) \approx 2.865K \Rightarrow T = 732.82K$. Referring back to the discussion in Section 4.4, this means that viscosity, v of motor oil decreases faster with increasing temperature compared to hydraulic oil. At $T = 732.82$, hydraulic oil will have higher viscosity than motor oil.

1. Analyse and comment on the readings in Table 1.1 with reference to Equation 1.5 and 1.7.

   Already discuss in Section 6.1.

2. Determine constants A and B for Redwood viscometer.

   Determined in Section 4.1.

3. Comment on the appropriateness of empirical law (Equation 1.7).

   Referring to Section 6.2 and Section 6.3, determined that correlation coefficient of both oils to be ~ -0.98 which is close the the value -1. Therefore, this indicates a very high negative linear correlation between $\log\{\log(v+08)\}$ with $\log(T)$. Thus, we conclude that the empirical formula describes the observed physical phenomena very well.

4. The size of the error cause by the instability of oil temperature.

   If oil temperature is not stable, then the size of error will increase. This is because if temperature of oil continues to increase dramatically during the collection of 50 $m\ell$ in beaker, the viscosity of the oil will change and value of time t will fluctuate, leading to a large error. This is why it is necessary to measure initial T_i and final T_f temperatures to get an average temperature T_{avg} of oil. Assuming the oil's temperature continues to increase at constant rate during collection of 50 $m\ell$, the average value gives us a good approximation between the two ends of temperature, T_i and T_f.

5. The importance of the factor 0.8.

   Referring to Table 3.3 and Table 3.5, it can be seen that the value of v ranges from $0.2280 - 2.400$. Since we need to perform log on v two times, if we did not add value 0.8, some values of v would be ≤ 1. Therefore, then we would obtain a complex number, $\mathbb{C}$ (which we want to avoid). For example, if $v = 0.5\ cm^2 s^{-1}$:

$$\log(v) = -0.3010$$
$$\Rightarrow \log\{\log(v)\} = log\{-0.3010\} = -0.5214 + 1.364i$$

7 Conclusion

In conclusion, for Redwood viscometer, using Equation 1.5 the value of constant A_1 was found to be $A_1 = (3.58 \pm 0.02) \times 10^{-3} cm^2 s^{-2}$ with percentage uncertainty of $\gamma_{A_1} = \frac{\Delta A_1}{A_1} \times 100\% = 0.06\%$ and correlation of determination ${R_1}^2 = 0.999$ indicating a very accurate value of A. Furthermore, correlation coefficient $r_1 = 0.999$ was obtained, indicating a very good linear correlation between v and t indicating Equation 1.5 is valid to model liquid viscosity. Meanwhile, using Equation 1.7 value of B was determined to be $B = -\left(4.9 \pm 0.7\right) cm^2$ with percentage uncertainty of $\gamma_B = \frac{\Delta B}{B} \times 100\% = 14.28\%$ and correlation of determination ${R_2}^2 = 0.9356$ which indicate value of B is acceptable. In addition, the value of A_2 was found to be $A_2 = 3.533 \times 10^{-3} cm^2 s^{-2}$. Comparing the value of A_1 with the previous section, we obtain a difference of $dA = 4.70\%$ which means both experiments agree and produce similar results for A. Furthermore, the correlation coefficient was found to be $r_2 = -0.9673$ meaning a good negative linear correlation between $\frac{v}{t}$ and $\frac{1}{t^2}$. Thus, these two results show us that the empirical Equation 1.6 is also valid to model liquid viscosity. Next, using Equation 1.7, for motor oil, it was found that $n_3 = -\left(14 \pm 2\right) cm^2 s^{-1} K^{-1}$ with percentage uncertainty of $\gamma_{n_3} = \frac{\Delta n_3}{n_3} \times 100\% = 14.28\%$ and correlation of determination ${R_3}^2 = 0.9512$ indicating an accurate value of n_3. Meanwhile, for hydraulic oil, it was found that $n_4 = -\left(20 \pm 4\right) cm^2 s^{-1} K^{-1}$ with percentage uncertainty of $\gamma_{n_4} = \frac{\Delta n_4}{n_4} \times 100\% = 20.00\%$ and correlation of determination ${R_4}^2 = 0.8773$ indicating an accurate value of n_4. For both oils, value correlation coefficient ~ 0.95 which indicates a good negative linear correlation between $\log\left\{\log\left(v + 0.8\right)\right\}$ against $\log\left(T\right)$. Therefore, the results supports that Equation 1.8 is suitable to be used to model viscosity of a liquid. Furthermore, we found that viscosity, v of motor oil decreases faster with increasing temperature compared to hydraulic oil. At $T = 732.82$, hydraulic oil will have higher viscosity than motor oil.

References

[1] (Britannica, 2024). Viscosity — Definition, Facts, Formula, Units, and Examples. Encyclopedia Britannica. https://www.britannica.com/science/viscosity, 2024.

[2] A. Kayode Coker. *Physical Properties of Liquids and Gases*. Ludwig's Applied Process Design for Chemical and Petrochemical Plants (Fourth Edition, 2007.

[3] University of Malaya Department of Physics. *Applied Physics Practical Lab Manual S3: Variation of Viscosity of Liquids with Temperature*. University of Malaya, 2012.

[4] E. Vitz, J.W. Moore, J. Shorb. Viscosity — Definition, Facts, Formula, Units, and Examples. Encyclopedia Britannica. 10.7:Viscosity.ChemistryLibreTexts.https://chem.libretexts.org/Bookshelves/General_Chemistry/ChemPRIME_(Moore_et_al.)/10%3A_Solids_Liquids_and_Solutions/10.07%3A_Viscosity, 2023, June 26.

