Laboratory 1 Report: Investigation of Fluid Properties

Contents

Objective: To measure and analyze the density, specific gravity, and viscosity of fluids using experimental methods.

Background/Theory

Density:
Defined as mass per unit volume, calculated using:

$$\rho = \frac{M}{V}$$

Specific Weight:
Weight per unit volume, calculated using:

$$\gamma = \rho g$$

Specific Gravity:
Ratio of a fluid's density to water's density:

$$SG = \frac{\rho_{fluid}}{\rho_{water}}$$

Viscosity:

↑ Back to top

- Dynamic viscosity (μ) measures resistance to shear stress:
  $$\tau = \mu \frac{du}{dy}$$
- Kinematic viscosity (ν) relates dynamic viscosity to fluid density:
  $$\nu = \frac{\mu}{\rho}$$

Stokes' Law:
Terminal velocity of a sphere in a fluid relates to viscosity:

$$u = \frac{gd^2}{18\nu}\left(\frac{\sigma}{\rho} - 1\right)$$

where

- u = velocity in meters/second
- d is sphere diameter in meters
- ν is viscosity in $\frac{N \cdot s}{m^2}$ or $Pa \cdot s$
- σ is density of the sphere in $\frac{kg}{m^3}$
- ρ is density of the liquid in $\frac{kg}{m^3}$

The equation is be rearranged to solve for viscosity in terms of velocity (a measured value) as

$$\nu = \frac{gd^2}{18u}\left(\frac{\sigma}{\rho} - 1\right)$$

Materials and Equipment

1. Liquids: Water, salt water, glycerin
2. Measurement tools: Thermometer, hydrometer, graduated cylinders, scale, and stopwatch/iPhone
3. Steel balls of varying diameters
4. Other equipment: Beakers and recording devices

Experimental Setu

↑ Back to top

1. Fluids were prepared at ambient temperatures, with their densities and volumes measured.
2. Steel balls were dropped into graduated cylinders filled with fluids, and their fall times were recorded.

Procedure

1. **Temperature Measurement:** Record temperature of each fluid using a thermometer.
2. **Mass Measurement:** Use a scale to measure the mass of beakers and fluid samples.
3. **Volume Measurement:** Measure fluid volumes with graduated cylinders.
4. **Density Calculation:** Calculate density using mass and volume.
5. **Viscosity Tests:** Record the time for steel balls to fall through the fluids, and calculate dynamic and kinematic viscosities using recorded times and Stokes' Law.

Part I - Density Measurement

1. Ensure the balance is level and calibrated before use.
2. Record the empty beaker's mass (tare).
3. Add the assigned fluid to the beaker and record the combined mass.
4. Measure the fluid's volume using a graduated cylinder.

5. Repeat the above steps three times for each assigned fluid.
6. Record the temperature of the liquid.

Part II - Specific Gravity

1. Fill a tall, transparent container with the assigned fluid.
2. Gently submerge the calibrated hydrometer into the fluid.
3. Wait until the hydrometer stabilizes and record the specific gravity at the lower meniscus.
4. Repeat the process three times for each assigned fluid.

Part III - Viscosity

1. Select a steel sphere and measure its diameter using a micrometer.
2. Drop the sphere into the tall column of fluid at the guide.
3. Start the stopwatch when the sphere passes the first marker and stop it at the second marker.
4. Record the time taken for the sphere to travel the marked distance.
5. Repeat the steps three times for each sphere (or just use three spheres).

Safety Precautions

1. Ensure all glassware is handled carefully to avoid breakage.
2. Use care to avoid splashes; safety goggles would be a good idea.
3. Handle fluids and steel spheres with clean, dry hands to avoid contamination.
4. Maintain a clean workspace to avoid cross-contamination of samples.
5. Wash hands after experiments are completed.

Data Collection

↑ Back to top

Tables of Recorded Data:

The tables below are recorded and computed data for the experiment(s). The computational tools are listed in a following section.

Density

Fluid	Temperature (°C)	Mass Beaker (g)	Mass Beaker+Liquid (g)	Mass Liquid(g)	Volume (mL)	Density (g/L)	Density (literature)
Water	19.0	120	179.1	59.1	59	1001.7	998.2
Salt Water	19.5	120	204.0	84	75	1120.0	(Sat. Sol. NaCl) 1202
Glycerine	20.0	88	138.0	50	40	1250.0	1261

Specific Gravity

Fluid	SG (from density)	SG (Hydrometer)	Remarks
Water	1.000	1.00	Water is reference
Salt Water	1.118	1.08	Scale hard to read
Glycerine	1.248	N/A	No hydrometer available

