Wind Tunnel Free Stream Turbulence Measurement Using a Turbulence Sphere

Parham Khodadi*
A E 303, Section 3, with Dr. Xiaofeng Liu

This experiment investigates the aerodynamic characteristics of a NACA 43012A airfoil using wind tunnel testing and computational tools. Coefficients of lift, drag, and moment were obtained across a range of angles of attack and compared with theoretical and numerical predictions. Experimental results validated key aerodynamic trends while highlighting the limitations of surface pressure and wake-based measurement techniques. The study emphasizes the practical challenges of empirical data collection and the value of comparative methods in aerodynamic analysis.

I. Nomenclature

AoA OR α	=	Angle of Attack (degrees or radians)
C_p	=	Pressure coefficient (~)
C_l	=	Lift Coefficient (~)
C_d	=	Drag Coefficient (~)
C_a	=	Axial Force Coefficient (~)
C_n	=	Normal Force Coefficient (~)
$C_{m_{LE}}$	=	Pitching Moment Coefficient with respect to the Leading Edge (~)
$C_{m_{ac}}$	=	Pitching Moment Coefficient with respect to the Aerodynamic Center (~)
$\frac{x}{c}$	=	Unit Distance Along the Chord Line (~)
$\frac{y}{c}$	=	Unit Distance Perpendicular to Chord Line (~)
c	=	Chord Length (in.)

II. Introduction

Understanding the aerodynamic behavior of airfoils is critical in aerospace design, as they directly influence lift, drag, and pitching moment characteristics. This experiment investigates the NACA 43012A airfoil using both experimental and computational methods to analyze its performance at various angles of attack. Surface pressure data collected from wind tunnel testing is used to compute aerodynamic coefficients and identify flow separation, while wake survey data provides an independent method of estimating drag. These results are compared with theoretical data from NACA Report 610 [1], as well as computational predictions generated using XFOIL [**?**], a panel method solver developed for low Reynolds number flows. The experiment reinforces core concepts in airfoil aerodynamics and validates theoretical tools against empirical data [2, 3].

III. Theory

In this experiment, two primary methods were used to evaluate the aerodynamic performance of a NACA 43012A airfoil: surface pressure measurement and wake survey. Each method relies on different physical principles and provides a complementary way to estimate lift, drag, and pitching moment coefficients.

*Aerospace Engineering student, San Diego State University

A. Surface Pressure Method

The surface pressure method involves measuring the pressure at discrete ports on the upper and lower surfaces of the airfoil. These pressures are converted into the non-dimensional pressure coefficient:

$$C_p = \frac{p - p_\infty}{q_\infty} \tag{1}$$

where p is the local pressure, p_∞ is the freestream static pressure, and $q_\infty = \frac{1}{2}\rho_\infty V_\infty^2$ is the dynamic pressure [2].

From the distribution of C_p, the normal (C_n) and axial (C_a) force coefficients are computed through numerical integration. These are then used to compute the lift and drag coefficients via:

$$C_L = C_n \cos(\alpha) - C_a \sin(\alpha) \tag{2}$$

$$C_D = C_n \sin(\alpha) + C_a \cos(\alpha) \tag{3}$$

where α is the angle of attack [3].

The pitching moment about the leading edge is calculated using:

$$C_{m,LE} = \int_0^1 (C_{p,l} - C_{p,u}) x \, dx \tag{4}$$

and then shifted to the aerodynamic center (typically at $x/c = 0.238$) using:

$$C_{m,ac} = C_{m,LE} + f C_L x_{ac} \cos(\alpha) - C_D y_{ac} \cos(\alpha) + C_L y_{ac} \sin(\alpha) + C_D x_{ac} \sin(\alpha) \tag{5}$$

where x_{ac} and y_{ac} are the coordinates of the aerodynamic center [4].

B. Wake Survey Method

The wake survey provides a more direct estimate of drag by measuring the momentum deficit in the wake downstream of the airfoil. A multi-port Pitot rake records the local total pressures behind the airfoil, from which the velocity profile is reconstructed using Bernoulli's equation:

$$u = \sqrt{\frac{2(p_t - p)}{\rho}} \tag{6}$$

where p_t is the total pressure, p is the local static pressure, and ρ is the air density [3].

The drag per unit span is then estimated from the momentum loss:

$$D = \int \rho u (V_\infty - u) \, dy \tag{7}$$

which is converted to a drag coefficient via:

$$C_D = \frac{D}{q_\infty} \tag{8}$$

C. Comparison and Flow Behavior

Comparing the drag calculated via both methods helps validate experimental accuracy and understand contributions from pressure drag and skin friction. The surface pressure method typically underestimates total drag as it ignores skin friction, while the wake survey captures all profile drag. Additionally, by analyzing the C_p distribution across angles of attack, flow separation regions can be identified, providing insight into stall behavior and aerodynamic efficiency of the airfoil [1, 5].

IV. Experimental Setup

The experiment was conducted in the closed-circuit subsonic wind tunnel facility at San Diego State University. The test article used was a 2-D NACA 43012A airfoil with a 12-inch chord and 24-inch span. The airfoil was instrumented with 32 pressure ports (Ports 1–32) to measure pressure distributions on both the upper and lower surfaces. These ports were connected to a ScaniValve pressure transducer to convert pressure data into digital format for analysis.

A. Wake Rake Configuration

A wake rake, consisting of 20 brass pitot tubes (Ports 33–52), was mounted downstream of the airfoil to measure velocity profiles in the wake region. The wake rake was positioned based on the angle of attack and was rotated 17° clockwise from the vertical to align with the wake flow at -5° AoA. It rotated and translated horizontally as the AoA increased to capture wake development accurately.

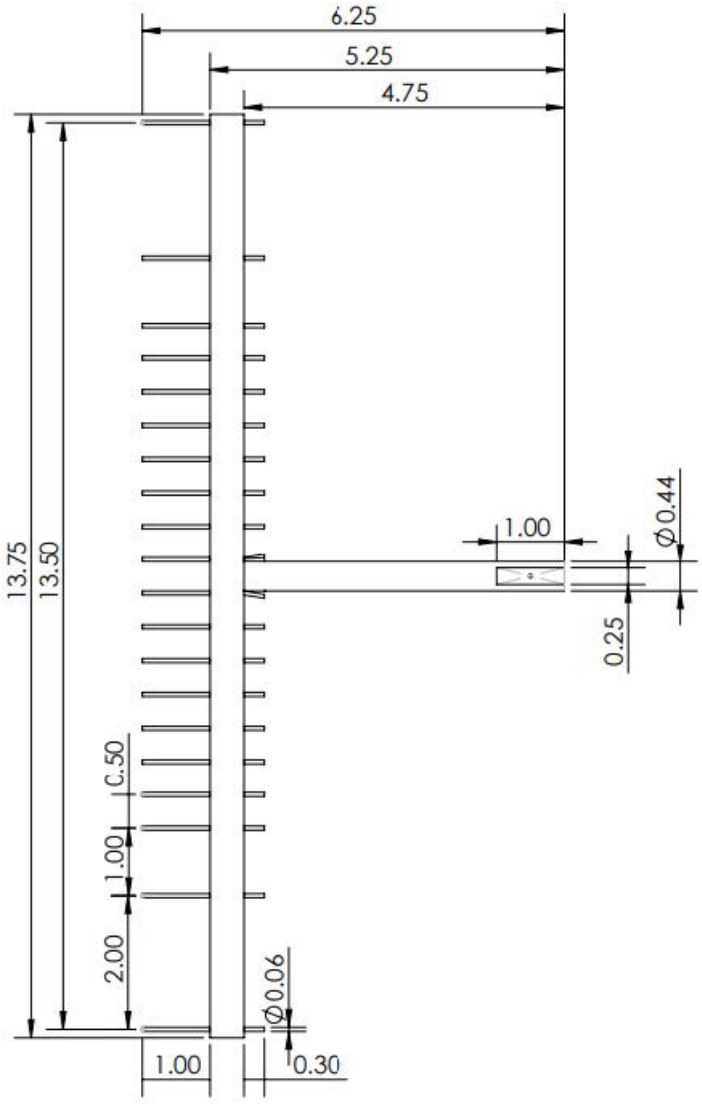

Fig. 1 Wake rake configuration with port dimensions (units in inches).