[5] Harvard. Specific Heats of Oil and Water. https://sciencedemonstrations.fas.harvard.edu/presentations/specific-heats-oil-and-water, n.d.

A Calculations

A.1 Data of Characteristics of Redwood Viscometer

Calculations will be using data of glycerol concentration (% weight) = 80 provided from Table 1.1.

A.1.1 One over time squared, $\frac{1}{t^2}$

$$\frac{1}{t^2} = \frac{1}{108.0^2} = 8.573 \times 10^{-5} \tag{A.1}$$

A.1.2 Viscosity over time, $\frac{v}{t}$

$$\frac{v}{t} = \frac{33.9 \times 10^{-2}}{108.0} = 0.3139 \times 10^{-2} \tag{A.2}$$

A.2 Data Obtained for Motor Oil

Calculations will be using constants $A = 3.533 \times 10^{-3} cm^2 s^{-1}$ and $B = -4.9 cm^2$ obtained from least-squares plot from Section 4.1.2. Assumption is that, since the viscometer used is the same, therefore values of A and B can be carried forward for other calculations.

A.2.1 Reading 1: Average Temperature of Oil, T_1

Calculations will be using data from Table 3.2, Reading 1: Ideal Temperature ($°C$) = 40.

$$T_1 = \frac{T_i + T_f}{2} = \frac{38.0 + 43.0}{2} = 40.50°C \tag{A.3}$$

A.2.2 Reading 2: Average Temperature of Oil, T_2

Calculations will be using data from Table 3.2, Reading 2: Ideal Temperature ($°C$) = 40.

$$T_2 = \frac{T_i + T_f}{2} = \frac{42.0 + 46.0}{2} = 44.00°C \tag{A.4}$$

A.2.3 Conversion of T_1 and T_2 into Kelvin

Calculations will values of T_1 and T_2 obtained previously.

$$T_1 = 40.50 + 273 = 313.50K, \quad \text{and} \quad T_2 = 44.00 + 273 = 317.00K \tag{A.5}$$

A.2.4 Average Temperature of Oil, T_{avg}

Calculations will be using data from Table 3.3 and $T_1 = 313.50K$, $T_2 = 317.00K$.

$$T_{avg} = \frac{T_1 + T_2}{2} = \frac{313.50 + 317.00}{2} = 315.25K \tag{A.6}$$

A.2.5 Average time taken for $50\ m\ell$ of oil to be collected, t_{avg}

Calculations will be using data from Table 3.3 and $t_1 = 370.8$, $t_2 = 359.0$.

$$t_{avg} = \frac{t_1 + t_2}{2} = \frac{370.8 + 359.0}{2} = 364.90s \tag{A.7}$$

A.2.6 Logarithm of $T_{avg}, \log(T_{avg})$

Calculations will be using the value of T_{avg} obtained previously.

$$\log(T_{avg}) = \log(315.25) \approx 2.499K \tag{A.8}$$

A.2.7 Kinematic viscosity, v

Calculations will be using the empirical formula for viscosity (Equation 1.6)

$$\begin{aligned} v &= At - \frac{B}{t} \\ &= 3.533 \times 10^{-3} \cdot 364.90 - \frac{4.9}{364.90} \\ &\approx 1.276 cm^2 s^{-1} \end{aligned} \tag{A.9}$$

A.2.8 Logarithm of logarithm of viscosity plus 0.8, $\log\{\log(v + 0.8)\}$

Calculations will be using the value of viscosity, v obtained previously.

$$\begin{aligned} \log\{\log(v + 0.8)\} &= \log\{\log(1.276 + 0.8)\} \\ &= \log(0.3172) \\ &\approx -0.4987 \end{aligned} \tag{A.10}$$

A.3 Least-squares Calculation

Calculations of least-squares will be on Section: Determination of B using data from Table 4.2. Also, since all four graphs plotted are least-squares, only method from this plot will be shown as it the same for the rest.

A.3.1 Least-squares gradient, m_2

$$\begin{aligned} m_2 &= \frac{n\sum xy - \sum x \sum y}{n\sum x^2 - \left(\sum x\right)^2} \\ &= \frac{5 \cdot 4.25895 \times 10^{-7} - 13.19787 \times 10^{-5} \cdot 1.7042 \times 10^{-2}}{5 \cdot 85.576977 \times 10^{-10} - (13.19787 \times 10^{-5})^2} \\ &= -4.7212 cm^2 \end{aligned} \tag{A.11}$$

A.3.2 Uncertainty of Least-Squares Gradient, Δm_2

$$\begin{aligned} \Delta m_2 &= \sqrt{\frac{n\sum y^2 - \left(\sum y\right)^2 - \frac{\left(n\sum xy - \sum x \sum y\right)^2}{n\sum x^2 - \left(\sum x\right)^2}}{(n-2)\left(n\sum x^2 - \left(\sum x\right)^2\right)}} \\ &= \sqrt{\frac{5 \cdot 0.58203 \times 10^{-4} - (1.7042 \times 10^{-2})^2 - \frac{(5 \cdot 4.25895 \times 10^{-7} - 13.19787 \times 10^{-5} \cdot 1.7042 \times 10^{-2})^2}{5 \cdot 85.576977 \times 10^{-10} - (13.19787 \times 10^{-5})^2}}{(5-2)\left(5 \cdot 85.576977 \times 10^{-10} - (13.19787 \times 10^{-5})^2\right)}} \\ &= 7.1492 \times 10^{-1} cm^2 \end{aligned} \tag{A.12}$$

