The University of Texas at Austin
Aerospace Engineering
and Engineering Mechanics
Cockrell School of Engineering

ASE 162M High-Speed Aerodynamics
Section 14275

Tuesay: 4:00 - 6:00 pm

Report 1:
Supersonic Flow over a Sphere

Andrew Doty
Due Date: 10/15/2024

Contents

1 Introduction

Supersonic flow over blunt bodies is a fundamental problem in compressible aerodynamics, with significant implications for the design of high-speed aircraft, spacecraft re-entry vehicles, and projectiles. When a supersonic flow encounters a blunt object such as a sphere, it forms a detached bow shock upstream of the body. The characteristics of this shock, particularly its standoff distance from the body, are crucial parameters in understanding the aerodynamic forces, heat transfer, and pressure distribution on the object.

The primary objective of this laboratory experiment is to investigate the behavior of supersonic flow over a sphere across a range of Mach numbers. Specifically, we aim to:

1. Visualize the flow field and shock structure using Schlieren and shadowgraph imaging techniques.
2. Measure and analyze the non-dimensional shock standoff distance as a function of Mach number.
3. Compare experimental results with theoretical predictions through curve fitting of empirical relations.
4. Examine the effects of Mach number and Reynolds number on the flow characteristics.

The experiment utilizes the Aerolab Variable Mach Number Wind Tunnel, capable of producing flows with Mach numbers ranging from 1.4 to 3.5. A sphere model is mounted in the test section, and flow visualization is achieved using a folded Schlieren system with a pulsed xenon arc lamp. Pressure measurements in the stagnation chamber and test section provide data for calculating flow parameters.

This report presents the experimental methodology, data analysis, and discussion of results. We will examine the Schlieren images to identify key flow features, calculate Reynolds numbers and Mach numbers for each test condition, and analyze the shock standoff distance data. Three curve-fitting models will be applied to the standoff distance measurements:

1. A qualitative scaling relation: $\frac{\delta}{D} = c\sqrt{\frac{1+\frac{\gamma-1}{2}M_\infty^2}{M_\infty - 1}}$
2. A basic power-law fit: $\frac{\delta}{D} = c\gamma^\alpha M^\beta$
3. An offset power-law fit: $\frac{\delta}{D} = c\gamma^\alpha (M-1)^\beta$

By comparing these models, we aim to gain insights into the physical mechanisms governing shock standoff distance and evaluate the applicability of different theoretical approaches.

The findings of this experiment contribute to our understanding of supersonic flow phenomena and provide valuable data for validating computational fluid dynamics (CFD) simulations and theoretical models. Such knowledge is essential for advancing the design and optimization of supersonic and hypersonic vehicles, as well as for predicting aerodynamic heating and loads in high-speed flight regimes.

2 Experimental Setup

The experiment was conducted using the Aerolab Variable Mach Number Wind Tunnel, capable of producing supersonic flows with Mach numbers ranging from approximately 1.63 to 3.25. The test section of the wind tunnel is nominally 3" * 3". A sphere model was mounted in the test section at a 0° angle of attack.

2.1 Calibration and Imaging

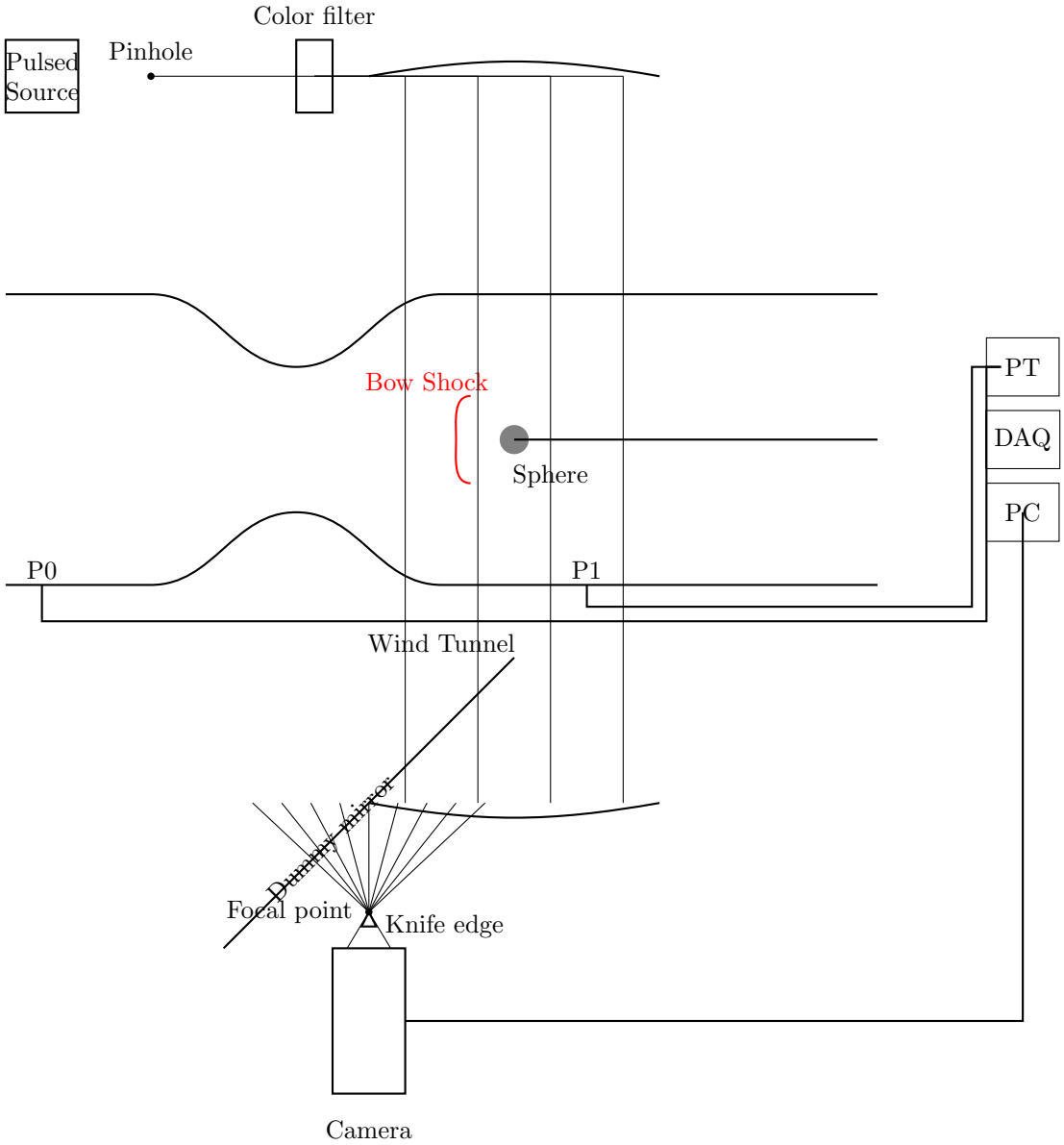
Prior to running the experiments, a calibration process was performed:

- A 30/30, 5mm grid points calibration sheet was used for spatial calibration.
- 10 grid spaces, equivalent to 50mm, were used to determine the pixel-to-length ratio and associated uncertainty.
- A blank image without the calibration sheet was taken, followed by an image with the calibration sheet in place.

Flow visualization was achieved using both shadowgraph and Schlieren imaging techniques. A folded Schlieren system utilizing a pulsed xenon arc lamp as the light source was employed. For each Mach number, three types of images were captured:

1. Shadowgraph
2. Vertical Schlieren
3. Horizontal Schlieren

For the Schlieren setup, a knife edge was placed at the focal point to enhance the visualization of density gradients. The vertical and horizontal indicate the orientation of the blade.



Note for Ryan, this was my first tikz diagram of this type, and I am still learning so please don't grade me too harshly :)

2.2 Data Acquisition

Pressure measurements were recorded using a LabVIEW-based data acquisition system. Two pressure transducers were used:

- Channel 0: Stagnation pressure (P_0)
- Channel 1: Freestream static pressure

The pressure data was recorded in volts (gauge pressure) and later converted to appropriate units using calibration constants.

2.3 Experimental Procedure

The experiment followed these steps for each run:

1. The wind tunnel was started in the sequence: power, hydraulic, then run.
2. The initial Mach number was set to approximately 3, with subsequent runs decreasing by increments of 0.25.
3. Once the LabVIEW system showed stable pressure readings, images were captured using the Pulnix color CCD camera and XCAP for Windows acquisition software.
4. The wind tunnel was operated only while the run button was held, and shut down in the reverse order of startup once the airflow stopped.
5. Mach number was adjusted by changing the area ratio and chamber pressure according to the manufacturer's specifications, referencing the "Approximate Minimum Stagnation Pressure vs Mach Number" chart.

Care was taken to monitor the pressure of the compressed air tank to ensure consistent flow quality across all runs. The experiment was repeated for Mach numbers of 3, 2.75, 2.5, 2.25, 2, and 1.75.

3 Results and Discussion

3.1 Reynolds Number and Mach Number Calculations

To calculate the Reynolds numbers and Mach numbers, we used the following equations:

3.1.1 Mach Number Calculation

The Mach number was calculated using the isentropic flow equation:

$$\frac{p_0}{p} = \left(1 + \frac{\gamma - 1}{2} M^2\right)^{\frac{\gamma}{\gamma - 1}} \tag{1}$$

where p_0 is the stagnation pressure, p is the static pressure, γ is the ratio of specific heats for air (1.4), and M is the Mach number.

3.1.2 Reynolds Number Calculation

The Reynolds numbers were calculated using:

$$Re_D = \frac{\rho_\infty U_\infty D}{\mu} \tag{2}$$

where ρ_∞ is the freestream density, U_∞ is the freestream velocity, D is the sphere diameter, and μ is the dynamic viscosity.

The viscosity was calculated using Sutherland's law:

$$\mu = \mu_0 \left(\frac{T}{T_0}\right)^{3/2} \frac{T_0 + C}{T + C} \tag{3}$$

where μ_0 is the reference viscosity, T_0 is the reference temperature, T is the flow temperature, and C is Sutherland's constant for air.

