Zewail City of Science, Technology and Innovation

Flow Simulation and Analysis over a Joukowski Airfoil

Assigned Parameters	
Maximum Thickness (t/c)**:**	11%
Maximum Camber (h/c)**:**	4.25%

Author:
Walid Sherif
202200702

Supervisors:
Dr. Mohamed Madbouly
Eng. Asmaa AlaaElDeen

January 22, 2026

Contents

1 Introduction

The study of aerodynamic flow over airfoils is a cornerstone of aerospace engineering. The **Joukowski airfoil** provides an analytical foundation for understanding lift generation and pressure distribution through the mathematical elegance of *conformal mapping*. By transforming a circle in the complex plane into a streamlined aerodynamic shape, potential flow theory can be used to predict surface pressures and aerodynamic coefficients.

In this project, a computational flow simulation is conducted based on a specific Joukowski geometry. The primary goal is to analyze the flow field under high-speed subsonic conditions and evaluate the performance of the airfoil regarding its lift and moment characteristics.

2 Technical Specifications and Givens

The simulation is governed by a specific set of input parameters and environmental conditions assigned to ensure high-speed aerodynamic accuracy.

Table 1: Summary of Input Parameters and Flow Conditions

Category	**Parameter**	**Value**
Flow Conditions	Free-stream Velocity (V_∞)	$125\,\mathrm{m\,s^{-1}}$
Airfoil Geometry	Chord Length (c)	$1.25\,\mathrm{m}$
	Max. Thickness Ratio (t/c)	11%
	Max. Camber Ratio (h/c)	4.25%

3 Simulation Results and Analysis

The following sections detail the visual and numerical outcomes of the MATLAB-based potential flow simulation.

3.1 Geometry Generation

Using the Joukowski transformation, the circle was mapped into a profile with 11 % thickness and 4.25 % camber. Figure 1 shows the smooth curvature of the resulting geometry.

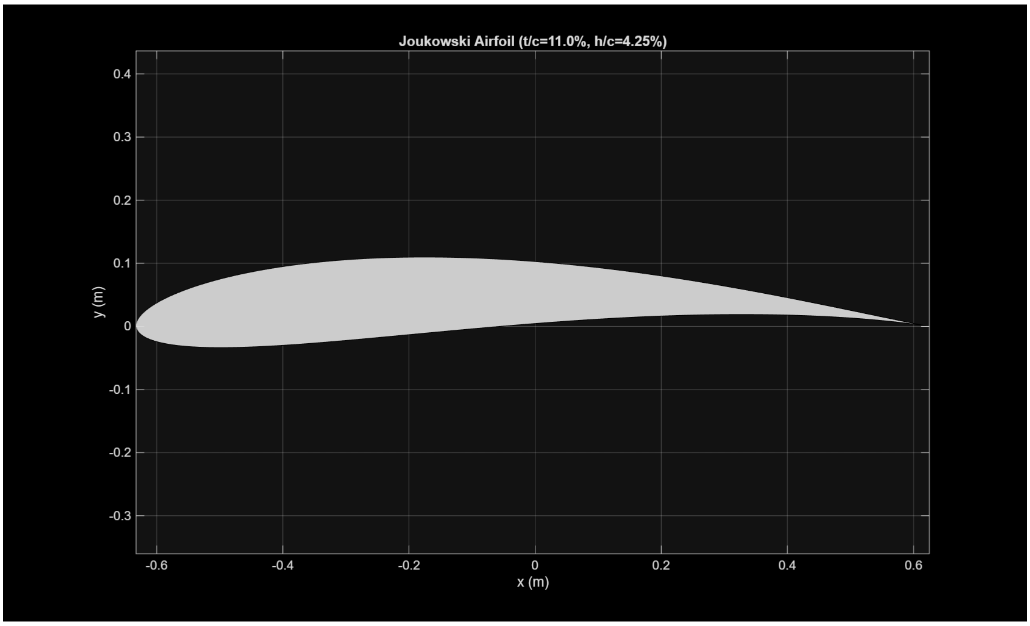

Figure 1: Generated Joukowski Airfoil Geometry profile.

3.2 Flow Field and Velocity Analysis

The streamline distribution (Figure 2) shows how the fluid follows the camber of the airfoil. The velocity contour in Figure 3 illustrates the acceleration on the upper surface, which is responsible for the pressure drop and subsequent lift.