A.3.3 Percentage Uncertainty, γ_{m_2}

After rounding up m_2.

$$\gamma_{m_2} = \frac{\Delta m_2}{m_2} \times 100\% = \frac{0.7}{4.9} = 14.28\% \tag{A.13}$$

A.3.4 y-Intercept, b_2

$$\begin{aligned} b_2 &= \frac{\sum x^2 \sum y - \sum x \sum xy}{n \sum x^2 - \left(\sum x\right)^2} \\ &= \frac{85.576977 \times 10^{-10} \cdot 1.7042 \times 10^{-2} - 13.19787 \times 10^{-5} \cdot 4.25895 \times 10^{-7}}{5 \cdot 85.576977 \times 10^{-10} - \left(13.19787 \times 10^{-5}\right)^2} \\ &= 3.533 \times 10^{-3} cm^2 s^{-2} \end{aligned} \tag{A.14}$$

A.3.5 Correlation Coefficient, r_2

$$\begin{aligned} r_2 &= \frac{n \cdot \sum xy - \left(\sum x\right)\left(\sum y\right)}{\sqrt{\left[n \cdot \sum x^2 - (x)^2\right]\left[n \cdot \sum y^2 - \left(\sum y\right)^2\right]}} \\ &= \frac{5 \cdot 4.25895 \times 10^{-4} - \left(13.19787 \times 10^{-5}\right)\left(1.7042 \times 10^{-2}\right)}{\sqrt{\left[5 \cdot 85.576977 \times 10^{-10} - \left(13.19787 \times 10^{-5}\right)^2\right]\left[5 \cdot 0.58203 - \left(1.7042 \times 10^{-2}\right)^2\right]}} \\ &= -0.9673 \end{aligned} \tag{A.15}$$

A.3.6 Coefficient of Determination, ${R_2}^2$

$${R_2}^2 = 1 - \frac{SS_{res}}{SS_{tot}} = 1 - \frac{\sum \left(y_i - \hat{y}_i\right)^2}{\sum \left(y_i - \bar{y}\right)^2} = 0.9356 \tag{A.16}$$

A.3.7 Difference in values of A

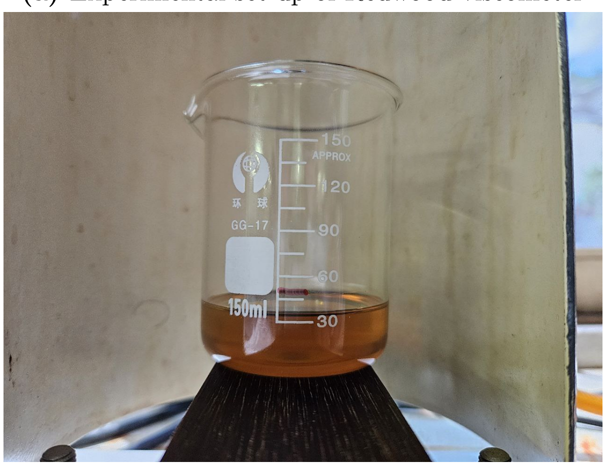
Calculations will be using value of A from Section 4.1.1, that is A_1 was determined to be $A_1 = 3.58 \times 10^{-3} cm^2 s^{-2}$. A_2 is the value of b_2, that is $A_2 \equiv b_2 = 3.533 \times 10^{-3} cm^2 s^{-2}$.

$$dA = \frac{A_1 - A_2}{A_1} \times 100\% = \frac{3.58 - 3.533}{3.58} \times 100\% = 4.70\% \tag{A.17}$$

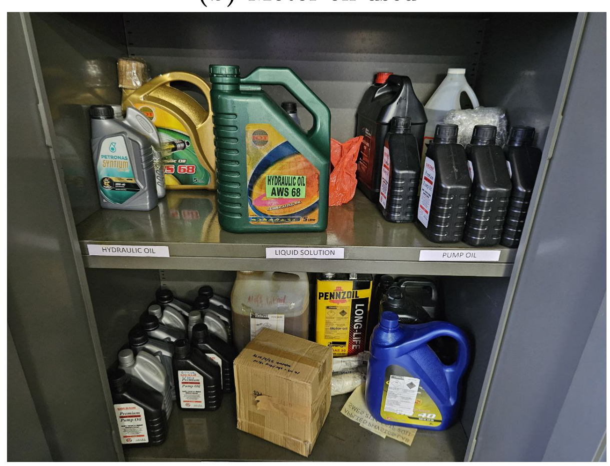
B Gallery

(a) Experimental set-up of Redwood viscometer

(b) Motor oil used



(c) 50 $m\ell$ of oil collected in beaker



(d) Hydraulic oil used (green canister)