Using the data from the pressure measurements and the Python script, we calculated the unit Reynolds numbers and diametric Reynolds numbers (Re_D) for all runs, as well as the Mach numbers. Sutherland's law was used for viscosity calculations. The results are presented in the following plot:

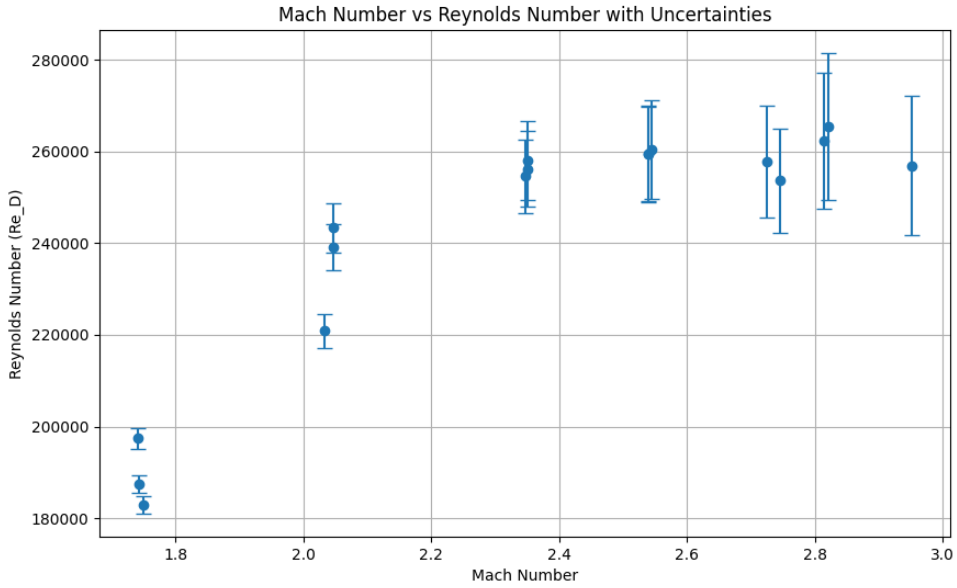


Figure 1: Mach number vs. Diametric Reynolds number

Figure **??** shows the relationship between the calculated Mach number and the diametric Reynolds number. We observe a positive correlation, with both Mach number and Reynolds number increasing together. This trend is expected due to higher flow velocities at increased Mach numbers.

The error bars represent propagated uncertainties from measurement errors. For Mach number, the uncertainty is primarily due to pressure measurement errors:

$$\delta M = \sqrt{\left(\frac{\partial M}{\partial p_0}\delta p_0\right)^2 + \left(\frac{\partial M}{\partial p}\delta p\right)^2} \tag{4}$$

where δp_0 and δp are uncertainties in stagnation and static pressure measurements, respectively.

For Reynolds number, uncertainties propagate from multiple sources:

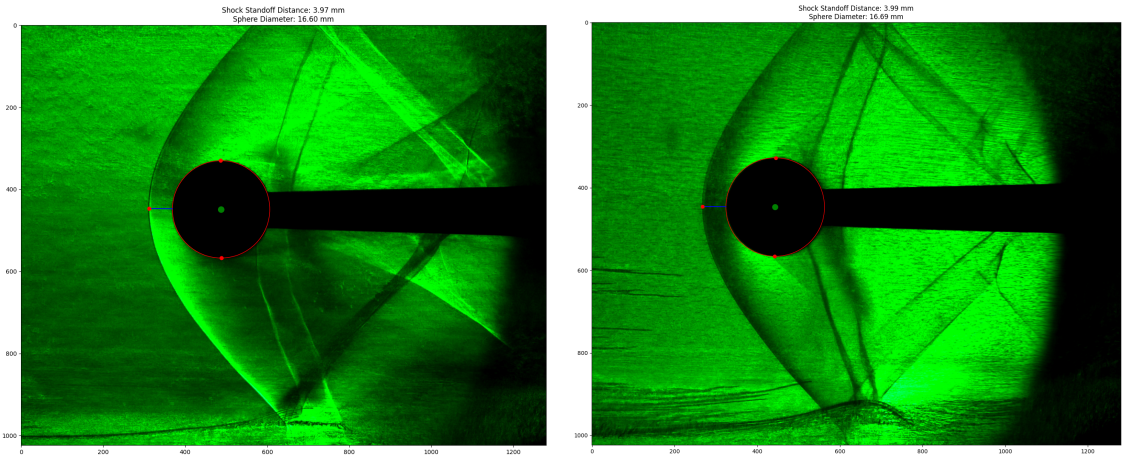
$$\delta Re_D = Re_D \sqrt{\left(\frac{\delta\rho}{\rho}\right)^2 + \left(\frac{\delta U}{U}\right)^2 + \left(\frac{\delta D}{D}\right)^2 + \left(\frac{\delta\mu}{\mu}\right)^2} \tag{5}$$

where $\delta\rho$, δU, δD, and $\delta\mu$ are uncertainties in density, velocity, diameter, and viscosity calculations, respectively.

The increasing size of error bars at higher Mach and Reynolds numbers indicates that measurement uncertainties have a more significant impact in these regimes, likely due to the nonlinear relationships in the governing equations.

3.2 Schlieren Images Analysis

We analyzed the Schlieren images for the highest and lowest Mach numbers to observe the flow features around the sphere.