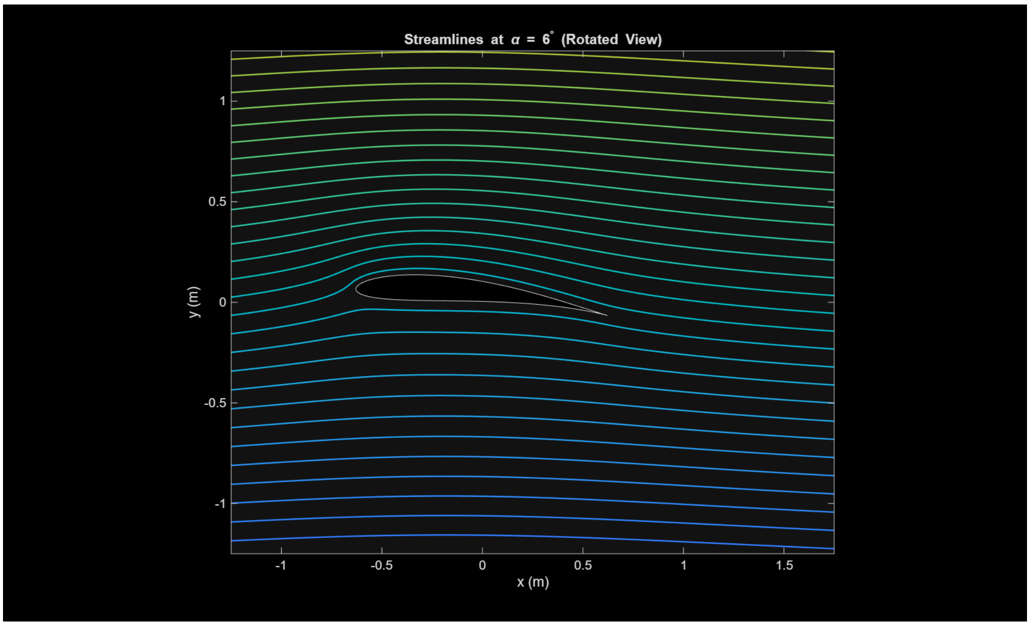

Figure 2: Streamlines at $\alpha = 6°$ showing stagnation points and curvature.

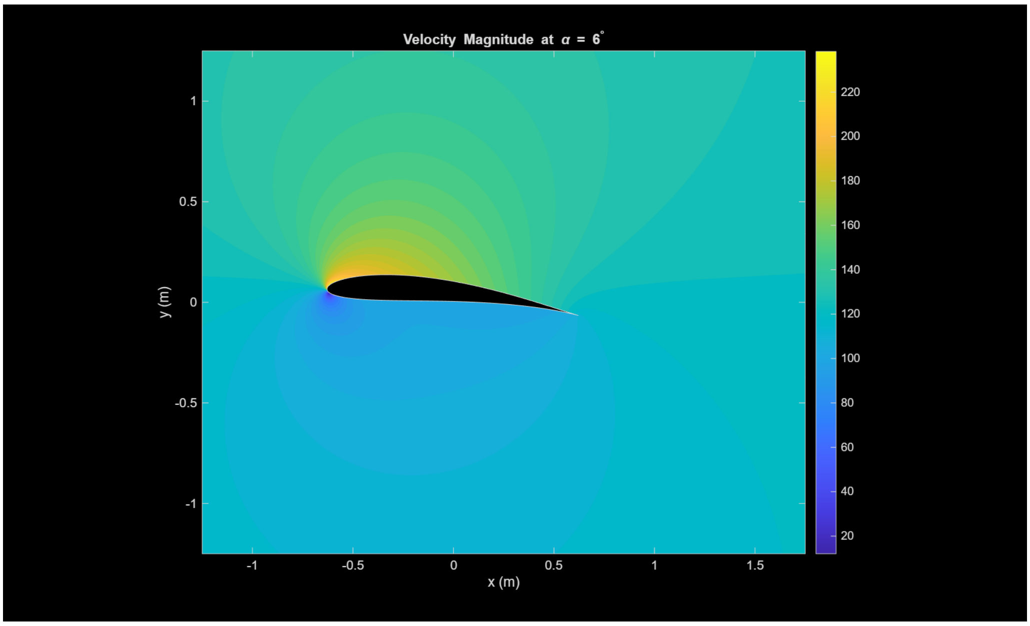

Figure 3: Velocity magnitude distribution around the airfoil.

3.3 Pressure Distribution and Performance

The pressure coefficient (C_p) plot (Figure 4) highlights the suction peak on the upper surface. The resulting Lift and Moment coefficients are plotted against the Angle of Attack (α) in Figures 5 and 6.