Figure B.1: Various apparatus, procedures and materials of the experiment

C Code

C.1 Scatter plot and comparison graphs

```python
import numpy as np
import matplotlib.pyplot as plt
import sys
from matplotlib import ticker

try:
    (data1 := np.loadtxt('g_data.txt', skiprows=1))
    (data2 := np.loadtxt('g_data0.txt', skiprows=1))
except FileNotFoundError:
    print('File not found')
    sys.exit(1)

# Figure settings
plt.rcParams['figure.figsize'] = [8,5] # Set figure size
plt.rcParams['figure.dpi'] = 300              # Resolution for figure

# Set white background and black edges
plt.rcParams['figure.facecolor'] = '#ffffff'  # White background
plt.rcParams['figure.edgecolor'] = '#000000'  # Black edge color

# Font settings
plt.rcParams['figure.titlesize'] = 18
plt.rcParams['axes.labelsize'] = 13
plt.rcParams['xtick.labelsize'] = 11
plt.rcParams['ytick.labelsize'] = 11
plt.rcParams['legend.fontsize'] = 12
plt.rcParams['font.family'] = 'Times New Roman'

# Marker settings
plt.rcParams['lines.markersize'] = 7

# ============================================================================
# 1. Motor Oil (v against T)
# ============================================================================

# Create the figure and axes objects
fig1, ax1 = plt.subplots()

# Title, x and y axis of plot
ax1.set_title(r'Graph of viscosity, $v$ against temperature, $T$ (Motor Oil)',\
              fontsize=18)
ax1.set_xlabel(r'Temperature, $T$ $(K)$', fontsize=11)
ax1.set_ylabel(r'Kinematic viscosity, $v$ $(cm^2s^{-1})$', fontsize=11)

# Axes
ax1.set_xlim(300,360)
ax1.set_ylim(0,2.6)
ax1.xaxis.set_major_locator(ticker.MultipleLocator(10))
ax1.xaxis.set_minor_locator(ticker.MultipleLocator(1))
ax1.yaxis.set_major_locator(ticker.MultipleLocator(0.2))
ax1.yaxis.set_minor_locator(ticker.MultipleLocator(0.1))
ax1.grid(True, linestyle=':')

```

```python
# Plot
label1 = r'Kinematic Viscosity (Motor Oil)'
ax1.scatter(data1[:,0], data1[:,1], label=label1, color='r', marker='o')
ax1.legend()
fig1.savefig('g_expt.png')
plt.show()

# ============================================================================
# 2. Hydraulic Oil (v against T)
# ============================================================================

# Create the figure and axes objects
fig2, ax2 = plt.subplots()

# Title, x and y axis of plot
ax2.set_title(r'Graph of viscosity, $v$ against temperature, $T$ (Hydraulic Oil)',\
              fontsize=18)
ax2.set_xlabel(r'Temperature, $T$ $(K)$', fontsize=11)
ax2.set_ylabel(r'Kinematic viscosity, $v$ $(cm^2s^{-1})$', fontsize=11)

# Axes
ax2.set_xlim(300,360)
ax2.set_ylim(0,2)
ax2.xaxis.set_major_locator(ticker.MultipleLocator(10))
ax2.xaxis.set_minor_locator(ticker.MultipleLocator(1))
ax2.yaxis.set_major_locator(ticker.MultipleLocator(0.2))
ax2.yaxis.set_minor_locator(ticker.MultipleLocator(0.1))
ax2.grid(True, linestyle=':')

# Plot
label2 = r'Kinematic viscosity (Hydraulic Oil)'
ax2.scatter(data2[:,0], data2[:,1], label=label2, color='b', marker='o')
ax2.legend()
fig2.savefig('g_expt0.png')
plt.show()

# ============================================================================
# 3. Comparison between Motor and Hydraulic (v against T)
# ============================================================================

# Create the figure and axes objects
fig3, ax3 = plt.subplots()

# Title, x and y axis of plot
ax3.set_title(r'Comparison between graph of viscosity, $v$ against temperature, $T$
    (Motor and Hydraulic Oil)',\
              fontsize=12)
ax3.set_xlabel(r'Temperature, $T$ $(K)$', fontsize=11)
ax3.set_ylabel(r'Kinematic viscosity, $v$ $(cm^2s^{-1})$', fontsize=11)

# Axes
ax3.set_xlim(300,360)
ax3.set_ylim(0,2.6)
ax3.xaxis.set_major_locator(ticker.MultipleLocator(10))
ax3.xaxis.set_minor_locator(ticker.MultipleLocator(1))
ax3.yaxis.set_major_locator(ticker.MultipleLocator(0.2))
ax3.yaxis.set_minor_locator(ticker.MultipleLocator(0.1))
```

```python
ax3.grid(True, linestyle=':')

# Plot
ax3.scatter(data1[:,0], data1[:,1], label=label1, color='r', marker='o')
ax3.scatter(data2[:,0], data2[:,1], label=label2, color='b', marker='o')
ax3.legend()
fig3.savefig('g_combine.png')
plt.show()

# ===========================================================================
# 4. Comparison between Motor and Hydraulic (log{log[v + 0.8]} against log{T}
# ===========================================================================

def motor(x):
    return -15*x + 36.78

def hydraulic(x):
    return -20*x + 51.13

# Data points
maxx = 3.5
x = np.linspace(0, maxx, num=50)
y_motor = motor(x)
y_hydraulic = hydraulic(x)

# Create the figure and axes objects
fig4, ax4 = plt.subplots()

# Title, x and y axis of plot
ax4.set_title(r'Comparison between graph of $\log (\log (v+0.8))$ against $\log (T)
    $ (Motor and Hydraulic Oil)',\
              fontsize=12)
ax4.set_xlabel(r'$\log (T)$ $(K)$', fontsize=11)
ax4.set_ylabel(r'$\log (\log (v+0.8))$ $(cm^2s^{-1})$', fontsize=11)

# Intercept
ax4.scatter(2.865, -6, color='black', marker='s', s=25, label='Intercept')
ax4.axvline(2.865, ymax=0.2, color='black', linestyle='--', linewidth=1)

# Axes
ax4.set_xlim(0,maxx)
ax4.xaxis.set_major_locator(ticker.MultipleLocator(0.5))
ax4.xaxis.set_minor_locator(ticker.MultipleLocator(0.1))
ax4.yaxis.set_major_locator(ticker.MultipleLocator(10))
ax4.yaxis.set_minor_locator(ticker.MultipleLocator(2))
ax4.grid(True, linestyle=':')

# Plot
ax4.plot(x, y_motor, label=label1, color='r')
ax4.plot(x, y_hydraulic, label=label2, color='b')
ax4.legend()
fig4.savefig('g_combine2.png')
plt.show()
```

