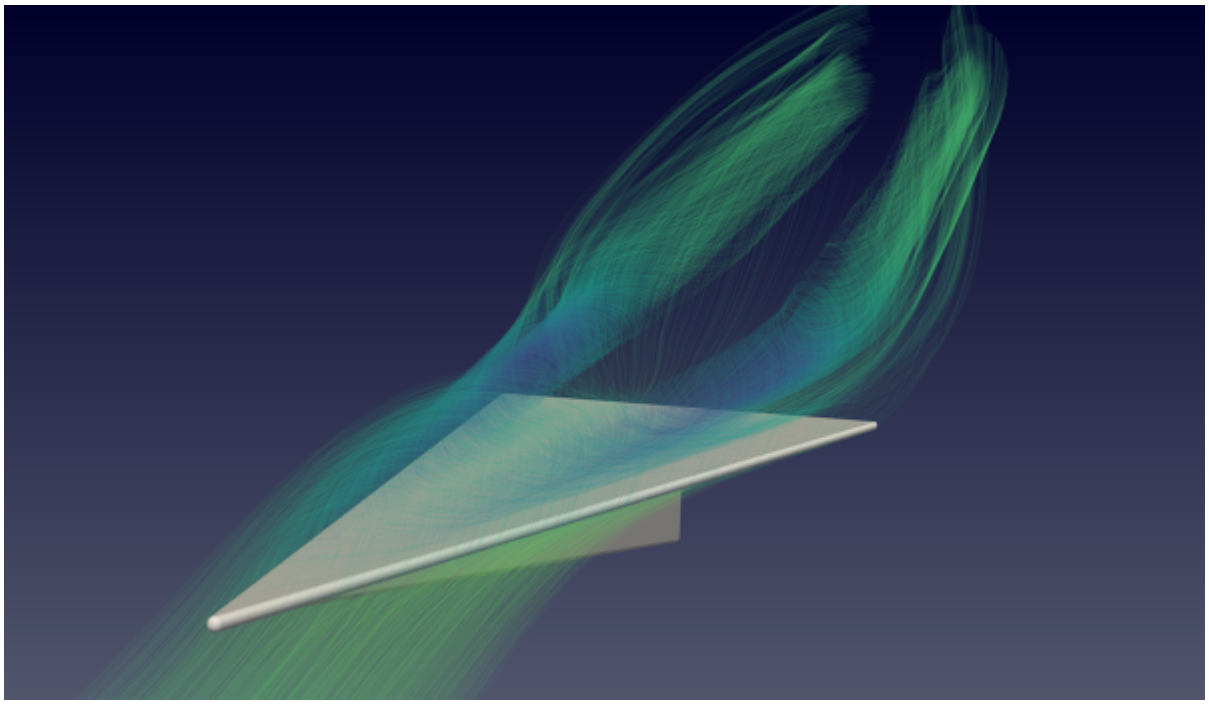
AE244

LOW SPEED AERODYNAMICS

(Prof. Dhwanil Shukla)



(Figure: Delta wing Aerodynamics)

ASSIGNMENT-2

ARPIT JAIN

22B0078

1. Team introduction and work share

Name	*Roll number*	*Contribution level*	*Specifics of Contribution*
Muriki Sree Baruni	*22B0059*	*5*	*Algorithms, Functions for Slope A_n, Cl and Gamma integrations*
Arpit Jain	*22B0078*	*5*	*Graphs, User defined Airfoils, Functions for vector field, gamma and circulation*
Milan Kumari	*22B0006*	*5*	*Functions to plot NACA, Cm and slope at general points*

2. Algorithms

Input cell takes various inputs from the user. They are as follows:

1. Choice of airfoil ‘y’ for NACA, ‘n’ for User defined polynomial function in ‘x’
2. Free stream velocity (U_infi)
3. Angle of attack in degrees (alpha_deg)

2.1- Algorithm for plotting camber line

The algorithm for plotting camber line for user defined airfoil (be it custom or NACA) is as follows:

If choice==’n’:

Step 1: Ask the user to input a valid python expression of a polynomial function in ‘x’

Step 2: Pass an array containing the x-coordinates of the chord line and the custom function as arguments to the function **generate_camber_points_userdef**.

Step 3: The function returns the y coordinates values of the camber line.

Step 4: Plot x_val vs y_val to obtain camber line plot. Set title, labels.

If choice==’y’:

Step 1: Ask the user to input a 4-digit number (as per NACA naming convention).

Step 2: Extract the required NACA airfoil parameters m and p from the input number by passing it to the function **separate_digits**.

Step 3: Generate x_values i.e., x coordinates for points on camber line.

Step 4: Pass the obtained m, p along with the x_values to obtain camber value y_values iteratively into the function **camber_line_function**. The function contains if - else if conditional statements.

If 0 < x_values[i] < p:

Y_values[i] = (m / (p**2)) * (2 * p * x_values[i] – x_values[i]**2).

Else if p <= x <= 1:

y_values [i] = (m / (1 - p) ** 2) * (1 - (2 * p) + 2 * p * x_values [i] – x_values [i]**2).

Step 5: Plot x_values vs y_values to obtain camber line plot. Set title, labels.

The equations used in computing camber values are obtained from NACA website.
http://airfoiltools.com/airfoil/naca4digit

	Front ($0 \le x < p$)	Back ($p \le x \le 1$)
Camber	$y_c = \frac{M}{P^2}(2Px - x^2)$	$y_c = \frac{M}{(1-P)^2}(1 - 2P + 2Px - x^2)$
Gradient	$\frac{dy_c}{dx} = \frac{2M}{P^2}(P - x)$	$\frac{dy_c}{dx} = \frac{2M}{(1-P)^2}(P - x)$

2.2 Algorithm for plotting camber line slope

	Front ($0 \le x < p$)	Back ($p \le x \le 1$)
Camber	$y_c = \frac{M}{P^2}(2Px - x^2)$	$y_c = \frac{M}{(1-P)^2}(1 - 2P + 2Px - x^2)$
Gradient	$\frac{dy_c}{dx} = \frac{2M}{P^2}(P - x)$	$\frac{dy_c}{dx} = \frac{2M}{(1-P)^2}(P - x)$

The algorithm for plotting camber line slope for user defined airfoil is as follows:

If choice==’n’:

Step 1: Pass an array containing the x-coordinates of the chord line and the custom function as arguments to the function **generate_camber_points_userdef** to obtain camber values.