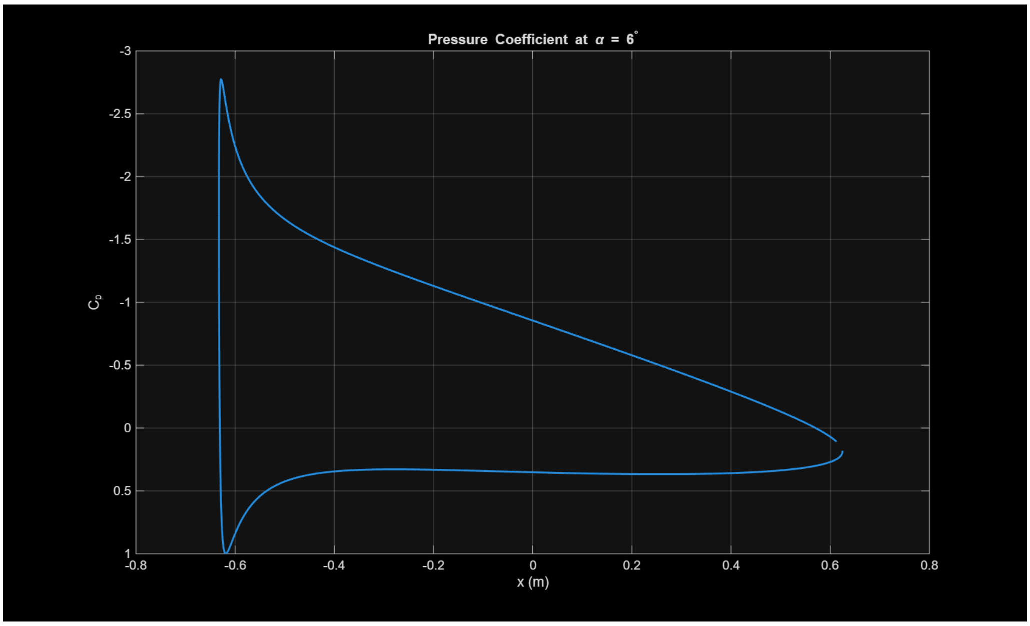

Figure 4: Pressure coefficient (C_p) distribution along the chord.

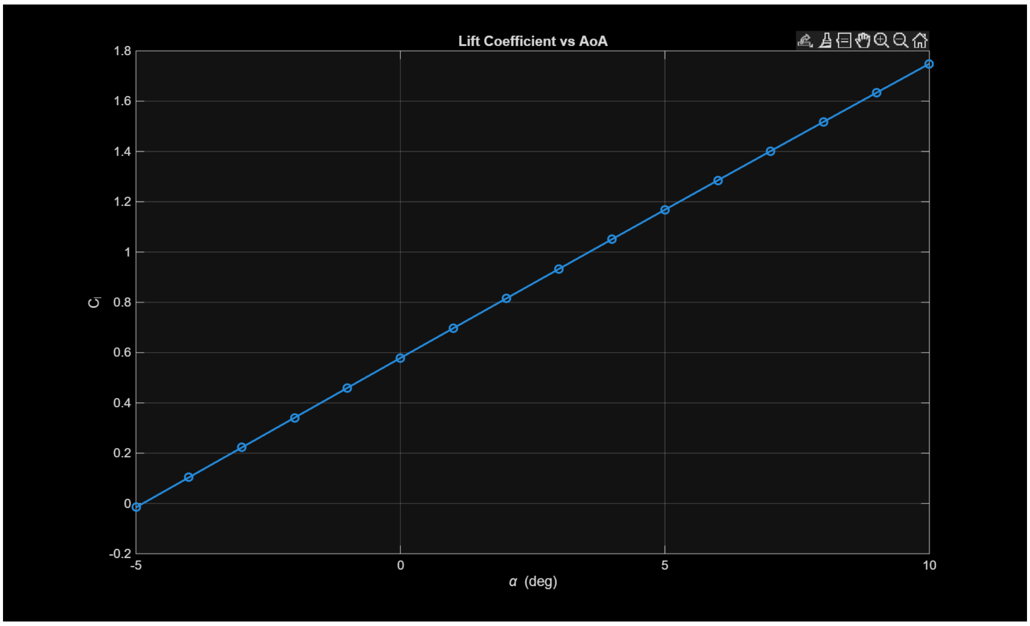

Figure 5: Lift Coefficient (C_l) vs. Angle of Attack (α).