B. Measurement System

The pressure ports were monitored using the ScaniValve system, which records pressures from both the airfoil and wake rake. Due to hardware limitations, only 47 ports could be connected to the transducer at once, requiring some measurements to be recorded using a water manometer. The manometer data served as backup validation.

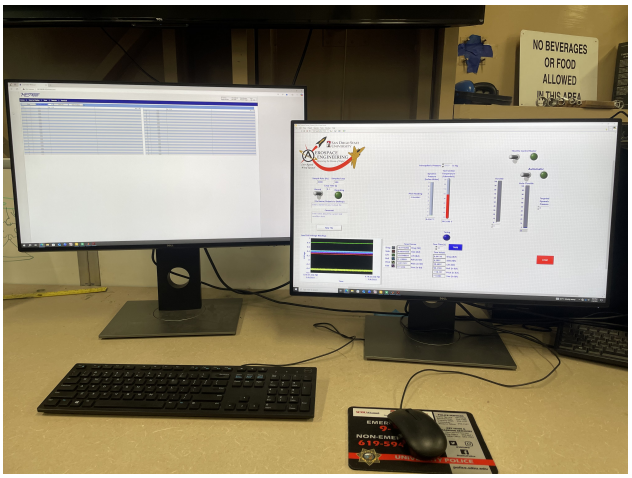

Fig. 2 SDSU wind tunnel computer and software.

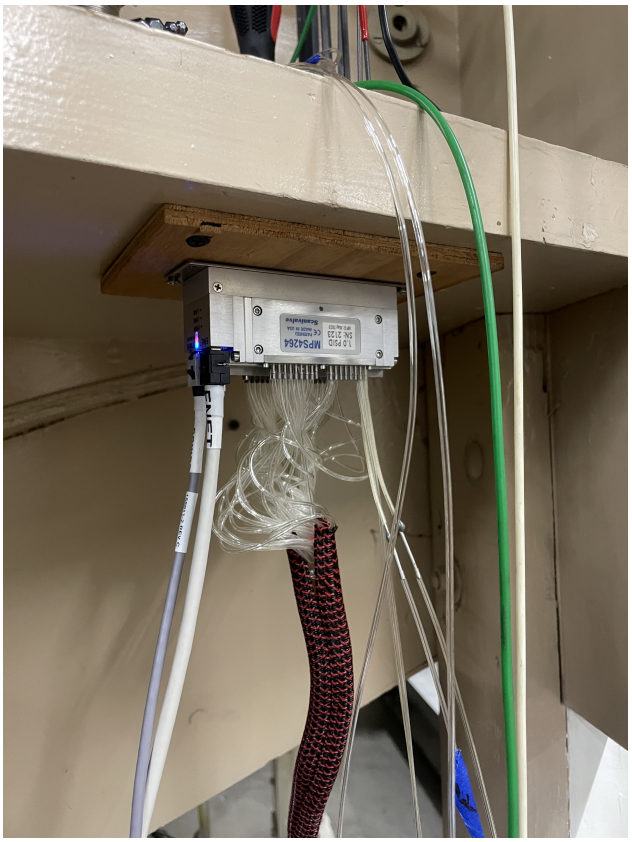

Fig. 3 ScaniValve Pressure Transducer.

Fig. 4 ScaniValve hardware tube connection.

Fig. 5 ScaniValve PDM 1500.

C. Test Procedure

The test spanned seven different angle-of-attack configurations: -5°, 0°, 5°, 10°, 15°, 20°, and 20° (with different wake rake positions). These positions were selected using the Eason 900X control panel, which also adjusted the airfoil AoA and wake rake orientation automatically. An inclinometer ensured correct wake rake alignment before data acquisition.

Fig. 6 Airfoil mounted vertically in test section with wake rake behind trailing edge.

D. Data Acquisition

Each test captured 4000 pressure samples per port over a fixed period. Static pressure, total pressure, and internal temperature were also recorded to compute air density and other relevant flow properties. The ambient pressure was set to 30.11 inHg and temperature to 79.5°F during the experiment.

The system assumed incompressible, steady flow with no pressure loss between ports and transducers. Port 33, the lowest wake rake port, was located outside the wake and used as the reference for freestream static pressure.

V. Experimental Procedure

The experiment was performed in the SDSU Subsonic Wind Tunnel as outlined in the lab documentation [4]. A NACA 43012A airfoil [5] with a 12-inch chord and 32-inch span was mounted horizontally inside the test section. The angle of attack, α, was varied from $-5°$ to $20°$ using a calibrated rotary disk. Barometric pressure and ambient temperature were recorded before each test to determine air density, which was computed using the ideal gas law:

$$\rho = \frac{P_{\text{amb}}}{R \cdot T} \tag{9}$$

where P_{amb} is the ambient pressure, R is the specific gas constant for air, and T is the absolute temperature.

Surface pressure data was collected using a Scanivalve system connected to 32 ports (16 upper, 16 lower) along the airfoil's surface. Each pressure tap's data was digitized using a LabVIEW-integrated analog-to-digital board at 18 Hz for 4000 samples per run. The pressure coefficient at each port was calculated using:

$$C_p = \frac{p - p_\infty}{q_\infty} \tag{10}$$

where p is the local pressure and q_∞ is the freestream dynamic pressure, given by:

$$q_\infty = \frac{1}{2}\rho u_\infty^2 \tag{11}$$

To calculate lift and drag via surface pressure measurements, the pressure coefficient distributions were integrated

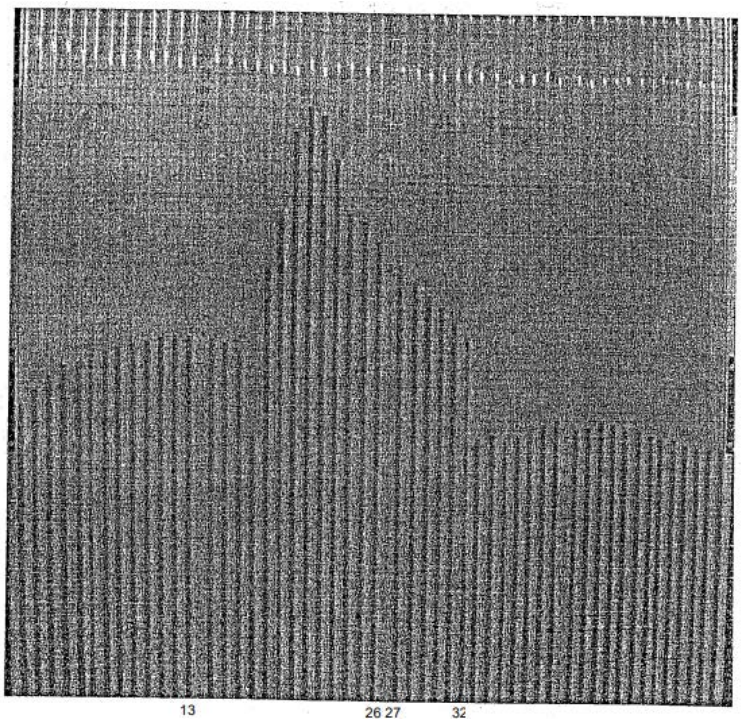

Fig. 7 Typical manometer reading at AoA = 5° with all tubes connected.

along the airfoil chord using numerical methods [6]. The lift and drag coefficients were computed as:

$$C_L = C_n \cos\alpha - C_a \sin\alpha \tag{12}$$

$$C_D = C_n \sin\alpha + C_a \cos\alpha \tag{13}$$

where C_n and C_a represent the normal and axial force coefficients, respectively.