Step 2: Utilise the obtained camber values for numerical differentiation (Central difference) of the custom polynomial using the function **camber_slope_user**.

Step 3: Iterate the function over all the x-coordinates to obtain camber slope at 1000 points.

Step 4: Plot x-coordinates vs camber slope values to obtain camber slope plot.

If choice==’y’

Step 1: Pass an array x containing the x-coordinates of the chord line into the function **camber_line_slope.**

Step 2: The function iterates over each point in the chord line using a for loop containing if-else if conditional statements to compute camber line slope and return an array of slopes at several points.

If 0 < x[i] < p:

y[i] = (m / (p**2)) * (2 * p – 2 * x[i]).

Else if p <= x[i] <= 1:

y[i] = (m / (1 - p) ** 2) * (2 * p – 2 * x[i]).

Step 3: Plot x vs slopes obtained from the camber_line_slope(x) to obtain camber line slope plot.

If the slope of the camber line is to be computed at any point on the camber line, then the function **camb_line_slope** that takes in the x coordinate of the point as an argument to return the slope at that point. This function also uses if-else if conditional statements as mentioned above.

If 0 < x < p:

y = (m / (p**2)) * (2 * p – 2 * x).

Else if p <= x <= 1:

y = (m / (1 - p) ** 2) * (2 * p – 2 * x).

2.3. Algorithm for computing Cl

The computation of Cl requires the values of A0 and A1. This is sought from the ‘Thin airfoil theory’.

Coefficient of Lift: $C_l = \pi(2A_0 + A_1)$

The computation of A0 and A1 boil down to evaluating the integrals shown below.

$$A_0 = \alpha - \frac{1}{\pi}\int_0^{\pi} \frac{dz}{dx}\, d\theta$$

$$A_n = \frac{2}{\pi}\int_0^{\pi} \frac{dz}{dx}\cos n\theta\, d\theta$$

Here's an algorithm for computing Cl:

A function **compute_A_n** is defined to compute An at different values of n. This function:

§ Initialises an array cos_array of size 1000 with zeros.

§ Calculates the cosine values for each theta value at index i and store them in cos_array using a for loop.

§ Computes y (the integrand) by multiplying camber slope with each element in cos_array.

§ Calculate the trapezoidal area under the curve formed by y values and theta values, and add it to ans.

§ If n is 0: A_n = (alpha_deg)*(math.pi/180) - (1/math.pi)* ans #from definition

§ Else: A_n = (2/math.pi)*ans #for all integral n > 0

§ Returns A_n

To compute Cl:

A function **compute_cl** is called by passing the argument alpha_deg.

This function computes Cl using the above mentioned equation Cl = 3.14*(2*A[0] + A[1]).

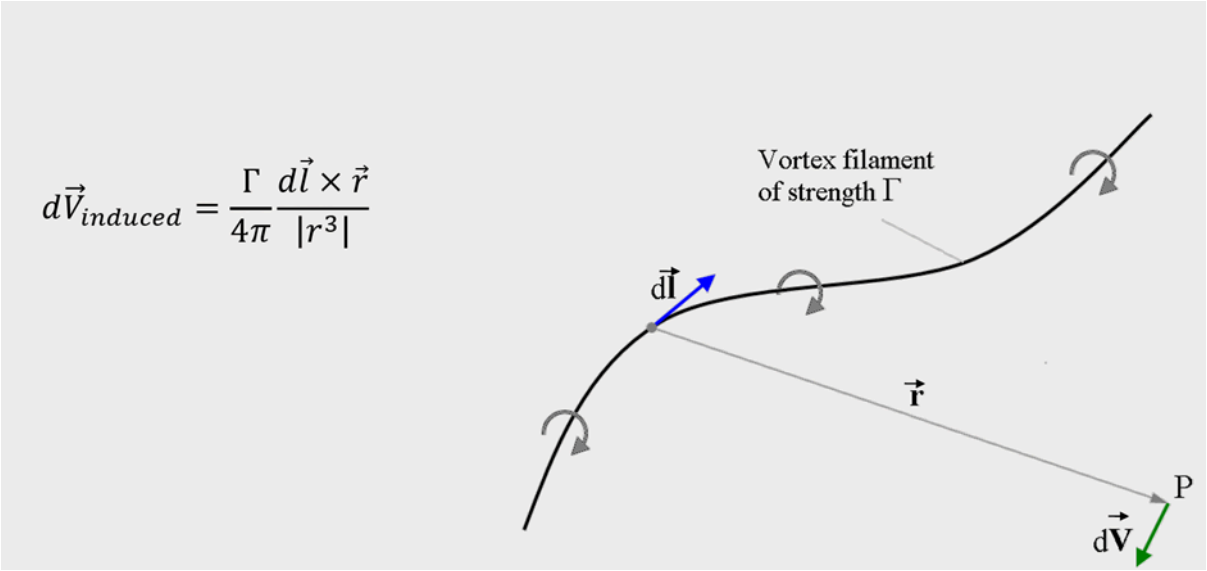
2.4. Algorithm for plotting vector field

At any point around airfoil,

Velocity = Free stream velocity + Velocity induced at that point by vortex filaments along camber line

Free stream velocity is determined by the user inputs (magnitude of free stream velocity, angle of attack).

Velocity induced at that point by vortex filaments along the camber line is computed using Biot – Savart's law.