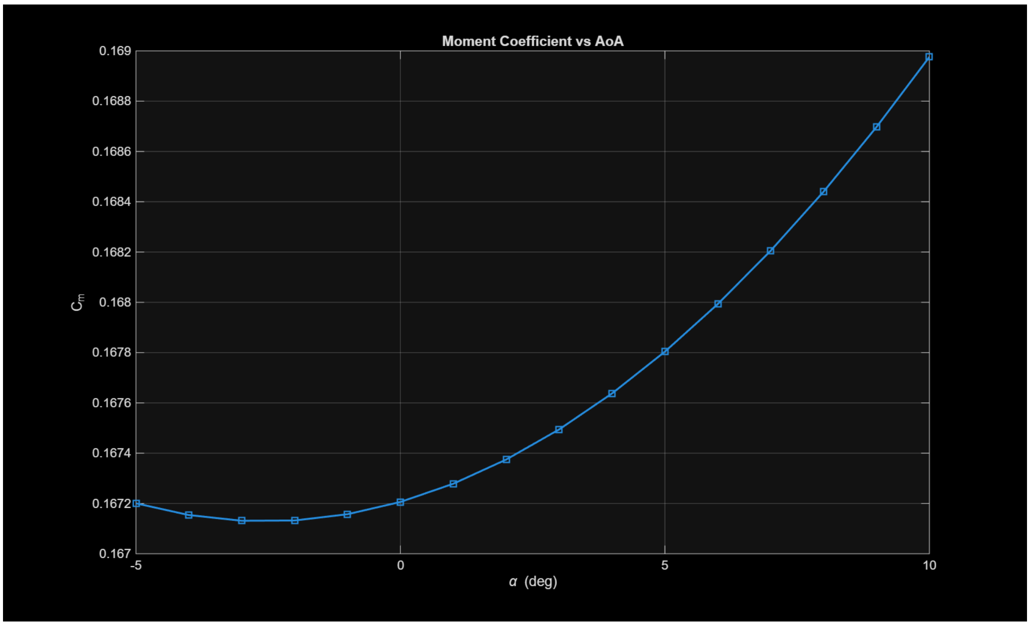

Figure 6: Moment Coefficient (C_m) vs. Angle of Attack (α).

4 Appendix: MATLAB Source Code

The simulation was implemented using the following script to calculate the Joukowski transformation and aerodynamic loads.

```
clc; clear; close all;

%% 1. Input Parameters
V_inf = 125;          % Free stream velocity (m/s)
c = 1.25;             % Chord length (m)
tc_ratio = 0.11;    % Max Thickness t/c
hc_ratio = 0.0425;  % Max Camber h/c

% Angles
alpha_vis = 6;                % AoA for flow visualization (degrees)
alpha_range = -5:1:10;        % AoA range for Cl and Cm plots

%% 2. Joukowski Transformation Setup (New Method)
% Equations from "Flow Past Joukowski Airfoil.pdf"
b = c / 4;
e = (tc_ratio) / 1.3;          % Thickness parameter [cite: 5]
beta = 2 * (hc_ratio);         % Camber parameter (radians) [cite: 5]
a = b * (1 + e) / cos(beta); % Cylinder radius [cite: 6]

% Center of Cylinder (z0)
x0 = -b * e;                   % [cite: 7]
y0 = a * beta;                 % [cite: 7]
z0 = x0 + 1i * y0;

%% 3. Generate Airfoil Geometry
theta = linspace(0, 2*pi, 400);
Z_prime_circle = a * exp(1i * theta); % Circle centered at origin (Z' plane)
Z_circle = Z_prime_circle + z0;        % Shifted circle (Z plane)
z_airfoil = Z_circle + b^2 ./ Z_circle; % Joukowski Transform (Z1 plane)

% Plot 1: Geometry
figure(1);
fill(real(z_airfoil), imag(z_airfoil), [0.8 0.8 0.8], 'EdgeColor', 'k');
axis equal; grid on;
xlabel('x (m)'); ylabel('y (m)');
title(sprintf('Joukowski Airfoil (t/c=%.1f%%, h/c=%.2f%%)', tc_ratio*100, hc_ratio*100));
saveas(gcf, 'Fig1_Geometry.png');

%% 4. Streamlines & Velocity (Rotated View)
% Grid Generation in Z' plane (r, theta)
```

```
r_vals = linspace(a, 8*a, 100);
theta_vals = linspace(0, 2*pi, 150);
[R_grid, Theta_grid] = meshgrid(r_vals, theta_vals);
Z_prime_grid = R_grid .* exp(1i * Theta_grid);

% Transform Grid to Physical Plane (Z1)
Z_grid = Z_prime_grid + z0;
Z1_grid = Z_grid + b^2 ./ Z_grid;

% Velocity Calculation in Z' Plane
alpha_rad = deg2rad(alpha_vis);
Gamma = 4 * pi * V_inf * a * sin(alpha_rad + beta); % Circulation

% Velocity components in Z' plane
% v_r' = V * cos(theta' - alpha) * (1 - a^2/r'^2)
vr_prime = V_inf .* cos(Theta_grid - alpha_rad) .* (1 - a^2 ./ R_grid.^2);
% v_theta' = -V * [sin(theta' - alpha)(1 + a^2/r'^2) + 2(a/r')sin(alpha+beta)]
vt_prime = -V_inf .* (sin(Theta_grid - alpha_rad) .* (1 + a^2 ./ R_grid.^2) + ...
                        2 * (a ./ R_grid) .* sin(alpha_rad + beta));

% Complex Velocity dW/dZ' = (vr - i*vt) * e^(-i*theta')
dW_dZ_prime = (vr_prime - 1i * vt_prime) .* exp(-1i * Theta_grid);

% Derivative of Transformation dZ1/dZ'
% Z1 = Z + b^2/Z   -> dZ1/dZ = 1 - b^2/Z^2
% Z = Z' + z0      -> dZ/dZ' = 1
dZ1_dZ_prime = 1 - b^2 ./ (Z_prime_grid + z0).^2;

% Velocity in Physical Plane: V = (dW/dZ') / (dZ1/dZ')
V_complex = dW_dZ_prime ./ dZ1_dZ_prime;
V_mag = abs(V_complex);

% Stream Function (Psi) Calculation
W = V_inf .* (Z_prime_grid .* exp(-1i*alpha_rad) + (a^2 ./ Z_prime_grid) .* exp(1i*alpha_rad)) + ...
      1i * Gamma / (2*pi) * log(Z_prime_grid ./ a);
Psi = imag(W);

% ROTATION FOR PLOTTING (Airfoil at 6 deg, Flow Horizontal)
% We rotate the coordinate system by -alpha.
rot_angle = -alpha_rad;
Z1_grid_rot = Z1_grid * exp(1i * rot_angle);
z_airfoil_rot = z_airfoil * exp(1i * rot_angle);

% Plot 2: Streamlines
figure(2);
```

```
contour(real(Z1_grid_rot), imag(Z1_grid_rot), Psi, 60, 'LineWidth', 1.2); hold on;
fill(real(z_airfoil_rot), imag(z_airfoil_rot), 'k');
axis equal; axis([-c c+0.5 -c c]);
title(['Streamlines at \alpha = ' num2str(alpha_vis) '^\circ (Rotated View)']);
xlabel('x (m)'); ylabel('y (m)');
saveas(gcf, 'Fig2_Streamlines.png');

% Plot 3: Velocity Distribution
figure(3);
contourf(real(Z1_grid_rot), imag(Z1_grid_rot), V_mag, 50, 'LineColor', 'none');
colorbar; hold on;
fill(real(z_airfoil_rot), imag(z_airfoil_rot), 'k');
title(['Velocity Magnitude at \alpha = ' num2str(alpha_vis) '^\circ']);
axis equal; axis([-c c+0.5 -c c]);
xlabel('x (m)'); ylabel('y (m)');
saveas(gcf, 'Fig3_Velocity.png');

%% 5. Pressure Coefficient Cp
% Calculate on airfoil surface (r' = a)
theta_s = linspace(0.1, 2*pi-0.1, 300); % Avoid stagnation points for stability
Z_p_s = a * exp(1i * theta_s);
Z_s = Z_p_s + z0;
Z1_s = Z_s + b^2 ./ Z_s;

% Surface Velocity
vt_s = -V_inf .* (sin(theta_s - alpha_rad) * 2 + 2 * sin(alpha_rad + beta));
dZ1_dZ_p_s = 1 - b^2 ./ Z_s.^2;
V_surf = abs(vt_s ./ abs(dZ1_dZ_p_s));

Cp = 1 - (V_surf / V_inf).^2;

figure(4);
plot(real(Z1_s), Cp, 'LineWidth', 1.5);
set(gca, 'YDir', 'reverse'); grid on;
xlabel('x (m)'); ylabel('C_p');
title(['Pressure Coefficient at \alpha = ' num2str(alpha_vis) '^\circ']);
saveas(gcf, 'Fig4_Cp.png');

%% 6. Lift and Moment Coefficients
Cl_vec = []; Cm_vec = [];

for ang = alpha_range
    a_r = deg2rad(ang);