Wake surveys were conducted by positioning a Pitot rake downstream of the airfoil. The rake recorded velocity deficits in the wake by sampling the dynamic pressure at 20 ports, allowing for drag estimation via the momentum deficit method. The profile drag per unit span was computed from:

$$D' = \int \rho u(u_\infty - u)\, dy \tag{14}$$

and converted to a non-dimensional drag coefficient using the previously calculated q_∞.

Reference static and total pressure values were collected through dedicated static and Pitot probes located outside the wake region. These readings ensured accurate baseline conditions for Bernoulli-based velocity estimations:

$$u = \sqrt{\frac{2(p_0 - p)}{\rho}} \tag{15}$$

All data post-processing, including coefficient calculation, curve fitting, and plotting, was performed using MATLAB [7].

VI. Results and Data Reduction

The aerodynamic coefficients of lift (C_L), drag (C_D), and moment about the aerodynamic center ($C_{m,ac}$) were computed from pressure measurements taken along the surface of the NACA 43012A airfoil. The processed data from MATLAB is presented in this section alongside comparison plots using data from XFOIL and NACA Report 610 [1, 5, 8].

A. Lift Coefficient vs. Angle of Attack

The lift coefficient, C_L, was calculated using the pressure difference between the upper and lower surfaces of the airfoil at each angle of attack. The variation of C_L with angle of attack (α) is shown in Figure 8. As expected, C_L increases linearly with α up to 15° before dropping slightly due to the onset of flow separation near stall.

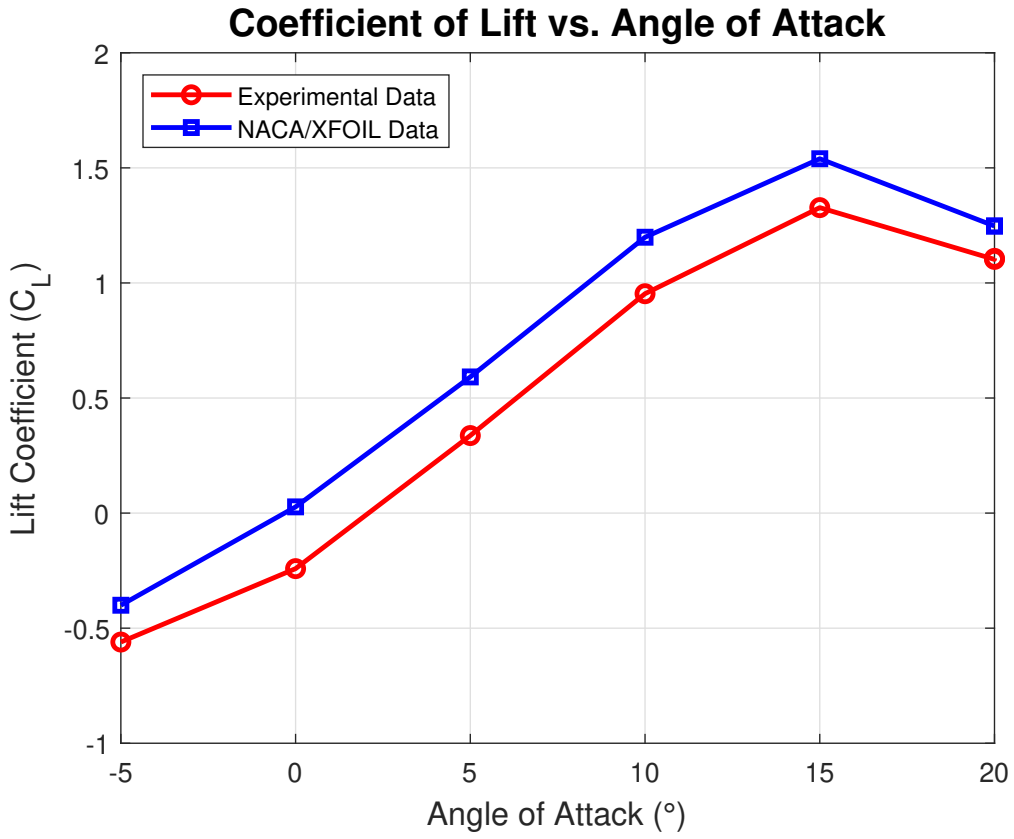

Fig. 8 Coefficient of Lift vs. Angle of Attack

B. Drag Coefficient vs. Angle of Attack

The drag coefficient, C_D, was determined using surface pressure distributions. As seen in Figure 9, C_D increases significantly beyond $\alpha = 10°$, consistent with growing boundary layer separation. This trend also supports the drop in lift observed in the previous section.

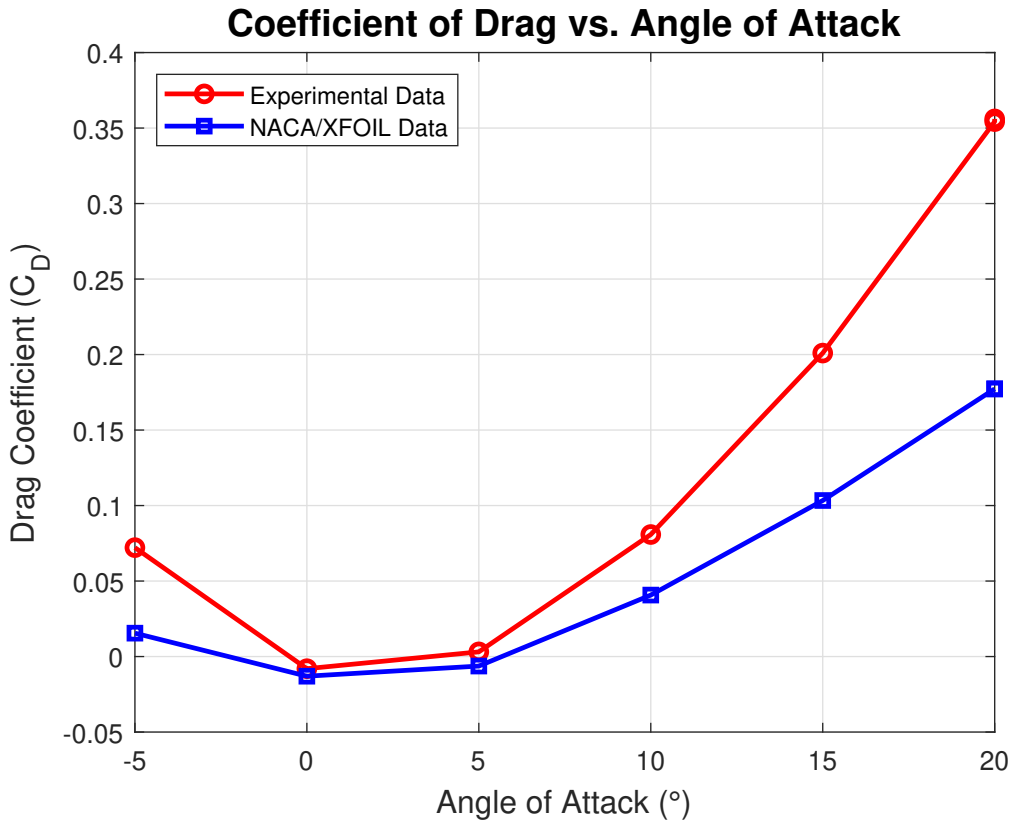

Fig. 9 Coefficient of Drag vs. Angle of Attack

C. Pitching Moment Coefficient vs. Angle of Attack

Figure 10 illustrates the behavior of the moment coefficient about the aerodynamic center, $C_{m,ac}$, with increasing angle of attack. The moment remains relatively constant between $\alpha = 0°$ and $15°$ before shifting rapidly at higher angles, indicating loss of flow attachment and altered pressure distribution around the quarter chord point.