Here's an algorithm to obtain a vector field plot:

Step 1: A function named **create_A_n_sin** is defined to create which

- Initialises an array subarray of size 1001 with zeros to store the summation results.
- Iterates over the range from 0 to 1000: to add the product of A_n to sin(n*theta) (for all n>0) to return sumarray

Step 2: A function **gamma_sin** is defined to evaluate circulation distribution along the camber line. The function computes cos_array and sin_array and iterates over 1 to n to return

$$\gamma(\xi) = (1/2) * \gamma(\theta) * sin\theta = 0.5 * 2U_\infty\{ A_0(1 + \cos\theta) + sin\theta\left(\sum_{n=1}^{\infty}(A_n \sin n\theta)\right) \}$$

Step 3: A function **velocity** (x_values,y_values) is defined to obtain velocity field plot in the meshgrid (4c*3c) where: x_values – array containing n = 1000 discrete points on chord line [x-coordinates]

y_values – array containing n = 1000 discrete points along camber line [y-coordinates]

This function:

- Initialises x_box , y_box arrays to define range of x and y values for meshgrid
- Creates a meshgrid using np.meshgrid() function.
- Initialises v_x, v_y arrays to store x and y components of velocity.
- Initialises r_x, r_y , r and ds arrays for distances and lengths of discrete (differential) elements.
- Computes velocity components:
  - For each point in the meshgrid, compute the distances r_x, r_y from the point to each and every point along camber line
  - Calculate overall distance, r and length of the segment, ds.
  - Compute v_x and v_y using Biot – Savart's law
  - Add the free-stream velocity components to v_x and v_y
  - Loop the above steps through each point in the meshgrid.

Plot the vector field using plt.quiver() with x, y, v_x, v_y and camber line for reference. Set labels, title.

Returns the v_x, v_y.

2.5 Algorithm for calculating circulation through line integral

$$-\Gamma = \oint_C \vec{V} \cdot \vec{ds} = \iint_A (\nabla \times \vec{V}) \cdot \hat{n}\, dA$$

The above definition of circulation helps us calculate circulation using velocity line integral.

Here's an algorithm to be implemented to compute line integral of velocity along a curve:

We chose to consider a rectangle. This rectangle is the same as the boundary of the meshgrid used in the velocity vector field plot.

ü A function named **circulation_line_integral** (g) is defined that accepts a parameter (argument), number of grid points along the square grid.

This function:

§ computes the grid spacing along x and y directions based on the number of grid points g.

§ initialises the line line integral vds to zero.

§ computes the line integral by summing the product of velocity component and grid spacing along each component

§ considers the contributions of all 4 sides of a rectangle, with appropriate signs.

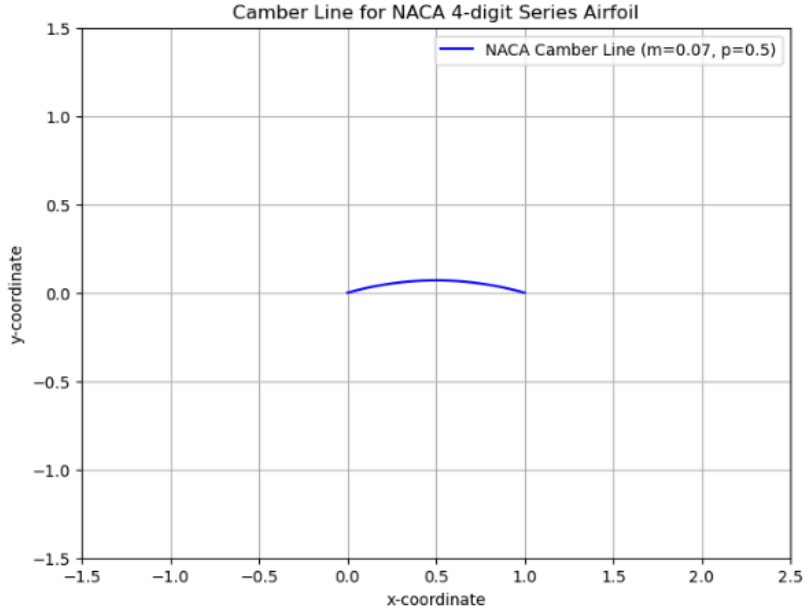
§ iterates the above 2 steps over each grid point along a rectangle using for loops.

§ And finally returns vds, the computed value of circulation using line integral.