Water Viscosity

Sphere ID	Diameter (mm)	Mass (g)	Distance (cm)	Time (s)	Kinematic Viscosity (m²/s)	Dynamic Viscosity (Pa·s)	Remarks
small	1.57	0.016	24.7	0.67	2.4958×10^{-08}	0.000025	
medium	3.14	0.127	24.7	0.40	2.4958×10^{-08}	0.000059	
large	12.67	8.45	24.7	0.10	2.4458×10^{-07}	0.000245	

Salt Water Viscosity

Sphere ID	Diameter (mm)	Mass (g)	Distance (cm)	Time (s)	Kinematic Viscosity (m²/s)	Dynamic Viscosity (Pa·s)	Remarks
small	1.57	0.016	24.7	0.65	1.8750×10^{-08}	0.000021	
medium	3.14	0.127	24.7	0.38	4.4643×10^{-08}	0.000050	
large	12.67	8.45	24.7	0.09	1.3721×10^{-07}	0.000194	

Glycerine Viscosity ↑ Back to top

Sphere ID	Diameter (mm)	Mass (g)	Distance (cm)	Time (s)	Kinematic Viscosity (m²/s)	Dynamic Viscosity (Pa·s)	Remarks
small	1.57	0.016	11.6	16.81	8.272×10^{-07}	0.001034	
medium	3.14	0.127	11.6	3.29	6.416×10^{-06}	0.000802	
large	12.67	8.45	11.6	3.03	9.7664×10^{-06}	0.012208	

Viscosity Summary

Results reported as $\bar{\nu}$ (SD)

Liquid ID	Kinematic Viscosity (m²/s)	Dynamic Viscosity (Pa·s)	Remarks
Water	9.8165×10^{-08} (1.268×10^{-07})	0.0001097 (0.0001184)	Literature value 0.0010016 100X too small
Brine	6.6868×10^{-08} (6.2279×10^{-08})	0.0000883 (0.0000927)	Saturated brine 3X larger than fresh; 300X too small
Glycerine	5.6699×10^{-06} (4.5161×10^{-06})	0.0046813 (0.0065193)	Literature value 0.56183 100X too small

Analysis

Data analysis scripts are listed in the appendix and were used to populate the tables above. Two key observations are:

1. The reported densities are consistent with values in the open literature, whereas the viscosities are 100-fold too small. This under-report could be a 100X error in the computational script in the appendix most likely attributed to a unit conversion error somewhere in the code.
2. The standard deviations and mean values of viscosity are about the same - generally that means low reliability in the experiments (If the standard deviation is close to the mean, it indicates a lack of precision.). Causes could be measurement errors, especially in time - or just as likely the spheres have not reached their terminal velocity in the cylinders. Either way (misreading the videos, or spheres still decelerating) the high standard deviation relative to the mean values means these results are suspect.

Conclusion(s)

Glycerine exhibited higher viscosity, evident from longer fall times of the steel balls. Salt content influenced water properties. The solution was close to saturation as indicated by residual solids in the storage vessel. The anticipated increased viscosity of the brine solution was not observed.

Likely sources of error include timing inaccuracies and limited trials for some measurements. A unit conversion error in the data reduction script may explain the 10(↑ Back to top es observed for viscosity.

As noted above in the data analysis section, these results are suspect because the standard deviation is same order of magnitude as the mean values These experiments should be repeated with more attention paid to gathering the viscosity data, including using taller cylinders for the Stoke's Law application (to ensure terminal velocity is reached), and better video logging to observe the spheres traverse the known distance.

Such efforts have relevance to lubrication systems, such as determining optimal oil viscosities for machinery, or selecting drag reducing agents for oil pipelines.

References

1. Engineering Fluid Mechanics - Chapter 2
2. Engineering Fluid Mechanics - Chapter 11 (Cd for spheres, p.414)
3. Cleveland, T. G. (2024) Fluid Mechanics Laboratory Notes to accompany CE-3105, Department of Civil, Environmental, and Construction Engineering, Whitacre College of Engineering.
4. Holman, J.P. (2012). *Experimental Methods for Engineers*, 8th Ed.
5. Laboratory 1 Instructional Video by Dr. Uddameri
6. Measuring Liquid Density (YouTube)
7. Measuring Viscosity Falling Sphere (YouTube)

Scripts to compute various values shown in tabulations

The script below is used to compute **density** from the recorded measurements; output is $\frac{grams}{Liter}$

```
# Function to ensure valid numeric input
def get_valid_float(prompt):
    while True:
        try:
            return float(input(prompt))
        except ValueError:
            print("Invalid input. Please enter a numeric value.")

# Function to ensure valid string input
def get_valid_string(prompt, default="Not Specified"):
    while True:
        user_input = input(prompt).strip()
        if user_input:
            return user_input
        else:
            print(f"Invalid input. Defaulting to '{default}'.")
            return default

# Prompt for inputs
liquidName = get_valid_string("Enter the name of the liquid (default: Not Specified): ")
mass_Beaker = get_valid_float("Enter the mass of the beaker (g): ")
mass_LiquidAndBeaker = get_valid_float("Enter the mass of the liquid and beaker combined (g): ")
vol_Liquid = get_valid_float("Enter the volume of the liquid (mL): ")

# Convert volume to liters
vol_Liquid = vol_Liquid / 1000  # Convert mL to Liters

# Calculate density
density = (mass_LiquidAndBeaker - mass_Beaker) / vol_Liquid

# Output the result
print("\nResults:")
print(f"Liquid Name: {liquidName}")
print(f"Density of {liquidName}: {round(density, 3)} g/L")
```

