Experimental analysis of the effect of Polarizers and Reflections on light inensity.

Sebastien Psarianos
Sofiya P'yavka

November 28, 2022

1 Introduction

The objective of this experiment was to determine the elementary charge using a Millikan apparatus. Oil droplets are electrically charged due to the friction they experience in the capillary of the apparatus' atomizer and then enter a chamber that consists of two charged parallel plates which produce a constant electric field. Three forces then act on the oil droplets; the downward gravitational force $F_g = m_o g$ where m_o is the mass of the oil droplet and g is the gravitational acceleration, the upward buoyant force $F_g = -m_a g$ where m_a is the mass of the air displaced by the oil droplet, the electric force $F_E = QE$ where Q is the charge of the oil droplet and E is the electric field and Stokes' resistive force $F_d = 6\pi r$ (long n) ν where r is the radius of the spherical oil droplet, (long n) is the viscosity and ν is the velocity under laminar flow conditions. Depending on the set voltage, the motion of the oil droplet changes as follows;

i) If the oil droplet is floating in the chamber, the net force is zero which can described by:

$$g(m_o - m_a) - \frac{Q v_{stop}}{d} = 0$$

Equation 1: Static particle equation

Where V_{stop} is the voltage which allows the droplet to remain suspended and d is the distance between the charged parallel plates.

ii) If the droplet is moving downward with terminal velocity ν_t, the net force is zero and can be described by:

$$g(m_o - m_a) - 6\pi r \text{ (long n) } \nu_t = 0$$

Equation 2: Terminal falling velocity equation

iii) If the oil droplet is accelerating upwards the equation of motion is described by

$$ma = -g(m_o - m_a) + QE - 6\pi r \text{ (long n) } \nu$$

Equation 3: Upwards accelerating droplet equation

The elementary charge can then be determined using two methods. If the experimenter determines the stopping voltage and terminal velocity of an oil droplet, the following equation can be derived from **Equation 1** and **Equation 2**.

$$Q = C\frac{v_t^{\frac{3}{2}}}{v_{stop}}$$

Equation 4: Method one charge calculation model

Where C is a constant defined by:

$$C = \frac{18\pi d\ (\text{long n})^{\frac{3}{2}}}{\sqrt{2}\sqrt{\rho_o - \rho_a}\sqrt{g}}$$

Constant 1

Where the densities $\rho_o = \frac{4}{3}\pi r^3 m_0$ and $\rho_a = \frac{4}{3}\pi r^3 m_a$
In contrast, if the experimenter determines the terminal velocity of the oil drop, the voltage V_{up} at which the oil droplet is moving upward with a constant speed ν_2, the following equation can be derived from **Equations 1,2 and 3**;

$$Q = C(\nu_1 + \nu_2)\frac{\nu_t^{\frac{1}{2}}}{\nu_{up}}$$

Equation 5: Method two charge calculation model

Where C is **Constant 1**.

2 Methodology and Procedure

3 Results

Note, all referenced functions, equations and constants can be found in the appendix.

To determine the uncertainty of the stopping and upward voltages, **Function 1** was defined based off of Hewlett-Packard 3476A Digital Multimeter specifications.

To find the charge of the oil droplet for each trial, the terminal velocity and constant upward velocity had to be determined from the experimental data. To convert the position data from pixels to units of meters, the position data was divided by a factor of 540000 since the calibration factor of the left camera was found to be $(540 \pm 1)px/mm$. Since the camera had a frame rate of $10Hz$, for each individual trial and its respective position data set, an array of times was generated where the time elapsed per frame was $0.1s$. The two sets of position and time data were then fitted to a linear model as implemented by **Function 2** by passing them through scipy's curve_fit function. The uncertainty was determined using curve_fit's covariance values since the diagonals of the covariance matrix represent the variability of each parameter and the square root of this was taken to calculate the standard error. From this an array of terminal and constant upward

velocities was determined.

Note **Constant 1** was derived using **Equation 1**, **Equation 2** and **Equation 3** and its numerical value was determined using known physical constants which can be found in the appendix.

Method 1

Equation 4 as implemented by **Function 3** was used to generate an array of charges for each individual trial by passing in the stopping voltage and terminal velocity arrays. The uncertainty of the charges was determined using **Equation 6** as implemented by **Function 5**.

Method 2

Equation 5 as implemented by **Function 4** was used to generate an array of charges for each individual trial by passing in the terminal velocity, constant upward velocity and upward voltage arrays. The uncertainty of the charges was determined using **Equation 7** as implemented by **Function 6**.