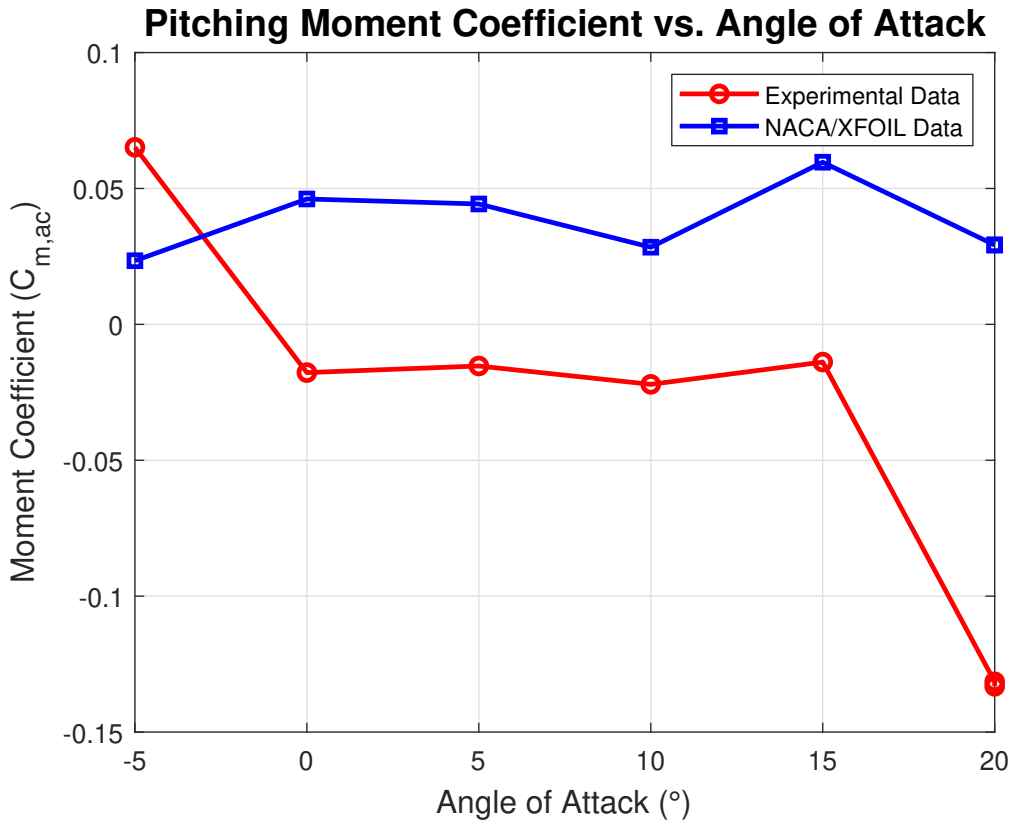

Fig. 10 Pitching Moment Coefficient vs. Angle of Attack

D. Comparison with XFOIL and NACA 610

Figures 8–10 include XFOIL predictions and NACA 610 data for the same airfoil. The experimental lift curve lies slightly below the theoretical data, likely due to wind tunnel wall effects and experimental uncertainties. Drag measurements are higher, especially at high angles, reflecting pressure drag induced by flow separation.

E. Surface Pressure Distribution and Flow Separation

To visualize pressure distribution on the upper surface, Figure 11 presents C_p vs. x/c for various α. As α increases, the suction peak near the leading edge becomes stronger and moves upstream. Beyond 15°, the flattened slope and plateau indicate separation.

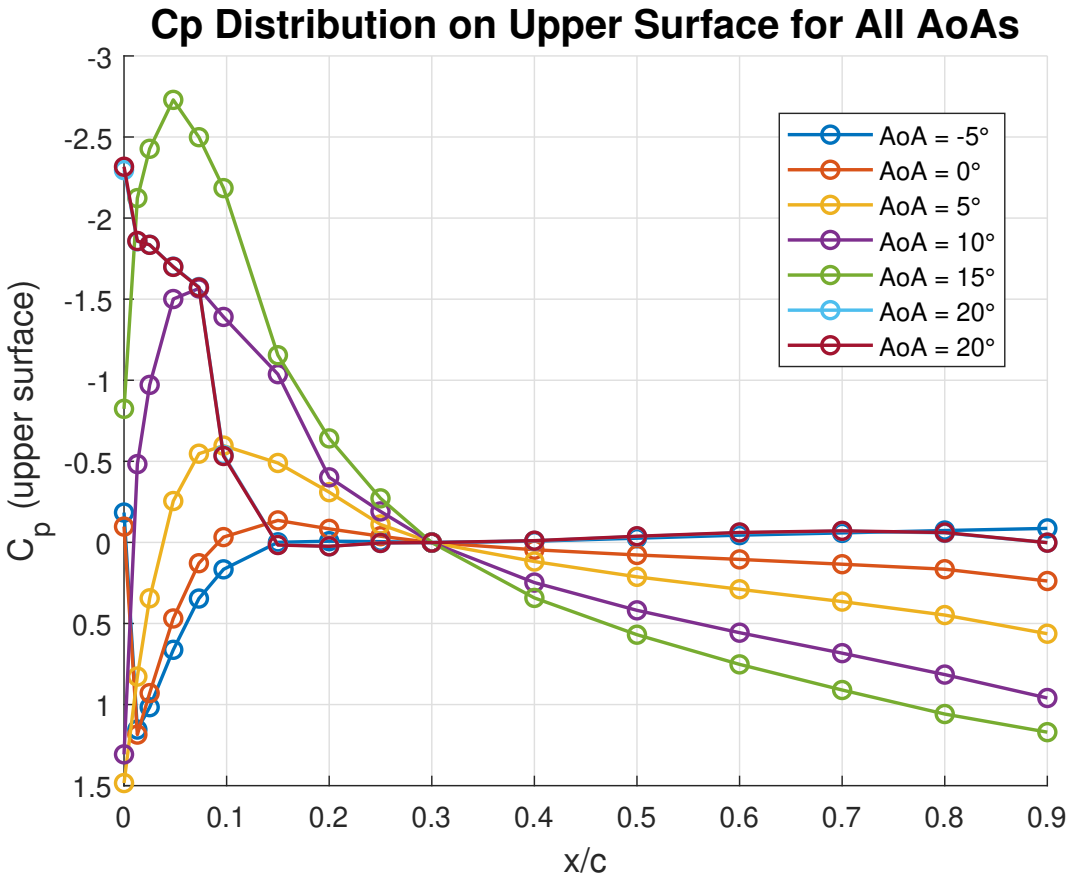

Fig. 11 Pressure Coefficient Distribution on Upper Surface

F. Wake Survey vs. Surface Pressure Drag

A momentum balance method was used to calculate drag from wake velocity deficits, as shown in Figure 12. Wake-based drag results are generally lower than surface-pressure-derived values, especially at mid-range α. This discrepancy is typical and can be attributed to incomplete capture of wake loss or pressure integration assumptions [2].