light/dark

The script below is used to compute **specific gravity** from the recorded measurements; output is dimensionless

```
# Script to calculate specific gravity

# Function to ensure a valid string input
def get_valid_string(prompt, default="Not Specified"):
    while True:
        user_input = input(prompt).strip()
        if user_input:
            return user_input
        else:
            print(f"Invalid input. Defaulting to '{default}'.")
            return default

# Function to ensure a valid numeric input
def get_valid_float(prompt):
    while True:
        try:
            return float(input(prompt))
        except ValueError:
            print("Invalid input. Please enter a numeric value.")

# Prompt for inputs
liquid_name = get_valid_string("Enter the name of the liquid (default: Not Specified): ")
density_named_liquid = get_valid_float("Enter the density of the named liquid (mg/L): ")
density_reference_liquid = get_valid_float("Enter the density of the reference liquid (mg/L): ")

# Calculate specific gravity
if density_reference_liquid != 0:  # Avoid division by zero
    specific_gravity = density_named_liquid / density_reference_liquid
else:
    specific_gravity = None
    print("Error: The density of the reference liquid cannot be zero.")

# Output the result
print("\nResults:")
print(f"Liquid Name: {liquid_name}")
if specific_gravity is not None:
    print(f"Specific Gravity (Named/Reference): {specific_gravity:.3f}")
else:
    print("Specific Gravity calculation failed due to invalid reference density.")
```

↑ Back to top

The script below is used to compute **viscosity** from Stoke's law using the recorded measurements; output is $Pa \cdot s$

```
# Estimate Viscosity from Stokes Law
# Support functions
def volSphere(diameter):
    import math
    diameter = diameter / 1000  # convert mm into m
    volSphere = (4. / 3.) * math.pi * (diameter / 2.)**3
    return volSphere

def _viscosity(diameter,velocity,sigma,rho):
    gravity = 9.8
    _viscosity = ((gravity * (diameter**2)) * ((sigma/rho)-1)) / (18.0 * velocity)
    return(_viscosity)

# Function to ensure a valid string input
def get_valid_string(prompt, default="Not Specified"):
    while True:
        user_input = input(prompt).strip()
        if user_input:
            return user_input
        else:
            print(f"Invalid input. Defaulting to '{default}'.")
            return default

# Function to ensure a valid numeric input
def get_valid_float(prompt):
    while True:
        try:
            return float(input(prompt))
        except ValueError:
            print("Invalid input. Please enter a numeric value.")

# Prompt for inputs
liquidName = get_valid_string("Enter the name of the liquid (default: Not Specified): ")
gravity = 9.81  # m/s^2 (constant)

# Numeric inputs
density_liquid = get_valid_float("Enter the liquid density (mg/L): ")
sphereDiameter = get_valid_float("Enter the sphere diameter (mm): ")
sphereMass = get_valid_float("Enter the sphere mass (g): ")
fall_distance = get_valid_float("Enter the fall distance (cm): ")
fall_time = get_valid_float("Enter the fall time (sec): ")

# Intermediate Calculations
sphereVolume = volSphere(sphereDiameter)  # volume in m^3
density_sphere = sphereMass / sphereVolume  # density in g/m^3
density_sphere /= 1000  # cor ↑ Back to top ^3
fallspeed = fall_distance / fa  " cm/sec
fallspeed = fallspeed/100 # convert to m/s
sphereDiameter = sphereDiameter/1000 # convert to m
# Output the results
print("\nResults:")
print(f"Liquid Name: {liquidName}")
print(f"Sphere Diameter: {sphereDiameter:.5f} meters")
print(f"Sphere Volume: {sphereVolume:.8f} cubic meters")
print(f"Sphere Mass: {sphereMass/1000:.5f} kg")
print(f"Fall Distance: {fall_distance/100:.5f} meters")
print(f"Fall Time: {fall_time:.3f} sec")
print(f"Fall Speed: {fallspeed:.5f} m/sec")
print(f"Liquid Density: {density_liquid:.3f} kg/m^3")
print(f"Sphere Density: {density_sphere:.3f} kg/m^3")
# Viscosity Calculations

viscosity = _viscosity(sphereDiameter,fallspeed,density_sphere,density_liquid)
# Viscosity Output
print(f"Stokes Flow Viscosity: {viscosity:.6f} Ns/m^2")
```