```

```
    % Lift Coefficient [cite: 44]
    % Cl = 2*pi*(1+e)*sin(alpha + beta)
    Cl = 2 * pi * (1 + e) * sin(a_r + beta);
    Cl_vec = [Cl_vec, Cl];

    % Moment Coefficient (Numerical Integration of Cp)
    Gam_i = 4 * pi * V_inf * a * sin(a_r + beta);
    vt_i = -V_inf .* (sin(theta_s - a_r) * 2 + 2 * sin(a_r + beta
   ));
    V_s_i = abs(vt_i ./ abs(dZ1_dZ_p_s));
    Cp_i = 1 - (V_s_i / V_inf).^2;

    % Integrate Moment about Quarter Chord (x = -b)
    x_ac = -b;
    M = 0;
    x_loc = real(Z1_s); y_loc = imag(Z1_s);
    for k = 1:length(x_loc)-1
        dx = x_loc(k+1) - x_loc(k);
        dy = y_loc(k+1) - y_loc(k);
        x_m = (x_loc(k+1) + x_loc(k))/2;
        y_m = (y_loc(k+1) + y_loc(k))/2;
        cp_m = (Cp_i(k+1) + Cp_i(k))/2;

        dFx = -cp_m * dy; % Normal force components (Cp acts
   normal)
        dFy = cp_m * dx;

        M = M + (x_m - x_ac)*dFy - (y_m)*dFx;
    end
    Cm = M / c;
    Cm_vec = [Cm_vec, Cm];
end

figure(5);
plot(alpha_range, Cl_vec, '-o', 'LineWidth', 1.5);
grid on; xlabel('\alpha (deg)'); ylabel('C_l');
title('Lift Coefficient vs AoA');
saveas(gcf, 'Fig5_Cl_alpha.png');

figure(6);
plot(alpha_range, Cm_vec, '-s', 'LineWidth', 1.5);
grid on; xlabel('\alpha (deg)'); ylabel('C_m');
title('Moment Coefficient vs AoA');
saveas(gcf, 'Fig6_Cm_alpha.png');

disp('Simulation Complete.');
```