(a) Lowest Mach number Horizontal(M = 1.75)

(b) Lowest Mach number Vertical (M = 1.75)

(c) Highest Mach number Horizontal (M = 3.0)

(d) Highest Mach number Vertical(M = 3.0)

Figure 2: Schlieren images at extreme Mach numbers

In Figure **??**, we can observe the following key features:

1. Bow shock: A strong, detached shock wave is visible in front of the sphere for both Mach numbers. The shock is more oblique and closer to the sphere at the higher Mach number.
2. Shock standoff distance: The distance between the bow shock and the sphere's leading edge is noticeably larger for the lower Mach number (Figure **??**) compared to the higher Mach number (Figure **??**).
3. Expansion region: Behind the sphere, we can see an expansion fan where the flow accelerates and turns around the sphere.

4. Wake region: A turbulent wake is visible downstream of the sphere, characterized by density gradients in the Schlieren images.
5. Boundary layer: Although not clearly visible due to the image resolution, a thin boundary layer is expected to form on the sphere's surface.

The changes in the flow structure as the Mach number increases from 1.75 to 3.0 are consistent with theory:

1. The bow shock moves closer to the sphere at higher Mach numbers, reducing the shock standoff distance.
2. The shock becomes stronger and more oblique at higher Mach numbers, as evidenced by the sharper contrast in the Schlieren image.
3. The wake region appears to be more compressed and elongated at the higher Mach number, likely due to the increased dynamic pressure.
4. These changes correlate with the increase in Reynolds number observed in Figure **??**.

3.3 Shock Standoff Distance Analysis

We analyzed the non-dimensional shock standoff distance δ/D as a function of Mach number using the formula:

$$\frac{\delta}{D} = f(M) \tag{6}$$

where δ is the shock standoff distance and D is the sphere diameter. The standoff distance δ and sphere diameter D were measured using a custom image analysis script developed for this experiment. This script processes the Schlieren images, allowing for precise measurement of the shock location and sphere dimensions. The visual can be shown below.

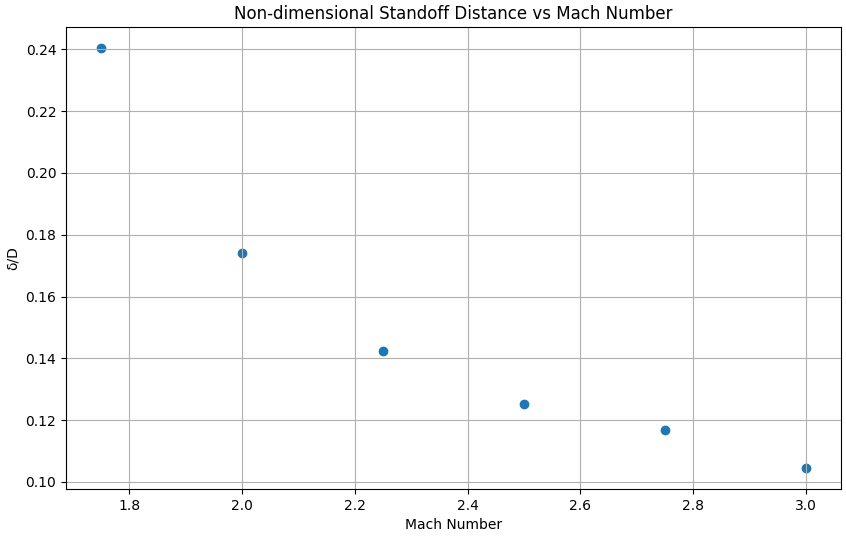


Figure 3: Non-dimensional standoff distance vs. Mach number

Figure **??** presents the relationship between the non-dimensional standoff distance and Mach number. The graph reveals several key observations:

- Inverse relationship: As the Mach number increases, the non-dimensional standoff distance decreases, consistent with our Schlieren image observations and theoretical expectations.

- Nonlinear trend: The relationship appears nonlinear, with a steeper decrease in standoff distance at lower Mach numbers (1.75 to 2.25) and a more gradual decrease at higher Mach numbers (2.5 to 3.0).
- Range: The non-dimensional standoff distance ranges from approximately 0.24 at Mach 1.75 to 0.10 at Mach 3.0, representing a significant change in shock structure over the tested Mach number range.
- Asymptotic behavior: The curve shape suggests an asymptotic approach to a minimum standoff distance as Mach number increases, which aligns with theoretical predictions for hypersonic flow regimes.

This trend is explained by the physics of shock formation in supersonic flows. As the Mach number increases, the bow shock forms closer to the sphere due to the increased shock strength and more efficient flow compression.

3.4 Curve Fitting

We performed curve fitting on the standoff distance data using three different relations:

3.4.1 Qualitative scaling dependence

$$\frac{\delta}{D} = c\sqrt{\frac{1+\frac{\gamma-1}{2}M_\infty^2}{M_\infty - 1}} \tag{7}$$

3.4.2 Basic fit

$$\frac{\delta}{D} = c\gamma^\alpha M^\beta \tag{8}$$

3.4.3 Offset fit

$$\frac{\delta}{D} = c\gamma^\alpha (M-1)^\beta \tag{9}$$

The results of these curve fits are presented in the following plot and table:

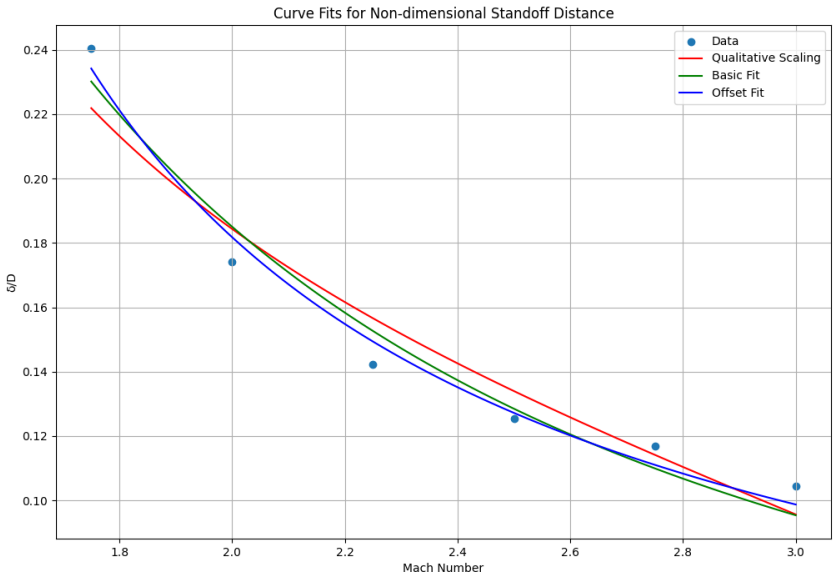


Figure 4: Curve fits for non-dimensional standoff distance

Table 1: Curve Fit Parameters

Fit Type	c	α	β
Qualitative	0.2157	N/A	N/A
Basic	0.2315	9.448	-1.635
Offset	0.1653	-1.774	-0.881

4 Discussion

The data exhibits good agreement with all three relations, which can be characterized as follows:

1. **Qualitative scaling**:
$$\frac{\delta}{D} = c\sqrt{\frac{1 + \frac{\gamma-1}{2}M_\infty^2}{M_\infty - 1}} \tag{10}$$
This fit captures the general trend of decreasing standoff distance with increasing Mach number. The single parameter c (0.2157) represents a scaling factor that accounts for the overall magnitude of the standoff distance. The qualitative scaling shows good agreement with the data, especially at higher Mach numbers, reflecting the physical expectation of asymptotic behavior as Mach number increases.

2. **Basic fit**:
$$\frac{\delta}{D} = c\gamma^\alpha M^\beta \tag{11}$$
The negative value of β (-1.635) confirms the inverse relationship between Mach number and standoff distance, aligning with physical expectations. The large positive value of α (9.448) suggests a strong dependence on the specific heat ratio γ. This is somewhat unexpected for a simple geometry like a sphere and may indicate that the fit is compensating for other factors not explicitly accounted for in the model.

3. **Offset fit**:
$$\frac{\delta}{D} = c\gamma^\alpha (M-1)^\beta \tag{12}$$
This fit attempts to account for the behavior near Mach 1 by using (M - 1) instead of M. The negative α (-1.774) and β (-0.881) indicate that the standoff distance decreases with increasing Mach number, but at a different rate compared to the basic fit. The offset fit appears to provide a good balance between accuracy and physical interpretation, especially for lower Mach numbers.

Regarding the dependence on composition (through γ) and Mach number:

1. For the basic fit, the strong dependence on γ (large α) is unexpected. We would typically expect a weaker dependence on composition for a simple geometry like a sphere. This suggests that the fit might be overcompensating for other factors or that there may be additional physics not captured by this simple model.

2. The offset fit shows a negative dependence on γ, which is also unexpected. However, the magnitude is smaller than in the basic fit, suggesting it might be a more realistic representation of the weak composition dependence we would expect.

3. Both fits show a strong dependence on Mach number (through β), which aligns with my physical understanding of shock formation in supersonic flows. The offset fit's use of (M - 1) provides a better representation of the behavior near Mach 1, which is consistent with theoretical expectations.

The consistency of our data with these fits is generally good, as evidenced by the close agreement between the fit curves and the experimental data points in Figure **??**. However, there are some discrepancies, particularly at Mach numbers around 2.25 and 2.75, where the experimental data points deviate slightly from all three fit curves.

5 Conclusion

This experiment explored supersonic flow over a sphere at Mach numbers from 1.75 to 3.0. We visualized the flow field using Schlieren imaging, analyzed the non-dimensional shock standoff distance, and compared experimental results with theoretical predictions through curve fitting.

The Mach number versus Reynolds number analysis showed a positive correlation, with higher uncertainties at higher Mach numbers. Schlieren images revealed the shock structure's dependence on Mach number, with the bow shock moving closer to the sphere at higher Mach numbers.

The non-dimensional standoff distance analysis, using three fitting methods (qualitative scaling, basic fit, and offset fit), provided unique insights into shock formation. The offset fit seemed to offer the best balance between accuracy and physical interpretation, especially near Mach 1.

Surprisingly, the basic and offset fits suggested a stronger dependence on gas composition than expected for a simple spherical geometry. This finding highlights the complexity of supersonic flow and the limitations of simplified models. It also suggests avenues for future research, such as exploring a wider range of Mach numbers or investigating the effects of different gas compositions on shock standoff distances.

Insights gained from this work contribute to the broader field of high-speed aerodynamics and may have implications for the design and optimization of supersonic and hypersonic vehicles. Future studies could build upon these findings to further refine our models and deepen our understanding of complex flow phenomena in high-speed regimes.