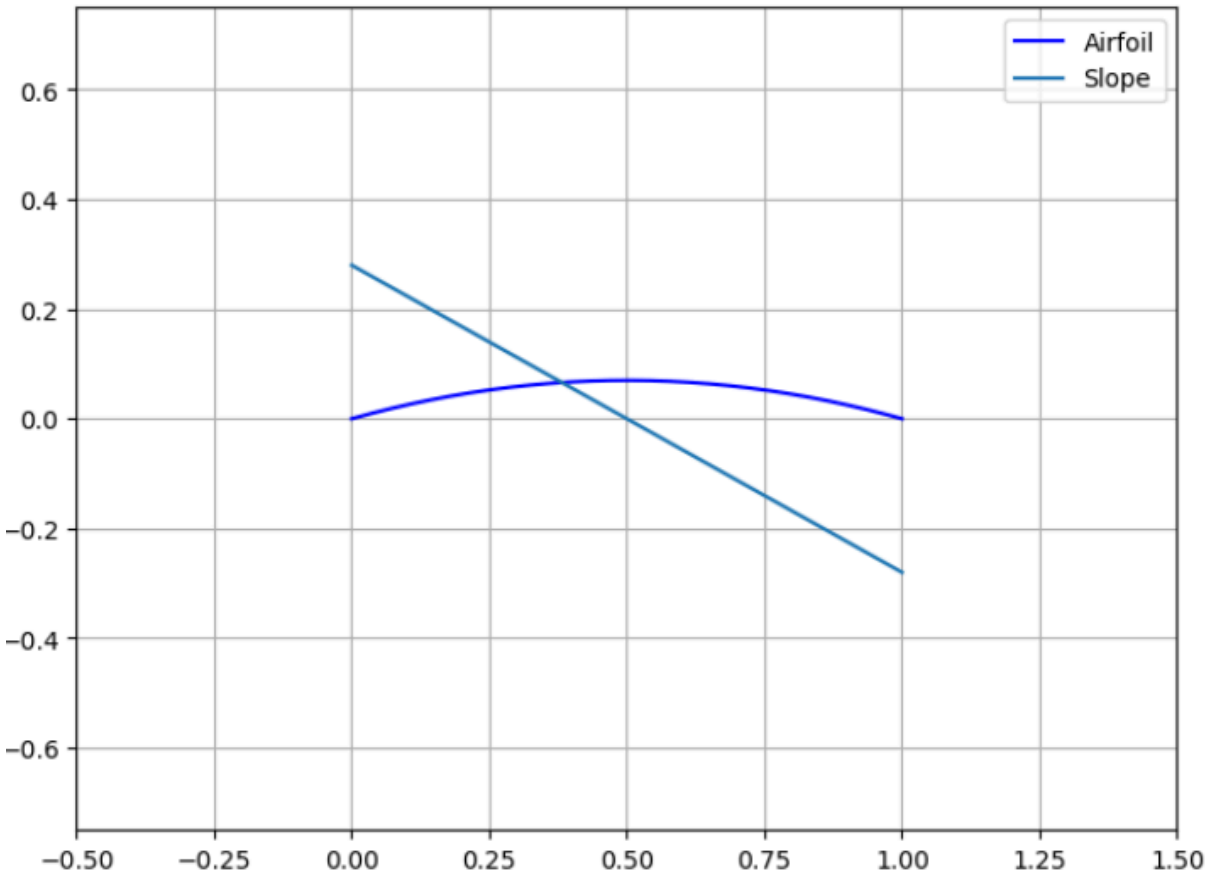
3. Airfoil Simulation and Results

3.1- *Camber line (‘y’ vs ‘x’) for the NACA airfoil*

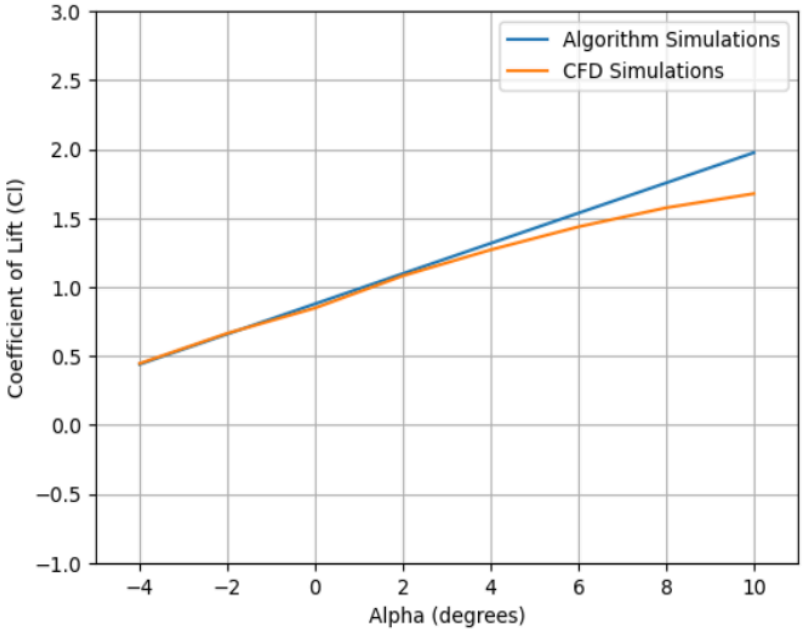
NACA 7511	
Maximum Camber	***7.8% of chord***
Maximum Camber position	***50% of chord***
Maximum Thickness	***11% of chord***



3.2- Slope of the camber line along x (dy/dx vs x)



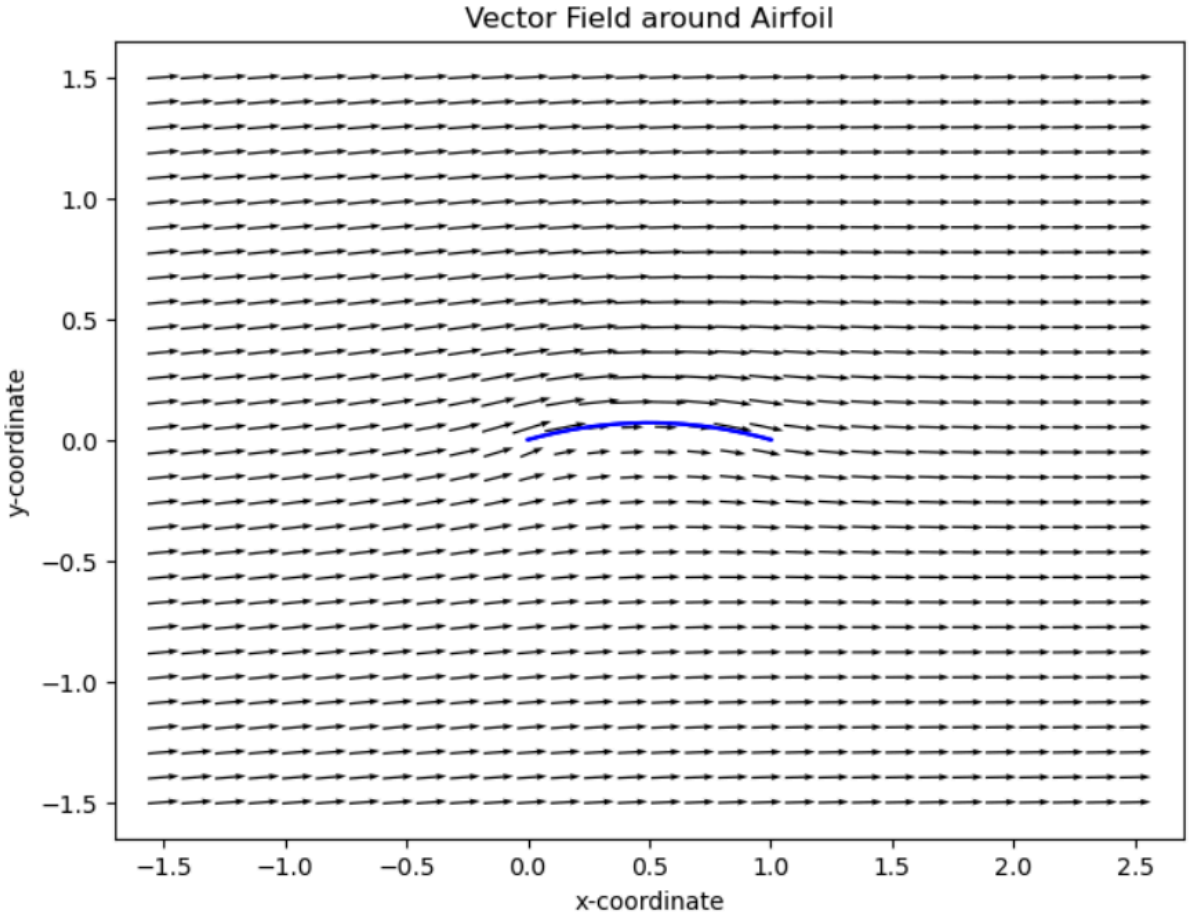
3.3-Cl vs α for the NACA airfoil from the program and CFD simulations on same chart