Histograms for both methods were then generated as depicted in **Figure 2** and **Figure 3**. **Function 7** was used to determine the average uncertainty of the charges for both methods.

INSERT GRAPHS AND GCD STUFF

4 Analysis and Discussion

5 Conclusion

6 Appendix

Add Physical Constants
Equations

$$g(m_o - m_a) - \frac{Q v_{stop}}{d} = 0$$

Equation 1: Static particle equation

$$g(m_o - m_a) - 6\pi r \text{ (long n) } \nu_t = 0$$

Equation 2: Terminal falling velocity equation

$$ma = -g(m_o - m_a) + QE - 6\pi r \text{ (long n) } \nu$$

Equation 3: Upwards accelerating droplet equation

$$Q = C\frac{v_t^{\frac{3}{2}}}{v_{stop}}$$

Equation 4: Method one charge calculation model

$$Q = C(\nu_1 + \nu_2)\frac{\nu_t^{\frac{1}{2}}}{\nu_{up}}$$

Equation 5: Method two charge calculation model

$$u(Q(x,y)) = \sqrt{\left(\frac{3c}{2}\frac{x^{\frac{1}{2}}}{y}\right)^2 (u(x))^2 + \left(\frac{-Cx^{\frac{3}{2}}}{y^2}\right)(u(y))^2}$$

Equation 6: Method one charge calculation error propagation (derived from general uncertainty propagation equation)

$$u(Q(x,y,z)) = \sqrt{\left(\frac{3}{2}\frac{C}{z}x^{\frac{1}{2}} + \frac{1}{2}C\frac{y}{zx^{\frac{1}{2}}}\right)^2 (u(x))^2 + \left(\frac{Cx^{\frac{1}{2}}}{z}\right)^2 (u(y))^2 + \left(\frac{-Cx^{\frac{3}{2}}}{z^2} - \frac{Cyx^{\frac{1}{2}}}{z^2}\right)^2 (u(z))^2}$$

Equation 7: Method two charge calculation error propagation (derived from general uncertainty propagation equation)

Functions

```
    def uncertaintyVoltage(reading, ran):
        return 0.006 * reading + 0.001 * ran
```

Function 1: Function used to calculate voltage uncertainty

```
def linear(time, velocity, intercept):
    return time * velocity + intercept
```

Function 2: General linear model

```
def chargeMethodOne(velocityOne, voltageOne):
    return constant * velocityOne ** (3 / 2) / voltageOne
```

Function 3: Method one charge model (implements Equation 4)

```
    def chargeMethodTwo(velocityOne, velocityTwo, voltageTwo):
    return (
        constant
        * (abs(velocityOne) + velocityTwo)
        * abs(velocityOne) ** (1 / 2)
        / voltageTwo
    )
```

Function 4: Method two charge model (implements Equation 5)

```
    def uncertaintyChargeMethodOne(
        velocityOne, voltageOne, uncertaintyOne, uncertaintyTwo
    ):
        return np.sqrt(
            (3 * constant / 2 * np.sqrt(abs(velocityOne)) / voltageOne) ** 2
            * uncertaintyOne ** 2
            + (-1 * constant * abs(velocityOne) ** (3 / 2) / voltageOne ** 2) ** 2
            * uncertaintyTwo ** 2
        )
```

Function 5: Method one uncertainty propagation (implements Equation 6)

```
    def uncertaintyChargeMethodTwo(
        velocityOne,
        velocityTwo,
        voltageTwo,
        uncertaintyOne,
        uncertaintyTwo,
        uncertaintyThree,
    ):
        return np.sqrt(
            (
                (3 * constant / 2 * np.sqrt(abs(velocityOne)))
                + (
                    constant
                    / 2
                    * velocityTwo
                    / voltageTwo
                    / np.sqrt(abs(velocityOne))
                )
            )
            ** 2
            * uncertaintyOne ** 2
            + (constant * np.sqrt(abs(velocityOne)) / voltageTwo) ** 2
            * uncertaintyTwo ** 2
            + (
                (-1 * constant * abs(velocityOne) ** (3 / 2) / voltageTwo ** 2)
                - (
                    constant
                    * velocityTwo
                    * np.sqrt(abs(velocityOne))
                    / voltageTwo ** 2
                )
            )
            ** 2
            * uncertaintyThree ** 2
        )
```

Function 6: Method two uncertainty propagation (implements Equation 7)

```
def averageUncertainty(uncertainty):
    s = 0
    for value in uncertainty:
        s += value ** 2
    return np.sqrt(s) / len(uncertainty)
```

Function 7: Function used to calculate average uncertainty for histogram

Raw Data