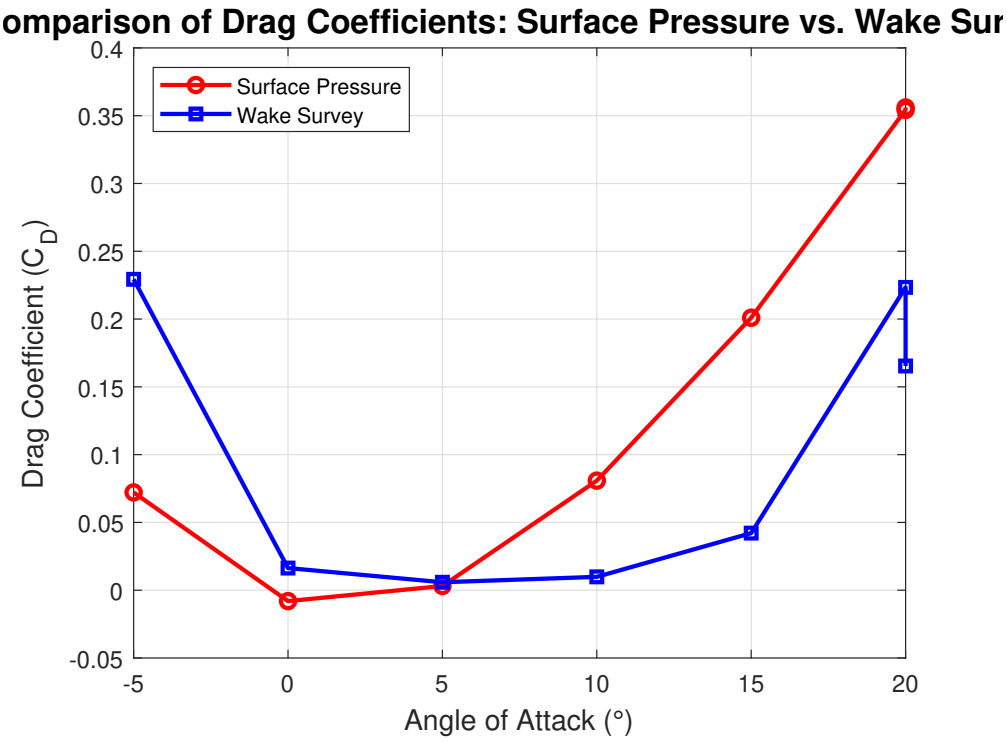

Fig. 12 Comparison of Drag Coefficients: Surface Pressure vs. Wake Survey

G. Wake Velocity Profiles

Finally, wake velocity profiles were computed using Pitot rake data. The velocity deficit behind the airfoil widens with increasing angle of attack, as seen in Figure 13, reflecting increased flow separation and drag.

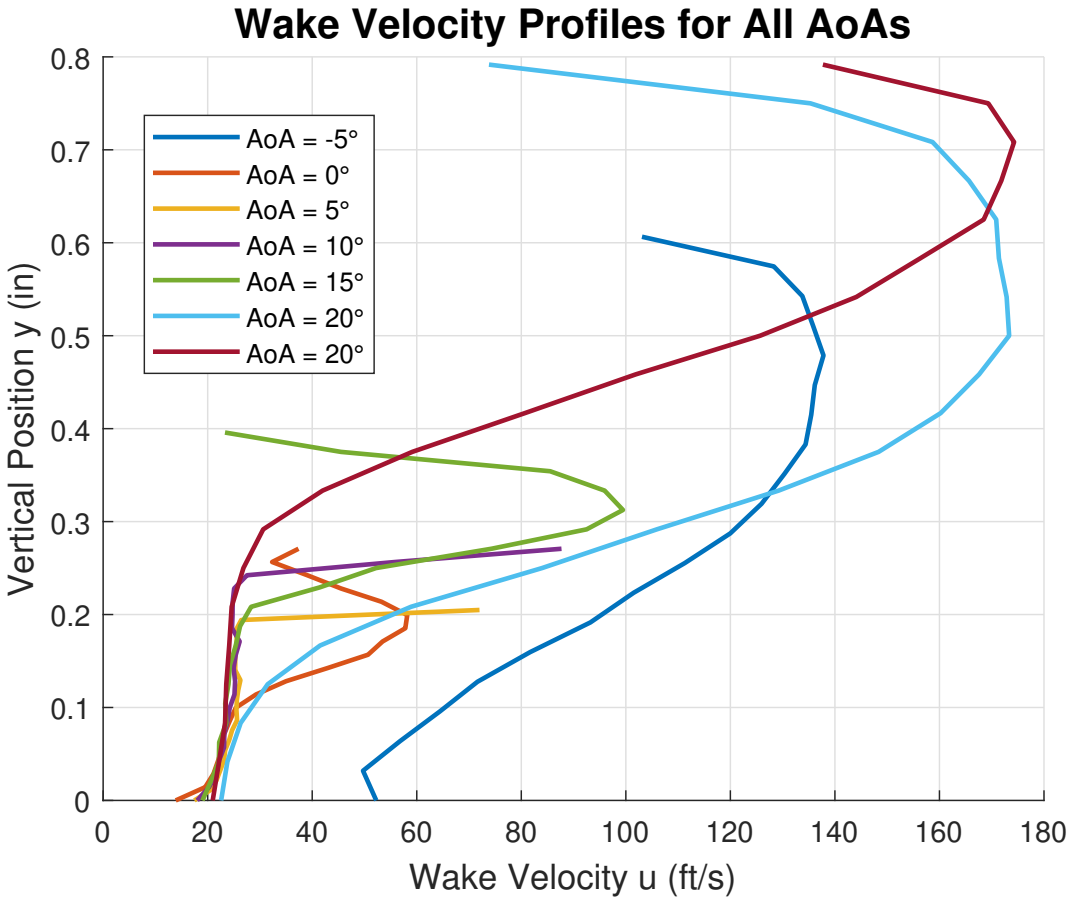

Fig. 13 Wake Velocity Profiles for All AoAs

All numerical calculations and data reduction processes were implemented in MATLAB [7]. Trapezoidal integration was applied for force and moment estimations [6].

VII. Discussion

The experimental results for the NACA 43012A airfoil generally follow expected aerodynamic trends, though several discrepancies with theoretical predictions were observed. The C_L vs. α curve demonstrates the expected linear relationship up to about 15° before decreasing near 20°, consistent with stall behavior [2]. The slope of the linear region closely matches that from XFOIL and the NACA 610 report [1, 5], suggesting that the pressure measurement system and calibration were largely accurate in capturing lift behavior.

However, measured drag coefficients were consistently higher than the theoretical values. This is likely due to the limitations of integrating pressure ports along the surface, which can omit skin friction components and exacerbate form drag due to three-dimensional flow effects or misalignments in the wind tunnel setup. The discrepancy is especially evident beyond $\alpha = 10°$, where flow separation becomes prominent. Furthermore, the drag predicted from the wake survey method was systematically lower than the pressure-based drag, which is typical due to potential underestimation of momentum loss across the wake or imperfect wake rake alignment [2, 4].

The moment coefficient $C_{m,ac}$ remained relatively stable through moderate angles of attack, as expected for symmetric or near-symmetric airfoils with minimal camber. Deviations at high α may reflect shifts in the center of pressure as stall initiates, altering the moment balance about the quarter chord point.

The C_p distributions clearly reveal flow separation. At low angles, a sharp suction peak near the leading edge followed by a gradual pressure recovery is observed, consistent with attached laminar or mildly turbulent flow. As α increases, the suction peak intensifies and recovery becomes flatter, especially around $\alpha = 15°$ and beyond. This plateau behavior is a classic indicator of flow separation on the upper surface [3]. The wake velocity profiles further corroborate

this interpretation—profiles widen and shift with increasing α, indicating momentum deficit due to the separated flow region downstream.

All data processing was conducted using MATLAB [7], and numerical integrations for force and moment coefficients were performed using the trapezoidal rule [6]. While XFOIL [8] provided a useful comparison for inviscid and mildly viscous conditions, it does not fully account for complex separation dynamics, especially under stall.

VIII. Conclusion

This experiment successfully characterized the aerodynamic performance of the NACA 43012A airfoil across a range of angles of attack. Key parameters such as C_L, C_D, and $C_{m,ac}$ were experimentally determined and compared against theoretical data from XFOIL and NACA 610. While the lift results closely matched theoretical expectations, drag and moment coefficients showed increasing deviation near stall due to flow separation. Pressure coefficient distributions and wake velocity profiles confirmed the presence and extent of separation. Overall, the lab validated fundamental aerodynamic principles and illustrated practical limitations in experimental measurements under non-ideal flow conditions.

Acknowledgments

The author would like to thank Dr. Xiaofeng Liu for his guidance and Teacher's Assistant Andrew Balolong for assistance during the experiment.

References

[1] Abbott, I. H., Doenhoff, A. E. V., and Stivers Jr, L. S., "Summary of Airfoil Data," *NACA Report 610*, 1959.

[2] Anderson, J. D., *Fundamentals of Aerodynamics*, 3rd ed., McGraw-Hill, New York, 2001.

[3] Anderson, J. D., *Introduction to Flight*, 6th ed., McGraw-Hill, New York, 2007.

[4] Liu, X., *AE 303 Lab 4 Airfoil Instructions*, 2025. San Diego State University, Aerospace Engineering Department.