C.2 Least-square gradient

```python
from leastSquares import Experiment
from leastSquares import Graph

# ==============================================================================
# 1. Glycerol (v against y)
# ==============================================================================

# Labels
t = r'Glycerol: Least-square graph of $v$ against $t$'
x = r'Time, $t$ $(s)$'
y = r'Kinematic viscosity, $v\times 10^{-2}$ $(cm^2s^{-1})$'

g_expt1: Experiment = Experiment('g_data1.txt', ex=0, ey=-2)

g_graph1: Graph = Graph(g_expt1, col_sty=['violet','^',7,'r'], title=[t,x,y],\
                        font=[16,13,11,11,'Times New Roman'], dxdy=[200,1],\
                        ticks=[200,50,1,0.25], err=False,\
                        intercept=True, save='g_expt1.png')
# Results
print(g_expt1)
g_graph1.plot_graph()

# ==============================================================================
# 2. Glycerol (v/t against 1/t^2)
# ==============================================================================

# Labels
t = r'Glycerol: Least-square graph of $\frac{v}{t}$ against $\frac{1}{t^2}$'
x = r'Kinematic viscosity over time, $\frac{v}{t}$ $(cm^2s^{-2})$'
y = r'One over time squared, $\frac{1}{t^2}$ $(s^{-2})$'

g_expt2: Experiment = Experiment('g_data2.txt', ex=-5, ey=-2)

g_graph2: Graph = Graph(g_expt2, col_sty=['peru','^',7,'r'], title=[t,x,y],\
                        font=[16,13,11,11,'Times New Roman'], dxdy=[1e-5,1e-4],\
                        ticks=[0.00001,0.000025,0.0001,0.000025], err=False,\
                        intercept=True, save='g_expt2.png')
# Results
print(g_expt2)
g_graph2.plot_graph()

# ==============================================================================
# 3. Motor Oil (log{log[v + 0.8]} against log{T} )
# ==============================================================================

# Labels
t = r'Motor Oil: Least-square graph of $\log (\log (v+0.8))$ against $\log (T)$'
x = r'$\log (T)$ $(K)$'
y = r'$\log (\log (v+0.8))$ $(cm^2s^{-1})$'

g_expt3: Experiment = Experiment('motor_data.txt', ex=0, ey=0)

g_graph3: Graph = Graph(g_expt3, col_sty=['b','o',5,'r'], title=[t,x,y],\
                        font=[16,13,11,11,'Times New Roman'], dxdy=[0.01,0.2],\
                        ticks=[0.01,0.0025,0.2,0.05], err=False,\
```

```
                            intercept=False, save='g_expt3.png')
# Results
print(g_expt3)
g_graph3.plot_graph()


# ================================================================================
# 4. Hydraulic Oil (log{log[v + 0.8]} against log{T} )
# ================================================================================

# Labels
t = r'Hydraulic Oil: Least-square graph of $\log (\log (v+0.8))$ against $\log (T)$'
x = r'$\log (T)$ $(K)$'
y = r'$\log (\log (v+0.8))$ $(cm^2s^{-1})$'

g_expt4: Experiment = Experiment('hydraulic_data.txt', ex=0, ey=0)

g_graph4: Graph = Graph(g_expt4, col_sty=['g','o',5,'b'], title=[t,x,y],\
                            font=[16,13,11,11,'Times New Roman'], dxdy=[0.01,0.2],\
                            ticks=[0.01,0.0025,0.2,0.05], err=False,\
                            intercept=False, save='g_expt4.png')
# Results
print(g_expt4)
g_graph4.plot_graph()
```

Output

```
Experiment 1:
Least-squares Equation: y = 3.5713e-03x -3.8339e-02
Gradient: m = 3.5713e-03 +- 2.1432e-05
Gradient % Uncertainty: 0.6001%
Centroid and Intercept: cx = 5.3598e+02, cy = 1.8758e+00, b = -3.8339e-02
Correlation Coefficient: r = 0.9999
Coefficient of Determination: R2 = 0.9999

Experiment 2:
Least-squares Equation: y = -4.7212e+00x +3.5330e-03
Gradient: m = -4.7212e+00 +- 7.1492e-01
Gradient % Uncertainty: 15.1428%
Centroid and Intercept: cx = 2.6396e-05, cy = 3.4084e-03, b = 3.5330e-03
Correlation Coefficient: r = -0.9673
Coefficient of Determination: R2 = 0.9356

Experiment 3:
Least-squares Equation: y = -1.4925e+01x +3.6784e+01
Gradient: m = -1.4925e+01 +- 1.6900e+00
Gradient % Uncertainty: 11.3235%
Centroid and Intercept: cx = 2.5162e+00, cy = -7.7032e-01, b = 3.6784e+01
Correlation Coefficient: r = -0.9753
Coefficient of Determination: R2 = 0.9512

Experiment 4:
Least-squares Equation: y = -2.0712e+01x +5.1133e+01
Gradient: m = -2.0712e+01 +- 3.8730e+00
Gradient % Uncertainty: 18.6996%
```

```
Centroid and Intercept: cx = 2.5160e+00, cy = -9.7782e-01, b = 5.1133e+01
Correlation Coefficient: r = -0.9366
Coefficient of Determination: R2 = 0.8773
```