Listing 1: Joukowski Airfoil Simulation Script

5 Ansys Bouns

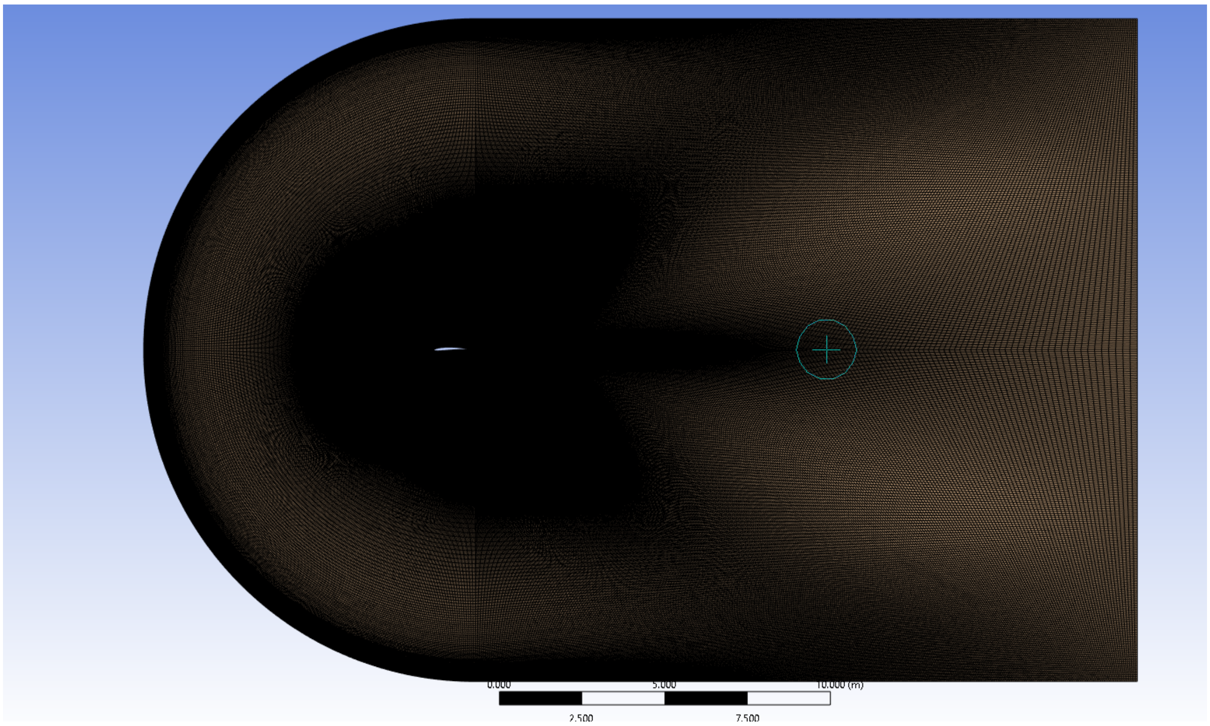

Figure 7: mesh

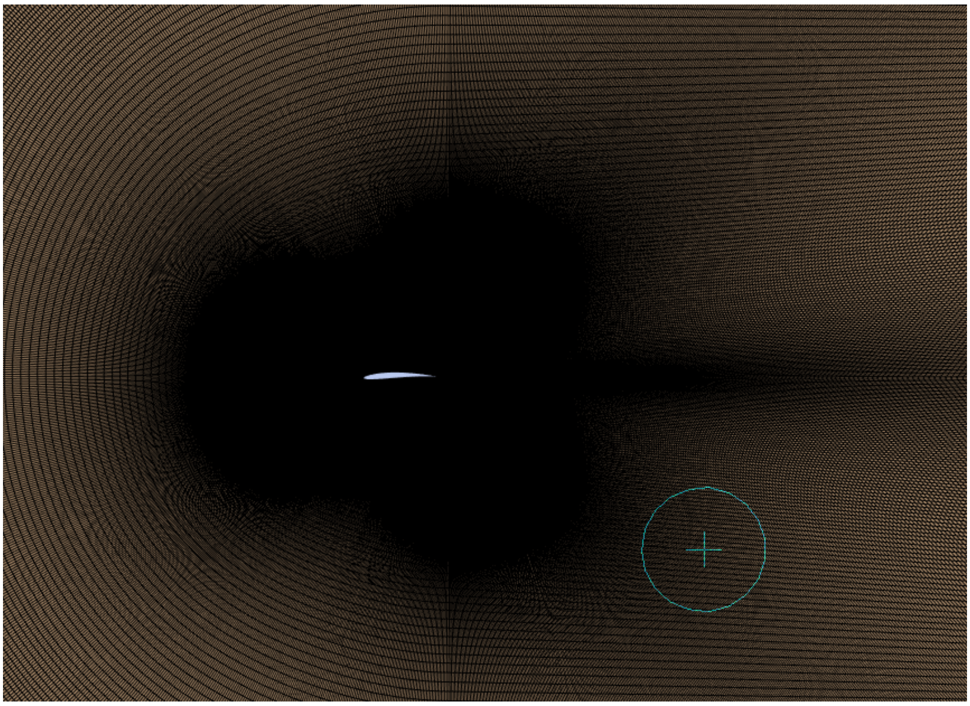