3.4- Observations from results obtained in (3.3)-

1. *Algorithm Results: When we compare the plots generated by our written algorithm with those from CFD simulations, we notice that our algorithm tends to show the data slightly higher up on the graph.*
2. *Coefficient of Lift: The values of lift coefficient (Cl) derived from our algorithms appear to be on the higher side.*
3. *Linear Plot: Our algorithm produces a linear plot, which means it assumes certain conditions based on thin airfoil theory*
4. *Impact of Flow Separation: Our algorithm doesn't account for flow separation, which means that as we increase the angle of attack, the Cl values keep rising without reaching a peak.*
5. *Effect of Flow Separation on Cl vs Alpha Curve: If we observe the Cl vs Alpha curve, we can see that after a certain angle, the slope decreases due to flow separation phenomena, which our algorithm doesn't consider.*

3.5- Vector field plot around the airfoil (domain size: 4c times 3c) at α = 3°



3.6- Vector field plot discussions-

1. Tangential Airflow: The reason the air flows tangentially along the airfoil is because of friction effects.
2. The arrows indicate the airflow direction at their tail, representing the airflow at a specific point.
3. When the camber is positive , it tends to create a circulation pattern that's clockwise. This circulation affects how air flows around the airfoil.
4. Increasing the camber tends to increase the circulation around the airfoil. However, this increase doesn't account for flow separation in our algorithm.
5. The wing's structure influences airflow, resulting in greater velocity above its camber in comparison to below it.

3.7- Circulation using Line Integral around the entire airfoil at α = 3°

Circulation around the airfoil is determined by calculating a line integral within a rectangular area measuring 4c by 3c.

Circulation (Line Integral Approach) = 18.590

3.8-Bound circulation by integrating circulation distribution along the camber line at α = 3°

Circulation (Bound Circulation Approach)=18.009

3.9-Discussion on the values obtained in 3.7 & 3.8

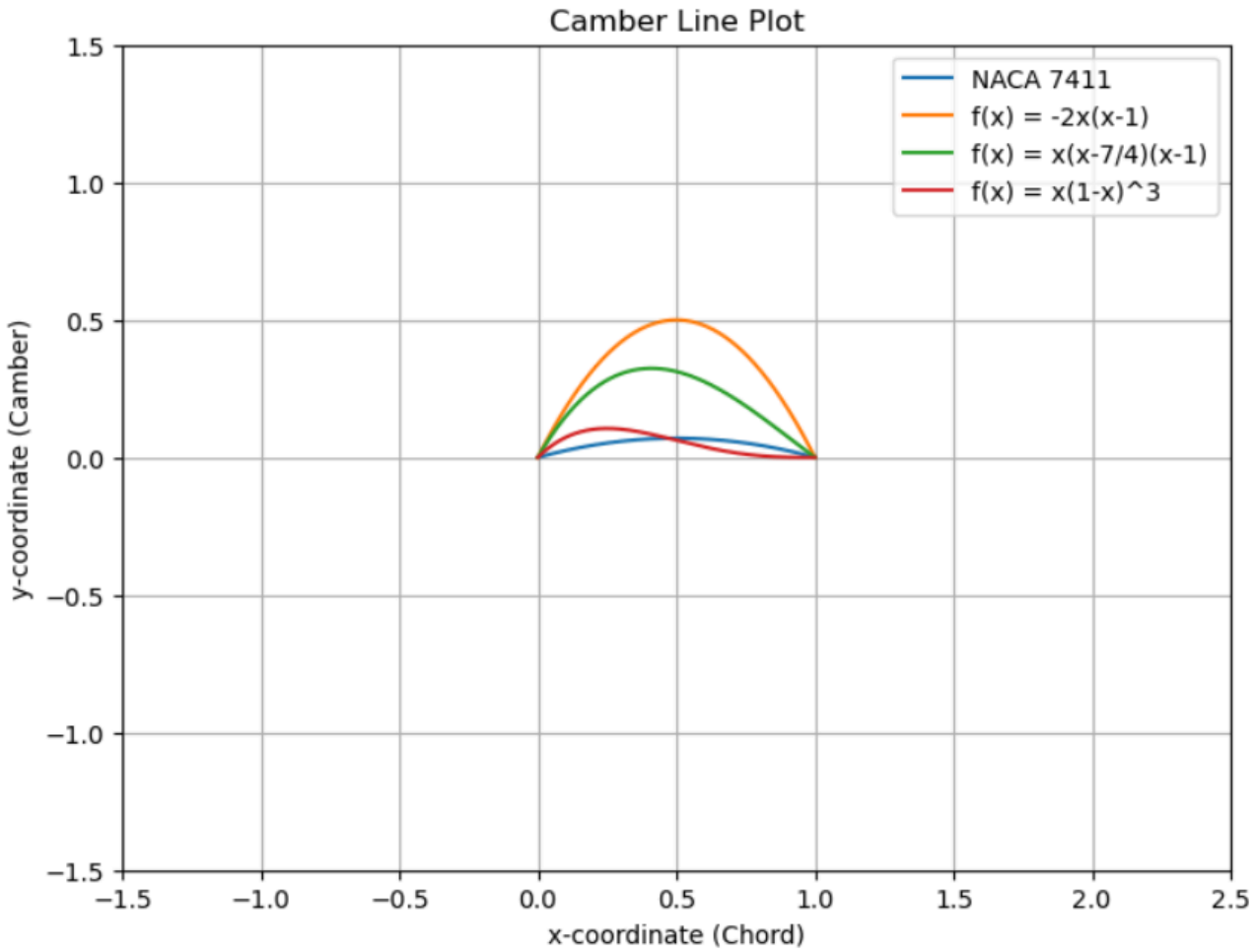
1. The circulation value derived from the distribution along the airfoil is lower compared to the circulation obtained by integrating along the rectangular region.

2. Actually, both circulations should be equal because they're generated by the airfoil itself. They must maintain the same value when integrated over a region encompassing it, ensuring the preservation of mass and momentum conservation principles.