C.3 Module for least-square gradient.

```python
# -*- coding: utf-8 -*-
"""
Created on Tue Oct 22 07:11:04 2024

@author: Errol
"""

import numpy as np
import sys

# %% Class to calculate all required variables

class Experiment(object):
    tag = 1

    # %% Data attributes
    def __init__(self, filename, ex=0, ey=0, Eby=None):
        '''
        Contructs all attributes need to plot least-squares line.

        How to use:
            1) print(<instance>)
            2) .get(<str>, summ=<bool>)
            3) .show()
            4) .show_extra())

        Arguments:
            filename (.txt): One line header, following lines are measurement
                             values of xi and yi seperated by ' '. Each line
                             represents one measurement/observation
            ex       (int) : Exponent of 10^ex for x values
            ey       (int) : Exponent of 10^ey for y values
            Eby      (int) : Set error bars as uncertainty of
                             measuring instrument

        Attributes:
            xi  (numpy array): Raw data values from experiment
            yi  (numpy array): Raw data values from experiment
            yt  (numpy array): Transformed yi data using transform method
            x   (numpy array): Transformed xi data using transform method
            y   (numpy array): Mean of y for each observation
            Sy  (numpy array): Sample stdev of yt
            Eby (numpy array): Error bar of y (sample stdev of mean of y)
            xP2 (numpy array): x^2 element-wise
            yP2 (numpy array): y^2 elementwise
            xy  (numpy array): x*y element-wise
            n   (int)        : Total number of datapoints
            cx  (float)      : Centroid of x
            cy  (float)      : Centroid of y
            m   (float)      : Least-square gradient
```

```
            um  (float)      : Uncertainty of gradient
            b   (float)      : Least-square intercept
            r   (float)      : Coefficient of correlation
            R2  (float)      : Coefficient of determination

            <parameter>Sum (float): Sum of <parameter> array
            is_y_mean (bool)      : Used in class PlotGraph

        '''
        # Assign id to experiments
        self.id = Experiment.tag
        Experiment.tag += 1

        # Loads in data from txt file
        self.load_data(filename)

        # Transform
        self.transform(ex, ey)

        # Length of array (number of observations)
        self.n = len(self.x)

        # Determine if y values are mean of multiple measurements
        self.is_y_mean = bool(len(self.yt[0])-1)

        # Calculates error bars of y
        self.errorBars(Eby)

        # Array of values
        self.process()

        # Sum of parameters
        self.summation()


        # Gradient and intercept
        self.gradient()
        self.centroid()
        self.intercept()

        # Correlation coefficient
        self.cc_cod()


    # %% [1] Store x, y values from txt file to pylab array
    def load_data(self, filename):
        '''
        Reads in and store a simple series of (xi,yi) data as Numpy arrays
        from a text file. *Note: Data must first be converted into desired
        form (i.e. 1/x or ln y).

        '''
        try:
            mydata = np.loadtxt(filename, skiprows=1)
        except FileNotFoundError:
            print('File not found: ', filename)
            sys.exit(1)
```

```python

        # Store data as array to allow vectorization
        self.xi, self.yi = mydata[0:,0], mydata[0:,1:]


    # %% [2] Multiply x and y by 10^n
    def transform(self, ex, ey):
        '''
        Exponent base 10 data xi and yi.
        Mean of y for each observation.

        '''
        self.x, self.yt = self.xi*(10**ex), self.yi*(10**ey)
        self.y = np.mean(self.yt, axis=1)


    # %% [3] Calculate error bars
    def errorBars(self, Eby):
        '''
        Calculates the sample stdev of y, Sy
        and sample stdev of mean of y, Eby.

        If each y values obtained from one measurement, Eby == Sy.

        If each y values is the mean of multiple measurements, Eby = Sy/sqrt(n)

        If want to use custom error bars, Eby = [list].

        If use uncertainty of instrument as Eby, then Eby = uncertainty.

        '''