Figure 8: airfoil mesh

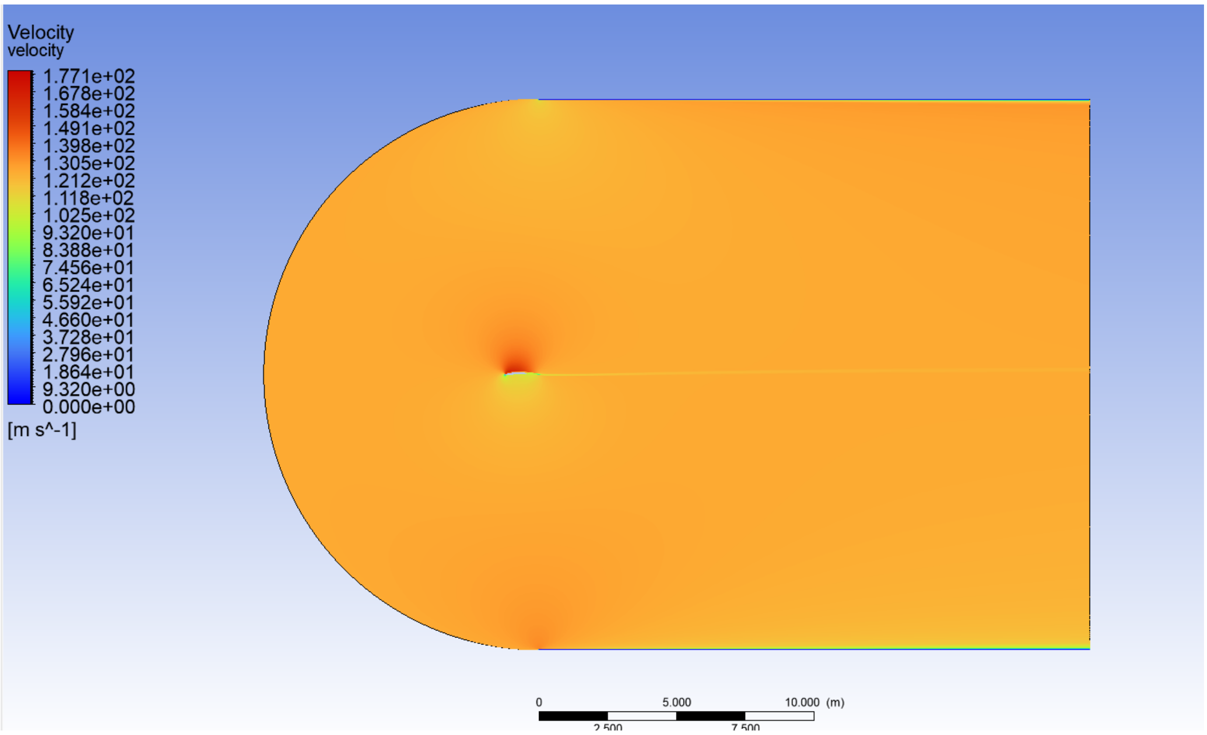

Figure 9: velocity contour

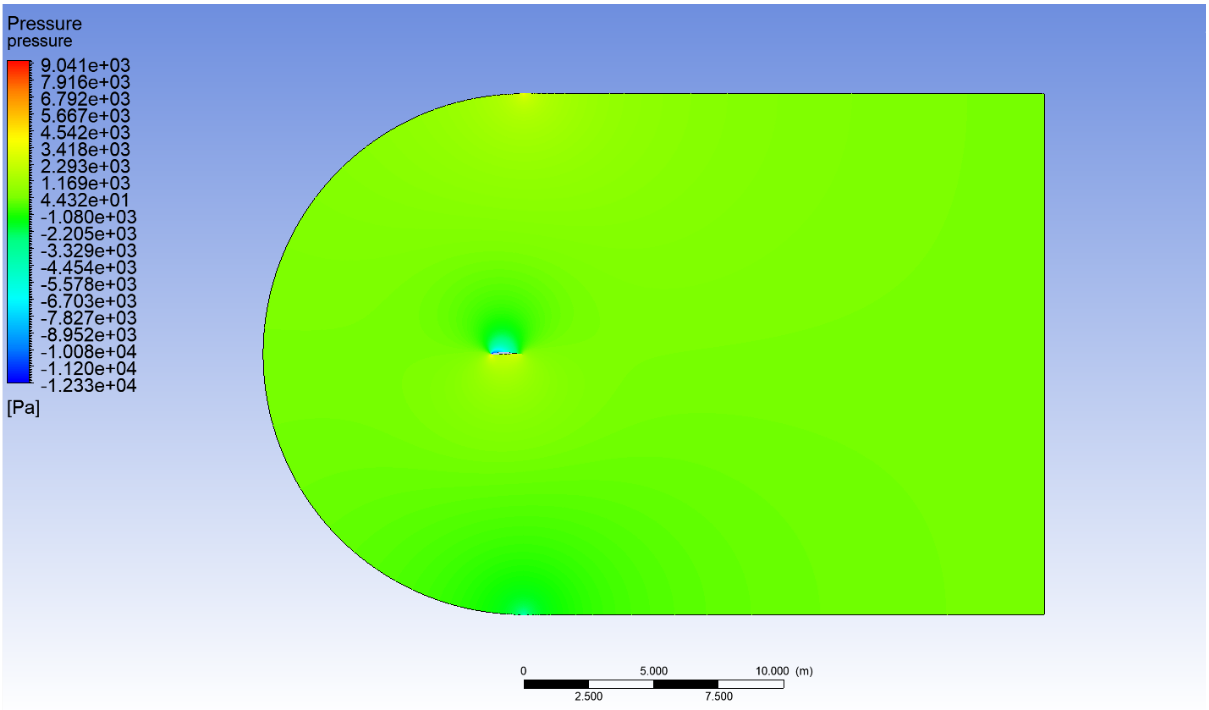

Figure 10: pressure contour