[5] Center, N. G. R., "NACA 43012A Airfoil Data Files," , 2024. Accessed via `https://m-selig.ae.illinois.edu/ads/coord_database.html`.

[6] Kreyszig, E., "Advanced Engineering Mathematics," *John Wiley & Sons*, 2011, p. Chapter 20.

[7] The MathWorks, Inc., *MATLAB Documentation*, The MathWorks, 2024. `https://www.mathworks.com/help/matlab/`.

[8] Drela, M., "XFOIL 6.99 User Guide," `https://web.mit.edu/drela/Public/web/xfoil/`, 2020. Accessed: 2025-03-28.

IX. Appendix

A. Python Code for .cp File Conversion

The following code was used in PyCharm IDE to convert the XFOIL '.cp' output files into CSV format compatible with MATLAB:

```python
import os
import pandas as pd

def read_cp_file(filepath):
    """Reads a .cp file from XFOIL and returns a DataFrame with columns x, y, Cp."""
    with open(filepath, 'r') as file:
        lines = file.readlines()[3:]  # Skip first 3 header lines
        data = [list(map(float, line.split())) for line in lines if line.strip()]
    return pd.DataFrame(data, columns=['x', 'y', 'Cp'])

def convert_cp_files_to_csv(folder_path):
    """Converts all .cp files in the folder to .csv format."""
    for filename in os.listdir(folder_path):
```

```
        if filename.endswith(".cp"):
            cp_path = os.path.join(folder_path, filename)
            df = read_cp_file(cp_path)

            # Output filename
            csv_name = filename.replace(".cp", ".csv")
            csv_path = os.path.join(folder_path, csv_name)

            df.to_csv(csv_path, index=False)
            print(f"[   ] Converted {filename}    {csv_name}")

if __name__ == "__main__":
    # Set this to the path where your .cp files are
    folder_path = "."  # Current directory; replace with full path if needed
    convert_cp_files_to_csv(folder_path)
```

B. MATLAB

The following MATLAB script [7] was used for all Data Reduction and Graph Plotting:

Listing 1 MATLAB Code for Data Analysis

```
%% A E 303 - Lab 4
% Author: Parham Khodadi
% Instructor: Xiaofeng Liu

clc; clear; close all;

%% Load Data

% NACA Data
angles = [-5, 0, 5, 10, 15, 20, 20]; % Angles of attack in degrees
naca_data = struct();

naca_files = {
    'NACA_Data/01-NACA43012A_a-05.csv'
    'NACA_Data/02-NACA43012A_a+00.csv'
    'NACA_Data/03-NACA43012A_a+05.csv'
    'NACA_Data/04-NACA43012A_a+10.csv'
    'NACA_Data/05-NACA43012A_a+15.csv'
    'NACA_Data/06-NACA43012A_a+20.csv'
    'NACA_Data/07-NACA43012A_a+20.csv'
};

for i = 1:length(angles)
    % Use curly braces to extract the file name string
    data = readmatrix(naca_files{i});

    % Store into struct
    naca_data(i).AoA = angles(i);
    naca_data(i).x = data(:,1);
    naca_data(i).y = data(:,2);
    naca_data(i).Cp = data(:,3);
end


% Experimental Data
exp_data = struct();
```

```
exp_files = {
    'Experimental_Data/q5AoA-5.csv'
    'Experimental_Data/q5AoA0.csv'
    'Experimental_Data/q5AoA5.csv'
    'Experimental_Data/q5AoA10.csv'
    'Experimental_Data/q5AoA15.csv'
    'Experimental_Data/q5AoA20-1.csv'
    'Experimental_Data/q5AoA20-2.csv'
};

for i = 1:length(exp_files)
    data = readmatrix(exp_files{i});

    % Store into struct
    exp_data(i).AoA = angles(i);
    exp_data(i).Raw = data;

    % Optional placeholders for future computed values
    exp_data(i).Cp = []; % Will compute later
    exp_data(i).x = []; % x/c values from port locations
    exp_data(i).y = []; % y/c values from port locations
end

% Normalize with q=0 AoA=0
exp_data_0 = struct();
exp_data_0.AoA = 0;
exp_data_0.Raw = readmatrix('Experimental_Data/q0AoA0.csv');
exp_data_0.Cp = []; % Will compute later
exp_data_0.x = []; % x/c values from port locations
exp_data_0.y = []; % y/c values from port locations

%% Port Locations
% From AE_303_Lab_4_Updated_Setup.pdf, page 10

% Lower surface ports 1  16
x_lower = [0.015, 0.029, 0.055, 0.080, 0.105, 0.157, 0.207, 0.257, ...
           0.306, 0.407, 0.507, 0.608, 0.708, 0.812, 0.912, 1.000];
y_lower = [-0.0089, -0.01156, -0.0160, -0.0190, -0.0216, -0.0262, -0.0299, ...
           -0.0330, -0.0353, -0.0393, -0.0402, -0.0389, -0.0353, -0.0259, ...
           -0.0132, 0.00];

% Upper surface ports 17  32
x_upper = [0.000, 0.013, 0.025, 0.048, 0.073, 0.097, 0.150, 0.200, ...
           0.250, 0.300, 0.400, 0.500, 0.600, 0.700, 0.800, 0.900];
y_upper = [0.000, 0.0394, 0.0514, 0.0678, 0.0794, 0.0868, 0.0933, ...
           0.0927, 0.0895, 0.0857, 0.0770, 0.0656, 0.0520, 0.0370, ...
           0.0244, 0.0125];

% Full airfoil port arrays (1  32 )
x_ports = [x_lower, x_upper];
y_ports = [y_lower, y_upper];

% Save for later Cp calculation
```

```
for i = 1:length(exp_data)
    exp_data(i).x = x_ports;
    exp_data(i).y = y_ports;
end

% q = 0 & AoA = 0
exp_data_0.x = x_ports;
exp_data_0.y = y_ports;

%% Conversions and Constants
inH2O_to_Psi = 0.036126; % inH2O to psi
inHg_to_Psi = 0.491154; % inHg to psi
F_to_R = 459.67;
P_amb = 30.11 * inHg_to_Psi; % psi
T_amb = 79.5 + F_to_R; % Rankine
R_air = 1716; %ft lb/(slug R)

%% Compute Experimental Cp

q_inf = 5 * inH2O_to_Psi;

cp_baseline = mean(exp_data_0.Raw(9:60, 2:4001), 2);  % From q0AoA0.csv

for i = 1:length(exp_files)
    % Read the entire CSV file into a matrix
    raw_full = readmatrix(exp_files{i});

    % Extract the block of numeric data:
    % Rows 9 to 60 (52 ports) and Columns 2 to 4001 (4000 samples)
    data = raw_full(9:60, 2:4001);

    % Compute the mean pressure for each port (average over samples)
    p_avg = mean(data, 2);
    p_corrected = p_avg - mean(cp_baseline, 2); % Subtract baseline offset

    % Use Port 33 as the freestream (static) pressure reference
    p_inf = p_avg(26);

    % Compute Cp for ports 1 to 32
    cp = (p_corrected(1:32) - p_inf) / q_inf;

    % Store the computed Cp as a column vector in exp_data
    exp_data(i).Cp = cp(:);
end

%% Compute Normal (Cn) and Axial (Ca) Force Coefficients

for i = 1:length(exp_data)
    % Extract Cp for current test (nonuniform: 1 32 ports)
    cp = exp_data(i).Cp(:);

    % Separate upper and lower surfaces
    cp_lower = cp(1:16);  % Ports 1 16
    cp_upper = cp(17:32); % Ports 17 32
```

```
    % Define uniform grid along the chord using 161 points
    x_uniform = linspace(min(x_lower), max(x_upper), 161)';
    % Interpolate the measured Cp data onto the uniform grid
    cp_lower_uniform = interp1(x_lower, cp_lower, x_uniform, 'linear', 'extrap');
    cp_upper_uniform = interp1(x_upper, cp_upper, x_uniform, 'linear', 'extrap');

    % Interpolate y-coordinates onto the uniform grid (for slope calculations)
    y_lower_uniform = interp1(x_lower, y_lower, x_uniform, 'linear', 'extrap');
    y_upper_uniform = interp1(x_upper, y_upper, x_uniform, 'linear', 'extrap');

    % Compute slopes on the uniform grid
    dy_dx_lower_uniform = gradient(y_lower_uniform, x_uniform);
    dy_dx_upper_uniform = gradient(y_upper_uniform, x_uniform);

    % ---- Cn: Normal force coefficient ----
    % (Using cp_lower - cp_upper, because the lower surface pressure is higher.)
    cn_integrand = cp_lower_uniform - cp_upper_uniform;
    cn = simpson_integration(x_uniform, cn_integrand);

    % ---- Ca: Axial force coefficient ----
    ca_integrand = cp_upper_uniform .* dy_dx_upper_uniform - cp_lower_uniform .* dy_dx_low
    ca = simpson_integration(x_uniform, ca_integrand);

    % Store results
    exp_data(i).Cn = cn;
    exp_data(i).Ca = ca;
end



%% Compute Lift and Drag Coefficients

for i = 1:length(exp_data)
    % Define Angle of Attack Alpha (in degrees)
    alpha = exp_data(i).AoA;

    % Calculate Lift Coefficient (CL) and Drag Coefficient (CD)
    CL = exp_data(i).Cn * cosd(alpha) - exp_data(i).Ca * sind(alpha);
    CD = exp_data(i).Cn * sind(alpha) + exp_data(i).Ca * cosd(alpha);

    % Store the results in the exp_data structure
    exp_data(i).CL = CL;
    exp_data(i).CD = CD;
end

% Display the computed Lift and Drag Coefficients
disp('Lift and Drag Coefficients for Experimental Data:');
for i = 1:length(exp_data)
    fprintf('AoA = %g  : CL = %.4f, CD = %.4f\n', exp_data(i).AoA, exp_data(i).CL, exp_data
end

%% Compute Pitching Moment Coefficient (Cm_LE and Cm_ac)
```

```
xac = 0.238;  % Aerodynamic center location
yac = 0.07;   % y/c location of aerodynamic center

for i = 1:length(exp_data)
    % Separate the measured Cp for upper and lower surfaces
    cp = exp_data(i).Cp(:);
    cp_lower = cp(1:16);
    cp_upper = cp(17:32);

    % Define uniform grid along the chord using 161 points
    x_uniform = linspace(min(x_lower), max(x_upper), 161)';
    % Interpolate Cp and y data onto the uniform grid
    cp_lower_uniform = interp1(x_lower, cp_lower, x_uniform, 'linear', 'extrap');
    cp_upper_uniform = interp1(x_upper, cp_upper, x_uniform, 'linear', 'extrap');
    y_lower_uniform = interp1(x_lower, y_lower, x_uniform, 'linear', 'extrap');
    y_upper_uniform = interp1(x_upper, y_upper, x_uniform, 'linear', 'extrap');

    % Compute slopes on the uniform grid (of the physical surfaces)
    dy_dx_lower_uniform = gradient(y_lower_uniform, x_uniform);
    dy_dx_upper_uniform = gradient(y_upper_uniform, x_uniform);

    % Compute moment integrals using Simpson's rule:
    % First term: (cp_upper - cp_lower) times the moment arm (x)
    I1 = simpson_integration(x_uniform, (cp_upper_uniform - cp_lower_uniform) .* x_uniform

    % Second term: Upper surface slope correction
    I2 = simpson_integration(x_uniform, cp_upper_uniform .* dy_dx_upper_uniform .* y_upper

    % Third term: Lower surface slope correction
    I3 = simpson_integration(x_uniform, cp_lower_uniform .* dy_dx_lower_uniform .* y_lower

    % Total moment about the leading edge
    Cm_LE = I1 + I2 - I3;

    % Calculate Lift (CL) and Drag (CD) using the previously computed Cn and Ca
    alpha = exp_data(i).AoA;
    CL = exp_data(i).Cn * cosd(alpha) - exp_data(i).Ca * sind(alpha);
    CD = exp_data(i).Cn * sind(alpha) + exp_data(i).Ca * cosd(alpha);

    % Shift moment from the leading edge to the aerodynamic center
    Cm_ac = Cm_LE + CL * xac * cosd(alpha) - CD * yac * cosd(alpha) ...
                + CL * yac * sind(alpha) + CD * xac * sind(alpha);

    % Store the computed moments
    exp_data(i).Cm_LE = Cm_LE;
    exp_data(i).Cm_ac = Cm_ac;
end



%% Compute Aerodynamic Coefficients from NACA Data (NACA 610 43012A)

xac = 0.238;
yac = 0.07;
```

```
for i = 1:length(naca_files)
    % Read CSV data (skip header)
    data = readmatrix(naca_files{i});
    x = data(:,1); y = data(:,2); cp = data(:,3);