        # Eby is the uncertainty of instrument
        if type(Eby) == int or type(Eby) == float:
            self.Sy = np.full_like(self.y, Eby)
            self.Eby = self.Sy.copy()

        # Eby has custom values
        elif type(Eby) == list:
            assert len(Eby) == len(self.x), 'Length Eby is not equal to x'
            self.Sy = np.array(Eby)
            self.Eby = self.Sy.copy()

        # Singular y-values (normal sample stdev)
        elif self.is_y_mean == False:
            tot = 0
            mean = np.mean(self.y)
            for i in self.y:
                tot += (i-mean)**2
            Sy = np.sqrt(tot/(len(self.x)-1))

            self.Sy = np.full_like(self.y, Sy)
            self.Eby = self.Sy.copy()

        # Each y-values is the mean of a few measurements
        else:
            lis = []
```

```
            for i, mean in enumerate(self.y):
                lis.append(np.sqrt((np.sum((self.yt[i]-mean)**2))/\
                                   (len(self.yt[0])-1)))
            self.Sy = np.array(lis)
            self.Eby = self.Sy/np.sqrt(self.n)


    # %% [4] Process x^2, y^2 and xy values
    def process(self):
        '''
        Calculate arrays xP2, yP2 and xy.

        '''
        self.xP2, self.yP2, self.xy = self.x**2, self.y**2, self.x*self.y


    # %% [5] Sum of all arrays of data
    def summation(self):
        '''
        Calculates the sum of xi, yi, x, y, xP2, yP2 and xy arrays using numpy.

        '''
        variables = ['xi', 'yi', 'x', 'y', 'xP2', 'yP2', 'xy', 'Sy', 'Eby']
        for var in variables:
            setattr(self, var + 'Sum', np.sum(getattr(self, var)))


    # %% [6] Least-square gradient and its uncertainty
    def gradient(self):
        '''
        Calculate least-squares gradient and its uncertainty using formula
        by Year 1 Lab Manual.

        '''
        # Gradient function
        grad = lambda x, y, xP2, xy, n:\
            (n*xy - x*y) / (n*xP2 - x**2)

        # Gradient uncertainty function
        u_grad = lambda x, y, xP2, yP2, xy, n:\
            np.sqrt((n*yP2 - y**2 - (n*xy - x*y)**2 /
                        (n*xP2 - x**2)) / ((n-2) * (n*xP2 - x**2)))

        self.m = grad(self.xSum, self.ySum, self.xP2Sum,\
                      self.xySum, self.n)

        self.um = u_grad(self.xSum, self.ySum, self.xP2Sum,\
                         self.yP2Sum, self.xySum, self.n)


    # %% [7] Least-square centroid
    def centroid(self):
        '''
        Calculates centroids of least square line

        '''
        self.cx, self.cy = self.xSum / self.n, self.ySum / self.n
```

```python


    # %% [8] Least-square intercept
    def intercept(self):
        '''
        Calculate intercept of least square line.
        b = avg(y) - m * avg(x)

        '''
        self.b = self.cy - self.m * self.cx


    # %% [9] Correlation coefficient, r
    def cc_cod(self):
        '''
        Calculates correlation coefficient, r
        and coefficient of determination, R2

        '''
        self.r = ((self.n * self.xySum) - (self.xSum * self.ySum))/\
                    np.sqrt((self.n * self.xP2Sum - self.xSum**2) *\
                    (self.n*self.yP2Sum - self.ySum**2))

        SSR = np.sum((self.y - (self.m * self.x + self.b))**2)
        SST = np.sum((self.y - np.mean(self.y))**2)
        self.R2 = 1 - SSR/SST


    # %% [10] Print values of x and y method
    def show(self, summ=False):
        '''
        Prints out necessary information about least-square graph.
        Optional argument sum: Returns pylab sum of the arrays.

        '''
        variables = ['xi', 'yi', 'x', 'y']

        if summ == True:
            for var in variables:
                print('\nsum {}: {}'.format(var, getattr(self, var + 'Sum')))
        else:
            for var in variables:
                print('\n{}: {}'.format(var, getattr(self, var)))

        variables2 = ['cx', 'cy', 'm', 'um', 'b', 'r', 'R2']
        for var2 in variables2:
            print('\n{}: {:.4e}'.format(var2, getattr(self, var2)))

        return '\n' + '-' * 10 + 'Success' + '-' * 10


    # %% [11] Print values of x^2, y^2 and xy method
    def show_extra(self, summ=False):
        '''
        Prints additional information about x^2, y^2 and xy.
        Optional argument sum: Returns pylab sum of the arrays.