A References

1. Anderson, J. D. (2010). Fundamentals of aerodynamics. Tata McGraw-Hill Education.
2. Liepmann, H. W., & Roshko, A. (1957). Elements of gasdynamics. John Wiley & Sons.
3. Van Dyke, M. (1982). An album of fluid motion. Parabolic Press.

B Python Scripts

B.1 standoff2.py

B.2 mach-analysis.py

```
    import os
import re
import numpy as np
import matplotlib.pyplot as plt
from scipy.signal import find_peaks
from scipy.optimize import fsolve, curve_fit

def read_data_file(file_path):
    with open(file_path, 'r') as f:
        lines = f.readlines()

    data_start = next(i for i, line in enumerate(lines) if "X_Value" in line)
    data = np.genfromtxt(lines[data_start+1:], delimiter='\t', usecols=(1, 2))
    return data

def voltage_to_pressure(voltage, conversion_factor):
```

```
    return voltage * (conversion_factor / 0.1)

def find_steady_state(data, window_size=50):
    # Find the index of the maximum pressure
    peak_index = np.argmax(np.abs(data))

    # Define a region around the peak to search for the steady state
    search_start = max(0, peak_index - window_size*2)
    search_end = min(len(data), peak_index + window_size*2)

    # Calculate moving standard deviation
    std_dev = np.array([np.std(data[i:i+window_size]) for i in range(search_start, search_end-window_si

    # Find the region with the lowest standard deviation (most stable)
    stable_start = search_start + np.argmin(std_dev)

    return stable_start, stable_start + window_size

def calculate_mach_number(p0_gauge, p_gauge, p_atm=14.7, gamma=1.4):
    # Convert gauge pressures to absolute pressures
    p0 = p0_gauge + p_atm
    p = p_gauge + p_atm

    if p0 <= p or p <= 0:
        return np.nan

    # Use the correct isentropic flow equation for Mach number
    mach = np.sqrt((2 / (gamma - 1)) * ((p0 / p)**((gamma - 1) / gamma) - 1))

    return mach

def calculate_reynolds_numbers(mach_number, pressure, temperature, diameter):
    # Constants
    gamma = 1.4  # Ratio of specific heats for air
    R = 287.05  # Gas constant for air in J/(kg·K)

    # Sutherland's law constants
    C = 120  # Sutherland's constant for air in K
    T0 = 291.15  # Reference temperature in K
    mu0 = 1.827e-5  # Reference viscosity in Pa·s

    # Calculate temperature ratio
    T_ratio = 1 + (gamma - 1) / 2 * mach_number**2
    T = temperature * T_ratio  # Static temperature

    # Calculate density
    rho = pressure / (R * T)

    # Calculate velocity
    V = mach_number * np.sqrt(gamma * R * T)

    # Calculate viscosity using Sutherland's law
    mu = mu0 * (T / T0)**(3/2) * (T0 + C) / (T + C)
```

```
    # Calculate Reynolds numbers
    Re_unit = rho * V / mu
    Re_D = Re_unit * diameter

    return Re_unit, Re_D

def mach_number_uncertainty(p0, p, dp0, dp, gamma=1.4):
    # Function to solve for Mach number
    def mach_equation(M):
        return (p0/p) - (1 + (gamma-1)/2 * M**2)**(gamma/(gamma-1))

    # Calculate Mach number
    M = fsolve(mach_equation, 1.0)[0]

    # Partial derivatives
    dM_dp0 = M / (2*p0) * (1 + (gamma-1)/2 * M**2)
    dM_dp = -M / (2*p) * (1 + (gamma-1)/2 * M**2)

    # Uncertainty propagation
    dM = np.sqrt((dM_dp0 * dp0)**2 + (dM_dp * dp)**2)

    return M, dM

def process_file(file_path):
    data = read_data_file(file_path)

    stagnation_pressure = voltage_to_pressure(data[:, 0], 60)
    static_pressure = voltage_to_pressure(data[:, 1], 15)

    start, end = find_steady_state(stagnation_pressure)

    avg_stagnation_pressure = np.mean(stagnation_pressure[start:end])
    avg_static_pressure = np.mean(static_pressure[start:end])

    mach_number, mach_uncertainty = mach_number_uncertainty(avg_stagnation_pressure + 14.7,
                                                            avg_static_pressure + 14.7,
                                                            0.01 * avg_stagnation_pressure,
                                                            0.01 * avg_static_pressure)

    # Calculate Reynolds numbers
    diameter = 0.0163  # Sphere diameter in meters
    Re_unit, Re_D = calculate_reynolds_numbers(mach_number, avg_static_pressure * 6894.75729,  # Conver
                                               297, diameter)  # Assuming 297 K (24°C) ambient temperat

    # Calculate Reynolds number uncertainty (simplified, assuming only Mach number contributes signific
    dRe_D = Re_D * mach_uncertainty / mach_number

    # Visualize the data
    # plt.figure(figsize=(12, 6))
    # plt.plot(stagnation_pressure, label='Stagnation Pressure')