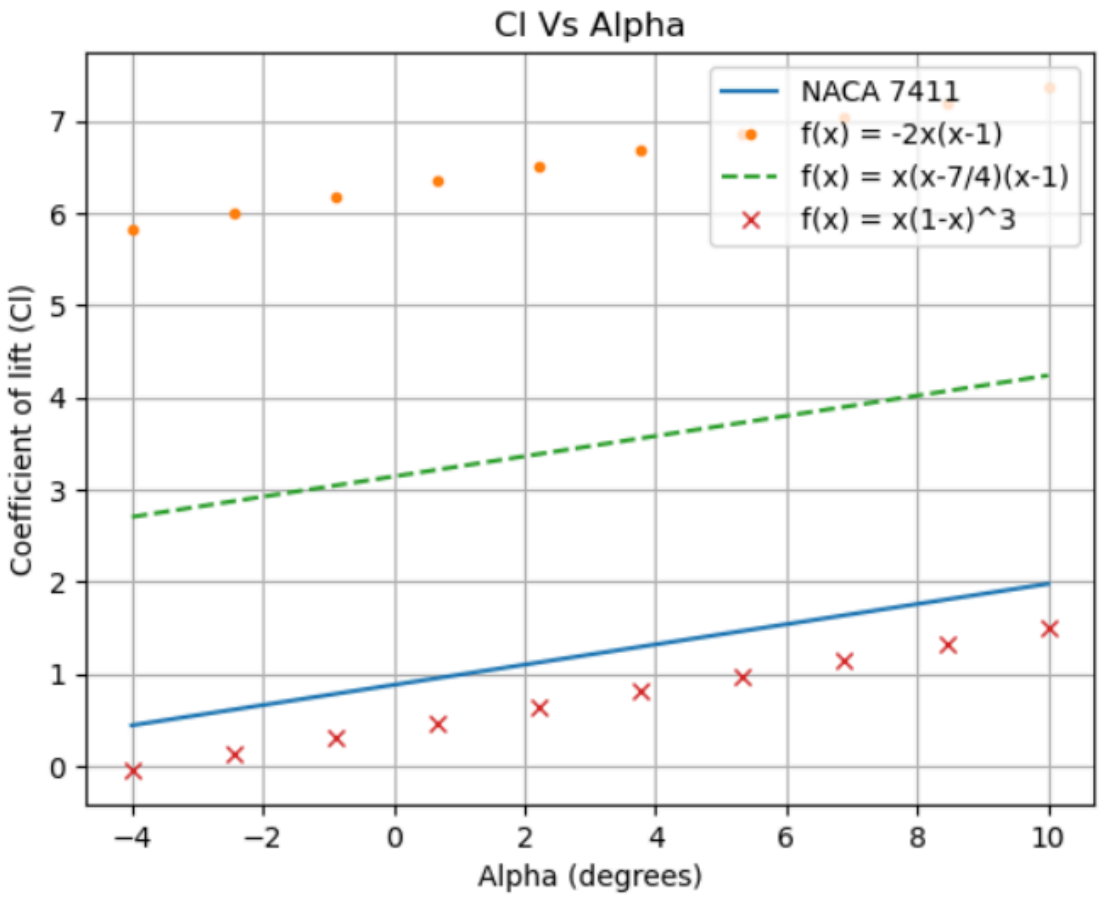
3. This difference occurs because we integrate the line integral by assuming ds as constant and we are just multiplying them with corresponding velocities at our boundary grid points and then summing over the complete boundary.

4. Novel Airfoil Properties

4.1 Camber line (‘y’ vs ‘x’) for my 3 airfoils and the NACA airfoil on same chart-



4.2- Cl vs α of the three airfoils along with that of the NACA airfoil on same chart