```

```python
        '''
        variables = ['xP2', 'yP2', 'xy',  'Sy', 'Eby']
        if summ == True:
            for var in variables:
                print('\nsum {}: {}'.format(var, getattr(self, var + 'Sum')))
        else:
            for var in variables:
                print('\n{}: {}'.format(var, getattr(self, var)))

        return '\n' + '-' * 10 + 'Success' + '-' * 10


    # %% [12] Getter method for all variables
    def get(self, var_name, summ=False):
        '''
        General method to get an attribute value.
        Set sum = True to get the sum of the attribute.

        '''
        attr_name = var_name + 'Sum' if summ else var_name
        try:
            return getattr(self, attr_name,\
                            'Sum of {} does not exist.'.format(var_name))
        except:
            raise NameError('Something went horribly wrong.')


    #%% [13] String method
    def __str__(self):

        s0 = 'Experiment {}:'.format(self.id)
        s1 = 'Least-squares Equation: y = {:.4e}x {:+.4e}'.format(self.m, self.b)
        s2 = 'Gradient: m = {:.4e} +- {:.4e}'.format(self.m, self.um)
        s3 = 'Gradient % Uncertainty: {:.4f}%'.format(100*abs(self.um/self.m))
        s4 = 'Centroid and Intercept: cx = {:.4e}, cy = {:.4e}, b = {:.4e}'\
            .format(self.cx, self.cy, self.b)
        s5 = 'Correlation Coefficient: r = {:.4f}'.format(self.r)
        s6 = 'Coefficient of Determination: R2 = {:.4f}'.format(self.R2)

        return '\n{}\n{}\n{}\n{}\n{}\n{}\n{}'.format(s0,s1,s2,s3,s4,s5,s6)


# %% Class to plot least-square graph

import matplotlib.pyplot as plt
from matplotlib import ticker

class Graph(object):

    # %% Data attributes

    def __init__(self, expt, col_sty=['b','^',8,'r'], figsize=[8,5],\
                  title=['t','x','y'], font=[18,14,12,12,'Times New Roman'],\
                      legen='best',dxdy=[0,0], ticks=False, err=True,\
                          intercept=False, save='Figure.png'):
        """
        Automatically plots least-square graph with <Experiment> object
```

```
        as argument.

        How to use:
            1) <instance>.plot_graph()

        Arguments:
            expt      (class) : Instance of experiment (least-square)
            col_sty   (list)  : Array contain [color, shape, size] of y-points
            figsize   (list)  : Array of float [width, height]
            title     (list)  : Array contain string [title,x-axis,y-axis]
            font      (str)   : Array contain  [title, x and y axis, ticks,
                                               legends, font]
            legen     (str)   : Location of legend
            dxdy      (list)  : Array containing int [dx,dy] to extend axis
            ticks     (list)  : Array containing [major_x, minor_x,
                                                 major_y, minor_y]
            err       (bool)  : Plot error bars
            intercept (bool)  : Plot least-square intercept
            save      (str)   : Name and type of file to save plot


        Attributes:
            x         (numpy array): Transformed xi data using transform method
            y         (numpy array): Mean of y for each observation
            Eby       (numpy array): Error bar of y (sample stdev of mean of y)
            cx        (float)      : Centroid of x
            cy        (float)      : Centroid of y
            m         (float)      : Least-square gradient
            b         (float)      : Least-square intercept
            is_y_mean (bool)       : Is y values mean of multiple measurements
        """

        variables = ['x', 'y', 'Eby', 'cx', 'cy', 'm', 'b', 'is_y_mean']
        for var in variables:
            setattr(self, var, expt.get(var))

        self.col_sty = col_sty
        self.figsize = figsize
        self.title = title
        self.font = font
        self.legen = legen
        self.dxdy = dxdy
        self.ticks = ticks
        self.err = err
        self.intercept = intercept
        self.save = save

    # %% [1] Linear function

    def line(self, x):
        return self.m * x + self.b


    # %% [2] Plot graph method

    def plot_graph(self):
        '''
```

```python
        Plot least-squares graph automatically.

        '''

        # Figure settings
        plt.rcParams['figure.figsize'] = self.figsize # Set figure size
        plt.rcParams['figure.dpi'] = 300              # Resolution for figure
        # Set white background and black edges
        plt.rcParams['figure.facecolor'] = '#ffffff'  # White background
        plt.rcParams['figure.edgecolor'] = '#000000'  # Black edge color

        # Font settings
        plt.rcParams['figure.titlesize'] = self.font[0]
        plt.rcParams['axes.labelsize'] = self.font[1]
        plt.rcParams['xtick.labelsize'] = self.font[2]
        plt.rcParams['ytick.labelsize'] = self.font[2]
        plt.rcParams['legend.fontsize'] = self.font[3]
        plt.rcParams['font.family'] = self.font[4]

        # Marker settings
        plt.rcParams['lines.markersize'] = self.col_sty[2]

        # Create the figure and axes objects
        fig, ax = plt.subplots()

        # Plot data points
        labell = ('Mean of ' if self.is_y_mean else '') + 'y-values'

        if self.err == True: # Plot error bars
            ax.errorbar(self.x, self.y, yerr=self.Eby, label=labell,\
                        color=self.col_sty[0],fmt=self.col_sty[1],\
                            ecolor='black',elinewidth=1,capsize=3, linewidth=2)
        else: # No error bars
            ax.scatter(self.x, self.y, label=labell, color=self.col_sty[0],\
                       marker=self.col_sty[1])

        # Plot least-square line with 200 points
        newx = np.linspace(min(self.x), max(self.x), 200)
        ax.plot(newx, self.line(newx), label='Least-square Gradient',\
                color=col_sty[3],linewidth=0.8)

        # Plot centroid and vertical line
        k = (self.cy - min(self.y) + self.dxdy[1])/ \
            (max(self.y) - min(self.y) + 2*self.dxdy[1])
        ax.scatter(self.cx, self.cy, color='black',marker='s',\
                   label='Centroid', s=25)
        ax.axvline(self.cx, ymax=k, color='black', linestyle=':', linewidth=1)

        # Plot intercept, b
        if self.intercept == True:
            ax.scatter(0, self.b, color='cyan', marker='s', s=25,\
                       label='Intercept')
            ax.axvline(0, ymax=1, color='black', linestyle='--',\
                       linewidth=1)

        # Title, x and y axis of plot
        ax.set_title(self.title[0], fontsize=self.font[0])
```

```
        ax.set_xlabel(self.title[1], fontsize=self.font[1])
        ax.set_ylabel(self.title[2], fontsize=self.font[1])

        # Axes limits
        ax.set_xlim(min(self.x)-self.dxdy[0], max(self.x)+self.dxdy[0])
        ax.set_ylim(min(self.y)-self.dxdy[1], max(self.y)+self.dxdy[1])

        # Ticks
        try: # Tick values are specified
            ax.xaxis.set_major_locator(ticker.MultipleLocator(self.ticks[0]))
            ax.xaxis.set_minor_locator(ticker.MultipleLocator(self.ticks[1]))
            ax.yaxis.set_major_locator(ticker.MultipleLocator(self.ticks[2]))
            ax.yaxis.set_minor_locator(ticker.MultipleLocator(self.ticks[3]))
        except: # No tick values were specified
            pass

        finally:
            # Gridlines
            ax.grid(True, linestyle=':')

            # Axes legends
            ax.legend(loc=self.legen)

            # Save fig
            fig.savefig(self.save)
            plt.show()
```