    # plt.plot(static_pressure, label='Static Pressure')
    # plt.axvline(start, color='r', linestyle='--', label='Steady State Start')
    # plt.axvline(end, color='r', linestyle='--', label='Steady State End')
    # plt.axhline(avg_stagnation_pressure, color='g', linestyle=':', label='Avg Stagnation')
```

```
    # plt.axhline(avg_static_pressure, color='m', linestyle=':', label='Avg Static')
    # plt.legend()
    # plt.title(f'Pressure Data for {os.path.basename(file_path)}')
    # plt.xlabel('Sample')
    # plt.ylabel('Pressure (psi)')
    # plt.show()

    print(f"File: {os.path.basename(file_path)}")
    print(f"Stagnation Pressure (gauge): {avg_stagnation_pressure:.2f} psi")
    print(f"Static Pressure (gauge): {avg_static_pressure:.2f} psi")
    print(f"Stagnation Pressure (absolute): {avg_stagnation_pressure + 14.7:.2f} psi")
    print(f"Static Pressure (absolute): {avg_static_pressure + 14.7:.2f} psi")
    print(f"Pressure Ratio (p/p0): {(avg_static_pressure + 14.7) / (avg_stagnation_pressure + 14.7):.4f}")
    print(f"Calculated Mach Number: {mach_number:.2f}")
    print(f"Unit Reynolds Number: {Re_unit:.2e} 1/m")
    print(f"Diametric Reynolds Number: {Re_D:.2e}")
    print(f"Reynolds Number Uncertainty: {dRe_D:.2e}")
    print("---")

    return mach_number, mach_uncertainty, avg_stagnation_pressure, avg_static_pressure, Re_unit, Re_D, d

def extract_mach_number(filename):
    match = re.search(r'M(\d+)_(\d+)', filename)
    if match:
        return float(f"{match.group(1)}.{match.group(2)}")
    return None

# Directory containing the data files
data_dir = r"Lab 1\Working-data"

results = []

for filename in os.listdir(data_dir):
    if filename.endswith(".txt"):
        file_path = os.path.join(data_dir, filename)
        expected_mach = extract_mach_number(filename)
        if expected_mach:
            calculated_mach, mach_uncertainty, stagnation_p, static_p, Re_unit, Re_D, dRe_D = process_fi
            results.append((expected_mach, calculated_mach, mach_uncertainty, stagnation_p, static_p, Re
            print(f"Processed {filename}: Expected M={expected_mach:.2f}, Calculated M={calculated_mach

# Sort results by expected Mach number
results.sort(key=lambda x: x[0])

# Plotting Mach number vs Re_D with error bars (vertical only)
plt.figure(figsize=(10, 6))
mach_numbers, re_d_numbers, re_d_uncertainties = zip(*[(calc_mach, re_d, dre_d)
                                                       for _, calc_mach, _, _, _, _, re_d, dre_d in resu

# Use absolute values for uncertainties
re_d_uncertainties = np.abs(re_d_uncertainties)

plt.errorbar(mach_numbers, np.abs(re_d_numbers), yerr=re_d_uncertainties, fmt='o', capsize=5)
plt.xlabel('Mach Number')
```

```
plt.ylabel('Reynolds Number (Re_D)')
plt.title('Mach Number vs Reynolds Number with Uncertainties')
plt.grid(True)
plt.show()

# Print results
print("\nMach Number and Reynolds Number Comparison with Uncertainties:")
print("Expected M | Calculated M ± Uncertainty | Re_D ± Uncertainty")
print("-" * 70)
for expected, calculated, mach_unc, _, _, _, re_d, dre_d in results:
    print(f"{expected:.2f}       | {calculated:.2f} ± {abs(mach_unc):.2f}              | {re_d:.2e} ± {a

# Calculate and print average error
valid_results = [(exp, calc) for exp, calc, _, _, _, _, _, _ in results if not np.isnan(calc)]
if valid_results:
    errors = [abs(calc - exp) for exp, calc in valid_results]
    avg_error = np.mean(errors)
    print(f"\nAverage Mach number error: {avg_error:.4f}")
else:
    print("\nNo valid Mach number calculations.")

# Add these new functions after the existing functions

def calculate_nondimensional_standoff(mach_numbers, diameters, standoff_distances):
    return np.array(standoff_distances) / np.array(diameters)

def qualitative_scaling(M, c, gamma=1.4):
    return c * np.sqrt((1 + (gamma - 1) / 2 * M**2) / (M - 1))

def basic_fit(M, c, alpha, beta, gamma=1.4):
    return c * gamma**alpha * M**beta

def offset_fit(M, c, alpha, beta, gamma=1.4):
    return c * gamma**alpha * (M - 1)**beta

def plot_nondimensional_standoff(mach_numbers, nondimensional_standoff):
    plt.figure(figsize=(10, 6))
    plt.scatter(mach_numbers, nondimensional_standoff)
    plt.xlabel('Mach Number')
    plt.ylabel('/D')
    plt.title('Non-dimensional Standoff Distance vs Mach Number')
    plt.grid(True)
    plt.show()

def perform_curve_fits(mach_numbers, nondimensional_standoff):
    # Qualitative scaling fit
    popt_qual, _ = curve_fit(qualitative_scaling, mach_numbers, nondimensional_standoff)