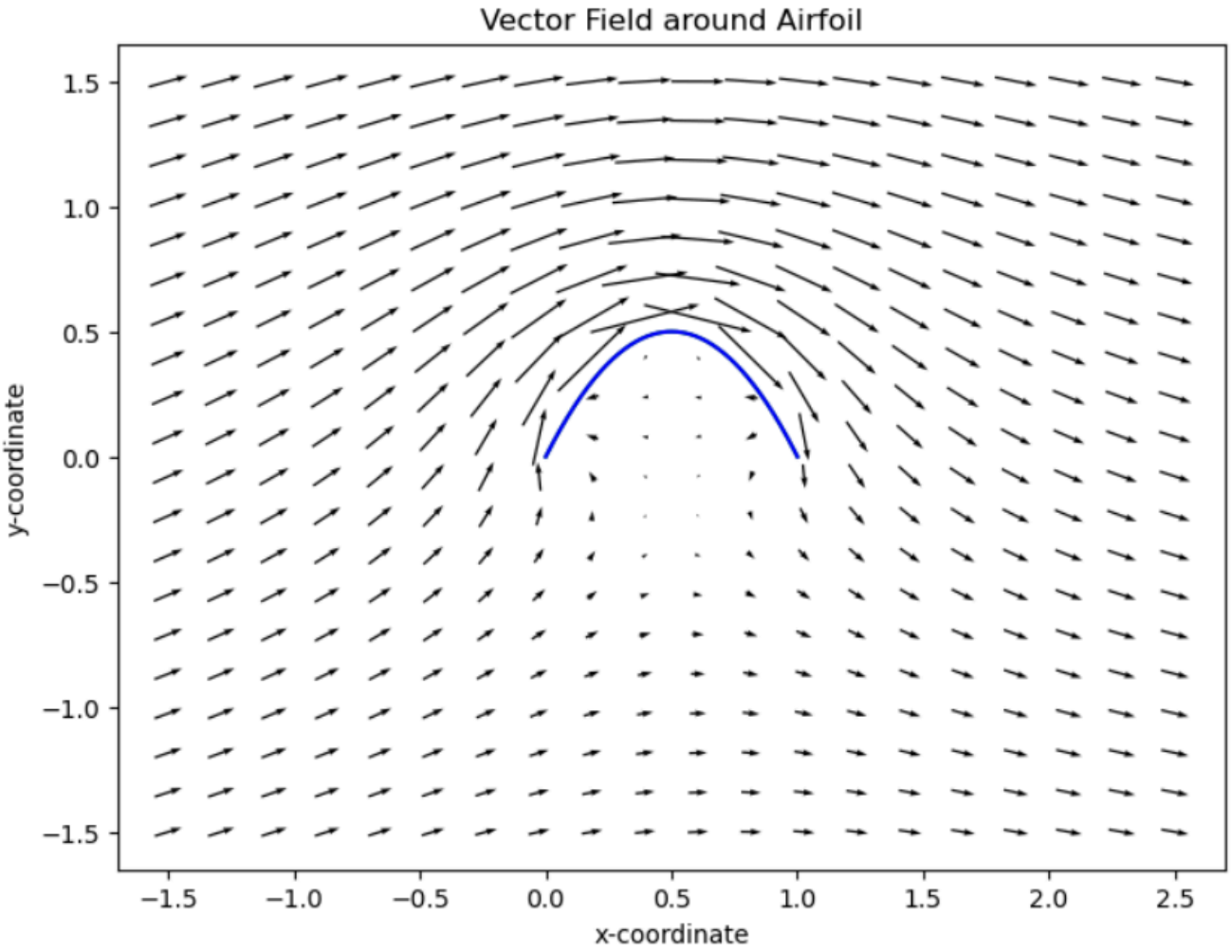


4.3 Discussion on the results obtained in (4.2)-

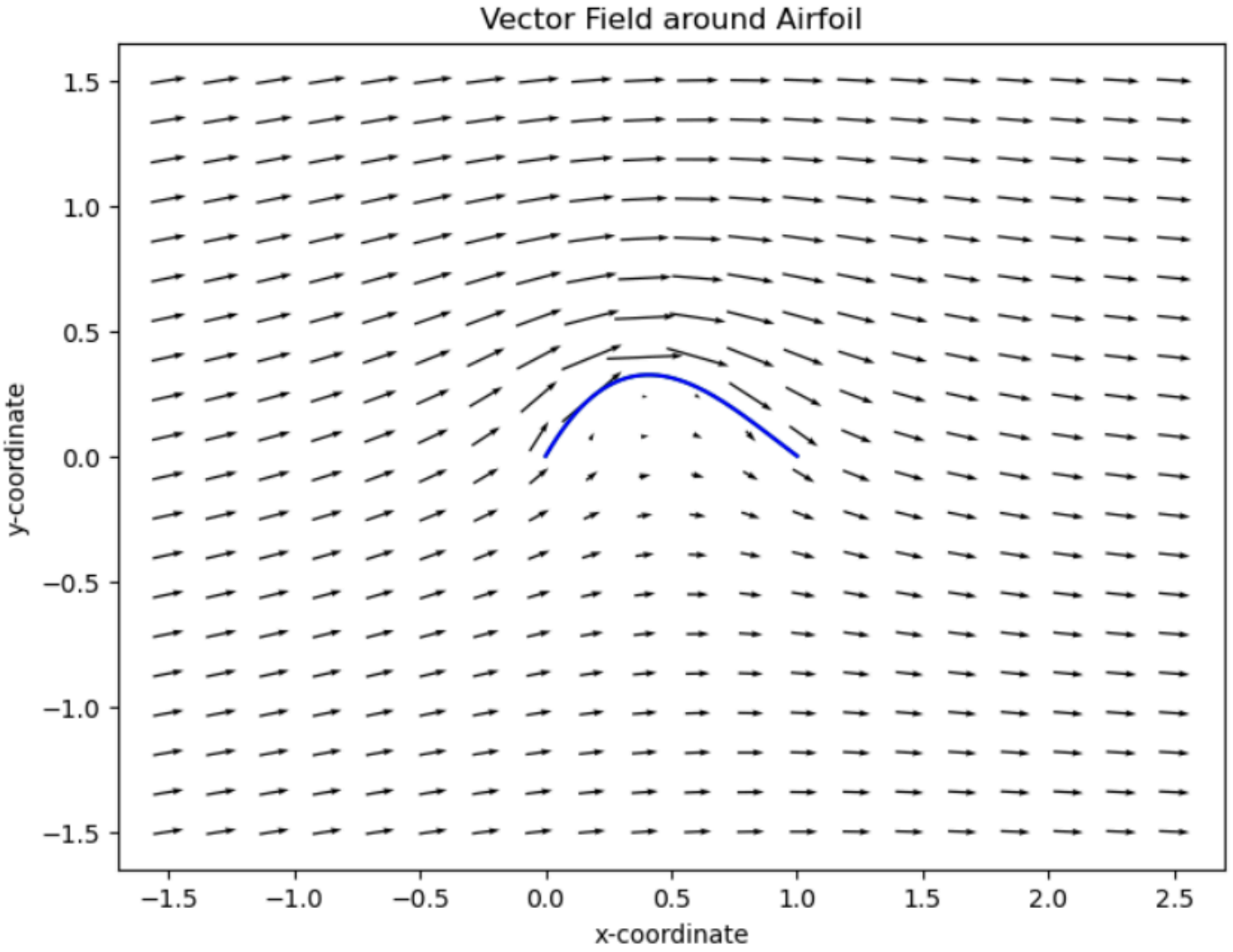
1. Based on Thin Airfoil Theory, the calculated Cl values reveal identical slopes in their curves, indicating a mere shift in their positions.
2. We can see that airfoils having more camber have more Cl values at specific angles of attack.
3. The reason for this is that an airfoil with more camber can generate a larger pressure contrast between its upper and lower surfaces, consequently leading to increased lift.
4. Plot of NACA7511 and function 3 have a similar plot and values for Cl vs alpha as there is not much difference in their shape.
5. We can see that function 1 has the maximum Cl values because of having more camber than other airfoils among them.