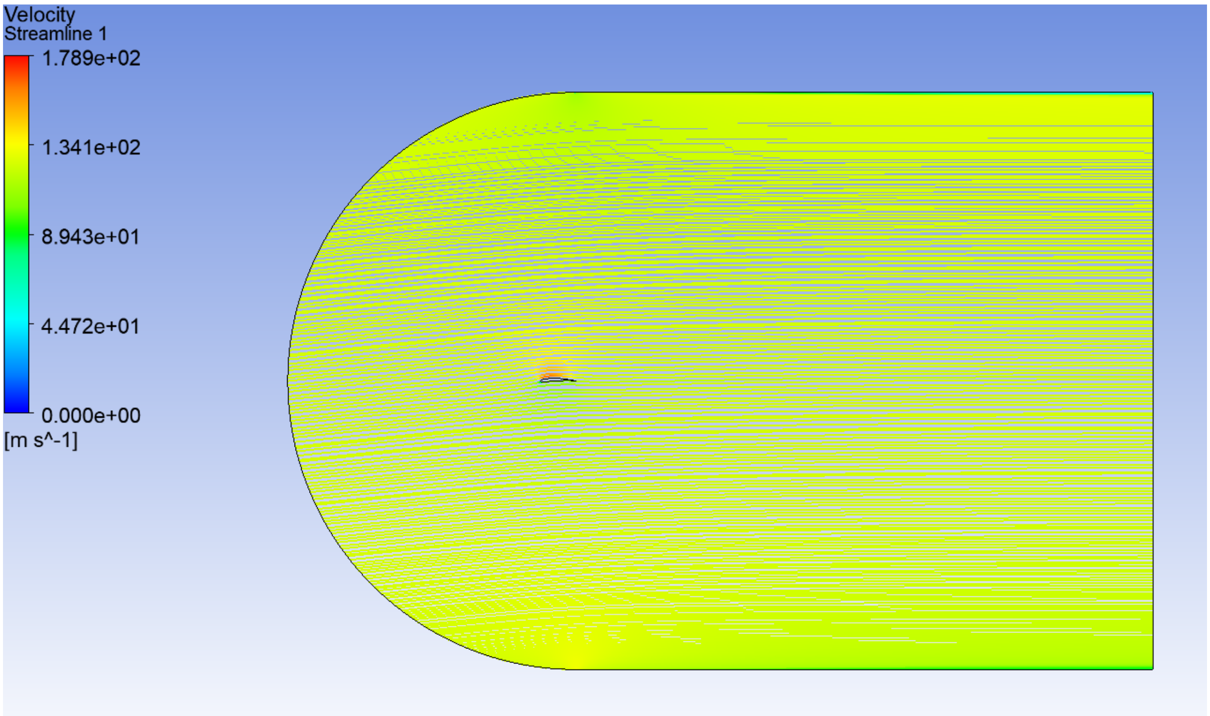

Figure 11: streamlines

	Cl
wall-surface_body	1.02555
	Cd
wall-surface_body	0.052561256
	Cm
wall-surface_body	-0.16032557

Figure 12: coefficients