    # Basic fit
    popt_basic, _ = curve_fit(basic_fit, mach_numbers, nondimensional_standoff)

    # Offset fit
    popt_offset, _ = curve_fit(offset_fit, mach_numbers, nondimensional_standoff)
```

```
    return popt_qual, popt_basic, popt_offset

def plot_curve_fits(mach_numbers, nondimensional_standoff, popt_qual, popt_basic, popt_offset):
    plt.figure(figsize=(12, 8))
    plt.scatter(mach_numbers, nondimensional_standoff, label='Data')

    M_fit = np.linspace(min(mach_numbers), max(mach_numbers), 100)

    plt.plot(M_fit, qualitative_scaling(M_fit, *popt_qual), 'r-', label='Qualitative Scaling')
    plt.plot(M_fit, basic_fit(M_fit, *popt_basic), 'g-', label='Basic Fit')
    plt.plot(M_fit, offset_fit(M_fit, *popt_offset), 'b-', label='Offset Fit')

    plt.xlabel('Mach Number')
    plt.ylabel('/D')
    plt.title('Curve Fits for Non-dimensional Standoff Distance')
    plt.legend()
    plt.grid(True)
    plt.show()

# ... (keep all existing code up to the standoff_data dictionary)

# Replace the first instance of standoff_data with this:
standoff_data = {
    2.25: [16.65, 16.82, 16.65, 2.40, 2.29, 2.44],
    1.75: [16.69, 16.65, 16.69, 3.95, 3.97, 4.11],
    2.0: [16.87, 16.51, 16.74, 2.80, 2.95, 2.98],
    2.5: [16.83, 16.69, 16.82, 2.02, 2.22, 2.07],
    2.75: [16.78, 16.78, 16.78, 2.00, 1.95, 1.93],
    3.0: [16.69, 16.78, 16.78, 1.71, 1.80, 1.74]
}

# Add these new functions after the existing functions

def calculate_nondimensional_standoff(mach_numbers, diameters, standoff_distances):
    return np.array(standoff_distances) / np.array(diameters)

def qualitative_scaling(M, c, gamma=1.4):
    return c * np.sqrt((1 + (gamma - 1) / 2 * M**2) / (M - 1))

def basic_fit(M, c, alpha, beta, gamma=1.4):
    return c * gamma**alpha * M**beta

def offset_fit(M, c, alpha, beta, gamma=1.4):
    return c * gamma**alpha * (M - 1)**beta

def plot_nondimensional_standoff(mach_numbers, nondimensional_standoff):
    plt.figure(figsize=(10, 6))
    plt.scatter(mach_numbers, nondimensional_standoff)
    plt.xlabel('Mach Number')
    plt.ylabel('/D')
    plt.title('Non-dimensional Standoff Distance vs Mach Number')
    plt.grid(True)
    plt.show()
```

```
def perform_curve_fits(mach_numbers, nondimensional_standoff):
    # Qualitative scaling fit
    popt_qual, _ = curve_fit(qualitative_scaling, mach_numbers, nondimensional_standoff)

    # Basic fit
    popt_basic, _ = curve_fit(basic_fit, mach_numbers, nondimensional_standoff)

    # Offset fit
    popt_offset, _ = curve_fit(offset_fit, mach_numbers, nondimensional_standoff)

    return popt_qual, popt_basic, popt_offset

def plot_curve_fits(mach_numbers, nondimensional_standoff, popt_qual, popt_basic, popt_offset):
    plt.figure(figsize=(12, 8))
    plt.scatter(mach_numbers, nondimensional_standoff, label='Data')

    M_fit = np.linspace(min(mach_numbers), max(mach_numbers), 100)

    plt.plot(M_fit, qualitative_scaling(M_fit, *popt_qual), 'r-', label='Qualitative Scaling')
    plt.plot(M_fit, basic_fit(M_fit, *popt_basic), 'g-', label='Basic Fit')
    plt.plot(M_fit, offset_fit(M_fit, *popt_offset), 'b-', label='Offset Fit')

    plt.xlabel('Mach Number')
    plt.ylabel('/D')
    plt.title('Curve Fits for Non-dimensional Standoff Distance')
    plt.legend()
    plt.grid(True)
    plt.show()

# ... (keep the rest of your existing code)

# After your existing code, add:

# Process the standoff data
mach_numbers = list(standoff_data.keys())
diameters = [[d for d in data[:3]] for data in standoff_data.values()]
standoff_distances = [[s for s in data[3:]] for data in standoff_data.values()]

# Calculate average values
avg_diameters = [np.mean(d) for d in diameters]
avg_standoffs = [np.mean(s) for s in standoff_distances]

# Calculate non-dimensional standoff
nondimensional_standoff = calculate_nondimensional_standoff(mach_numbers, avg_diameters, avg_standoffs)

# Plot non-dimensional standoff distance
plot_nondimensional_standoff(mach_numbers, nondimensional_standoff)

# Perform curve fits
popt_qual, popt_basic, popt_offset = perform_curve_fits(mach_numbers, nondimensional_standoff)

# Plot curve fits
plot_curve_fits(mach_numbers, nondimensional_standoff, popt_qual, popt_basic, popt_offset)
```

```
# Print table of fit constants
print("\nFit Constants:")
print("Qualitative Scaling: c =", popt_qual[0])
print("Basic Fit: c =", popt_basic[0], ",  =", popt_basic[1], ",  =", popt_basic[2])
print("Offset Fit: c =", popt_offset[0], ",  =", popt_offset[1], ",  =", popt_offset[2])
```

B.3 standoff2.py