    % Interpolate both surfaces onto a common x grid
    x_common = linspace(0, 1, 160)';  % Match resolution of input

    % --- Correctly split surfaces from XFOIL output ---
    % XFOIL order: LE     upper surface     TE     lower surface     LE
    n = floor(length(x)/2);  % Halfway split

    x_u = x(1:n);            y_u = y(1:n);            cp_u = cp(1:n);
    x_l = flipud(x(n+1:end)); y_l = flipud(y(n+1:end)); cp_l = flipud(cp(n+1:end));

    % Interpolate Cp and y values to common grid
    cp_u_i = interp1(x_u, cp_u, x_common, 'linear', 'extrap');
    cp_l_i = interp1(x_l, cp_l, x_common, 'linear', 'extrap');
    y_u_i  = interp1(x_u, y_u, x_common, 'linear', 'extrap');
    y_l_i  = interp1(x_l, y_l, x_common, 'linear', 'extrap');

    % Compute slopes
    dy_dx_u = gradient(y_u_i) ./ gradient(x_common);
    dy_dx_l = gradient(y_l_i) ./ gradient(x_common);

    % Force coefficients
    cn = trapz(x_common, cp_l_i - cp_u_i);
    ca = trapz(x_common, cp_u_i .* dy_dx_u - cp_l_i .* dy_dx_l);

    alpha = angles(i);
    CL = cn * cosd(alpha) - ca * sind(alpha);
    CD = cn * sind(alpha) + ca * cosd(alpha);

    % Moment integrals
    I1 = trapz(x_common, (cp_u_i - cp_l_i) .* x_common);
    I2 = trapz(x_common, cp_u_i .* dy_dx_u .* y_u_i);
    I3 = trapz(x_common, cp_l_i .* dy_dx_l .* y_l_i);
    Cm_LE = I1 + I2 - I3;

    % Cm at aerodynamic center
    Cm_ac = Cm_LE + CL * xac * cosd(alpha) - CD * yac * cosd(alpha) ...
                  + CL * yac * sind(alpha) + CD * xac * sind(alpha);

    % Store to struct if needed (optional)
    naca_data(i).CL = CL;
    naca_data(i).CD = CD;
    naca_data(i).Cm_ac = Cm_ac;

    % Print results
    fprintf('AoA = %g  : CL = %.4f, CD = %.4f, Cm_ac = %.4f\n', ...
        alpha, CL, CD, Cm_ac);
end
```

```
%% Plotting Aerodynamic Coefficients vs Angle of Attack

% Experimental Data
AoA_exp = [exp_data.AoA];
CL_exp  = [exp_data.CL];
CD_exp  = [exp_data.CD];
Cm_exp  = [exp_data.Cm_ac];

% Theoretical (NACA/XFOIL) Data
AoA_theory = [naca_data.AoA];
CL_theory  = [naca_data.CL];
CD_theory  = [naca_data.CD];
Cm_theory  = [naca_data.Cm_ac];

% --- Plot CL vs AoA ---
figure;
plot(AoA_exp, CL_exp, 'o-r', 'LineWidth', 1.8, 'MarkerSize', 6); hold on;
plot(AoA_theory, CL_theory, 's-b', 'LineWidth', 1.8, 'MarkerSize', 6);
grid on;
xlabel('Angle of Attack ( )', 'FontSize', 12);
ylabel('Lift Coefficient (C_L)', 'FontSize', 12);
title('Coefficient of Lift vs. Angle of Attack', 'FontSize', 14);
legend('Experimental Data', 'NACA/XFOIL Data', 'Location', 'northwest');
print('-depsc2', 'CL_vs_AoA.eps');

% --- Plot CD vs AoA ---
figure;
plot(AoA_exp, CD_exp, 'o-r', 'LineWidth', 1.8, 'MarkerSize', 6); hold on;
plot(AoA_theory, CD_theory, 's-b', 'LineWidth', 1.8, 'MarkerSize', 6);
grid on;
xlabel('Angle of Attack ( )', 'FontSize', 12);
ylabel('Drag Coefficient (C_D)', 'FontSize', 12);
title('Coefficient of Drag vs. Angle of Attack', 'FontSize', 14);
legend('Experimental Data', 'NACA/XFOIL Data', 'Location', 'northwest');
print('-depsc2', 'CD_vs_AoA.eps');

% --- Plot Cm_ac vs AoA ---
figure;
plot(AoA_exp, Cm_exp, 'o-r', 'LineWidth', 1.8, 'MarkerSize', 6); hold on;
plot(AoA_theory, Cm_theory, 's-b', 'LineWidth', 1.8, 'MarkerSize', 6);
grid on;
xlabel('Angle of Attack ( )', 'FontSize', 12);
ylabel('Moment Coefficient (C_{m,ac})', 'FontSize', 12);
title('Pitching Moment Coefficient vs. Angle of Attack', 'FontSize', 14);
legend('Experimental Data', 'NACA/XFOIL Data', 'Location', 'northeast');
print('-depsc2', 'Cm_ac_vs_AoA.eps');

%% Plot Cp Distribution for All Angles of Attack

% Loop through each AoA case
figure;
hold on;
colors = lines(length(exp_data)); % distinguish curves
```

```
for i = 1:length(exp_data)
    % Extract data
    x = exp_data(i).x(:);
    cp = exp_data(i).Cp(:);

    % Plot upper surface only (ports 17 32 )
    x_u = x(17:32);
    cp_u = cp(17:32);

    plot(x_u, cp_u, 'o-', 'LineWidth', 1.2, 'Color', colors(i,:), ...
        'DisplayName', sprintf('AoA = %d ', exp_data(i).AoA));
end

% Flip y-axis (by convention, lower Cp = more suction)
set(gca, 'YDir','reverse');

grid on;
xlabel('x/c', 'FontSize', 12);
ylabel('C_p (upper surface)', 'FontSize', 12);
title('Cp Distribution on Upper Surface for All AoAs', 'FontSize', 14);
legend('Location', 'best');
print('-depsc2', 'Cp_Distribution_UpperSurface.eps');


%% Compare Drag from Wake Survey vs Surface Pressure Measurement

% Constants
wake_angles = [50, 20, 15, 20, 30, 90, 90]; % degrees from AE303 Lab Setup
delta_y_prime = 0.5 * 1/12; % 0.5 in in feet

for i = 1:length(exp_data)
    raw = exp_data(i).Raw;

    % Wake rake pressures (Ports 41 60 )
    wake_block = raw(41:60, 2:end); % 20 ports x many samples
    p_rake = mean(wake_block, 2); % average pressure per port

    % Static & total pressures
    p_static = mean(raw(7, 2:end));
    p_total  = mean(raw(8, 2:end));

    % Dynamic pressure q1 = freestream
    q1 = p_total - p_static;

    % q2 = p_total_rake - p_static (assuming same static port for all)
    q2 = p_rake - p_static;

    % Ratio
    q2_q1 = q2 / q1;

    % Apply formula:
    % Cd = (2 / c) *     (sqrt(q2/q1) - q2/q1) dy
    integrand = real(sqrt(q2_q1) - q2_q1);
```

```
    % Get vertical spacing for this AoA:
    theta_rad = deg2rad(wake_angles(i));
    dy = delta_y_prime * sin(theta_rad);  % vertical distance between ports

    % Original wake grid from 20 ports:
    y_wake = (0:19)' * dy;  % This gives 20 points (19 segments), which is not allowed by

    % Create a new uniform grid with an odd number of points (e.g., 21 points)
    y_wake_new = linspace(y_wake(1), y_wake(end), 21)';
    % Interpolate the integrand onto the new grid
    integrand_new = interp1(y_wake, integrand, y_wake_new, 'linear', 'extrap');

    % Chord length in feet (assumed constant)
    c = 1.0;  % ft

    % Compute wake drag coefficient using Simpson's rule on the new grid:
    CD_wake = 2 / c * simpson_integration(y_wake_new, integrand_new);

    exp_data(i).CD_wake = CD_wake;
end



% ------------------------
% Plot comparison
% ------------------------
AoA = [exp_data.AoA];
CD_pressure = [exp_data.CD];
CD_wake = [exp_data.CD_wake];

figure;
plot(AoA, CD_pressure, 'o-r', 'LineWidth', 1.8, 'DisplayName', 'Surface Pressure');
hold on;
plot(AoA, CD_wake, 's-b', 'LineWidth', 1.8, 'DisplayName', 'Wake Survey');
grid on;
xlabel('Angle of Attack ( )', 'FontSize', 12);
ylabel('Drag Coefficient (C_D)', 'FontSize', 12);
title('Comparison of Drag Coefficients: Surface Pressure vs. Wake Survey', 'FontSize', 14)
legend('Location', 'northwest');
print('-depsc2', 'CD_Comparison_Wake_vs_Surface.eps');

%% Plot Wake Velocity Profiles for All Angles of Attack

% Wake rake info from AE_303_Lab_4_Updated_Setup.pdf
delta_y_prime = 0.5 * 1/12;  % Hypotenuse spacing between wake ports (feet)

% Angles from the table (degrees)
rake_angles = [50, 20, 15, 20, 30, 90, 90];  % One per AoA test
colors = lines(length(exp_data));

figure;
hold on;
```

```
for i = 1:length(exp_data)
    raw = exp_data(i).Raw;

    % Wake rake pressures: rows 41  60   (ports 33  52  )
    p_rake = mean(raw(41:60, 2:4001), 2);  % Average over samples

    % Total pressure for test (row 8), static pressure (row 7)
    p_total = mean(raw(8, 2:4001));
    p_static = mean(raw(7, 2:4001));

    % Internal temperature (row 5 or 6 depending on your spreadsheet layout)
    T_internal = raw(5,5) + F_to_R;  % F      R

    % Density
    rho = P_amb / (R_air * T_internal);

    % Compute wake velocity (Bernoulli)
    u_wake = sqrt(2 * (p_total - p_rake) / rho);  % Vector of 20 values

    % Each port is offset by delta_y_prime along rake      vertical spacing:
    y_wake = (0:19)' * delta_y_prime * sind(rake_angles(i));  % 20 x 1 vector

    % Plot
    plot(u_wake, y_wake, 'LineWidth', 1.6, ...
        'Color', colors(i,:), ...
        'DisplayName', sprintf('AoA = %d  ', exp_data(i).AoA));
end

grid on;
xlabel('Wake Velocity u (ft/s)', 'FontSize', 12);
ylabel('Vertical Position y (in)', 'FontSize', 12);
title('Wake Velocity Profiles for All AoAs', 'FontSize', 14);
legend('Location', 'best');
print('-depsc2', 'Wake_Velocity_Profiles.eps');

%% Functions
function I = simpson_integration(x, y)
    % Simpson's composite integration assumes an even number of segments.
    n = length(x);
    if mod(n-1,2) ~= 0
        error('Simpson''s rule requires an even number of segments (odd number of points).
    end
    h = (x(end)-x(1))/(n-1);
    I = y(1) + y(end) + 4*sum(y(2:2:end-1)) + 2*sum(y(3:2:end-2));
    I = I * h / 3;
end
```