4.4 Vector field plot around the three airfoils (domain size: 4c times 3c) at α = 3°

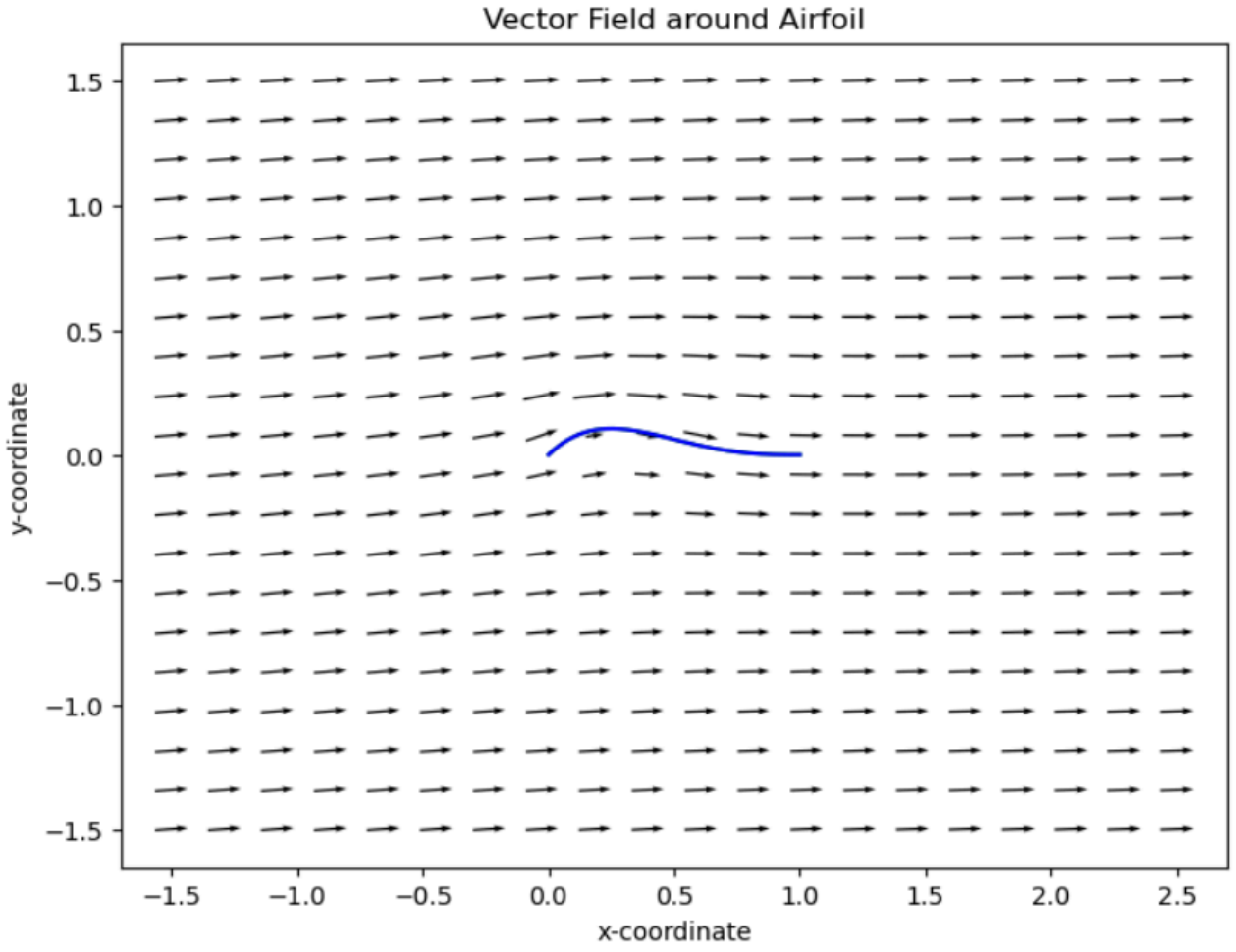
$$F_1(x) = -2x(x-1)$$



$$F_2(x) = x(x-7/4)(x-1)$$


Vector Field around Airfoil
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
y-coordinate
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
2.5
x-coordinate

$$F_3(x) = x(1-x)^3$$


Vector Field around Airfoil
y-coordinate
x-coordinate
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
2.5

4.5 Discussion on results obtained in vector field plots in (4.4)

1. Function 1 might not yield accurate results because it doesn't adhere to the thin airfoil assumption. At this point, the camber is positive, indicating a convex shape, resulting in a clockwise circulation. Additionally, the velocity of points above the camber exceeds that of points below it.
2. Function 2 its thickness is between the airfoil in function 1 and function 3. According to its graph it merely follows thin airfoil theory and might give approx results. Here the flow near the airfoil is less turbulent than in airfoil of function 1.
3. For function 3, the airfoil is thin and the thin airfoil theory will be more accurate. Camber of this is positive means it is convex , and it gives clockwise circulation, velocity of points above camber is more as compared below points.

5. Conclusion

5.1 Overall take on your code’s performance as compared to Ansys simulation, and possible reasons for deviations, if any

1. In our code, we're utilising Thin Airfoil Theory, which means it's more accurate when dealing with thin airfoils. However, if we're working with airfoils that have significant curvature (camber), there might be some inaccuracies in the results.

2. Unlike our algorithm, which focuses primarily on simplified airflow models, CFD software integrates sophisticated turbulence models to capture complex flow behaviours, including phenomena like boundary layer separation.

3. These advanced features significantly improve prediction accuracy, especially in scenarios where turbulent effects heavily influence aerodynamic performance.

4. Programs like Ansys and other Computational Fluid Dynamics (CFD) software may provide better results because they're designed for a wide range of airfoils, including those with substantial curvature.

5. These CFD programs employ advanced computational techniques such as meshing, finite volume, and finite element methods. These techniques enable a more detailed analysis of airflow around complex airfoil shapes, resulting in more precise predictions of aerodynamic behaviour.

5.2 Overall take on the performance of your airfoil as compared to the NACA airfoil

1. Function 1 Airfoil has more coefficient of lift than others because of having more camber.
2. By applying thin airfoil theory to our algorithm we reduced computation time and power as required for Ansys and CFD.
3. Plot of NACA7511 and function 3 have a similar plot and values for Cl vs alpha as there is not much difference in their shape.

6. Acknowledgements

1. Ghoshank Nnahe (22b0073)
2. Shrivardhan Kondekar (22b0054)
3. Nikhil Jha (22b0002)

7. References

1. Plotting Vector Fields in Python · Ajit Kumar (krajit.github.io)
2. http://airfoiltools.com/search/index
3. https://stackoverflow.com/questions/31815041/plotting-a-naca-4-series-airfoil
4. web.stanford.edu/~cantwell/AA200_Course_Material/The%20NACA%20airfoil%20series.pdf
5. https://pypi.org/project/airfoils/